SISTEMA INFORMÁTICO BASADO EN HARDWARE EMBEBIDO PARA LA DEPURACIÓN DE FUNCIONES ELECTRÓNICAS VÍA WEB

ALEXANDER VERA TASAMÁ

Tesis de grado presentada como
Requisito parcial para optar al título de
Doctor en Ingeniería
Énfasis en Ingeniería Electrónica

Director
ÁLVARO BERNAL NOREÑA, Ph.D

UNIVERSIDAD DEL VALLE
FACULTAD DE INGENIERÍA
PROGRAMA DE DOCTORADO EN INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA
GRUPO DE ARQUITECTURAS DIGITALES Y MICROELECTRÓNICA
SANTIAGO DE CALI
2015

A Dios, Fuerza vital en cuerpo y espíritu

A mi hija, María Alejandra, luz y aliento de mi vida

A mi esposa, Noemy, el amor de mi vida y mi sustento

A mi madre, Yolanda†, mi ángel guardián y ejemplo de vida

A mi padre, Jorge, amigo y patrocinador de mis sueños

A mis hermanos, siempre consejeros y cómplices

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

RESUMEN

El aprendizaje semipresencial se ha consolidado en el sector educativo durante los últimos años, dada la evolución de las Nuevas Tecnologías de la Información y de las Comunicaciones NTIC. Para *virtualizar* los espacios de aprendizaje se requieren plataformas que estén acordes con los modelos de enseñanza que se sugiere para cada campo. Particularmente en ingeniería, las didácticas experienciales son esenciales e involucran laboratorios, simulaciones y verificaciones entre otros aspectos metodológicos.

Los laboratorios remotos permiten a los estudiantes la realización de prácticas con experimentos relativos a campos específicos. Muchos de los trabajos de investigación que se desarrollan actualmente fomentan la implementación de modelos de enseñanza/aprendizaje alternativos. Esto constituye un reto para la educación en ingeniería, pues sus contenidos y actividades altamente prácticos son difíciles de implementar en el aprendizaje a distancia. Este trabajo presenta una arquitectura para un laboratorio remoto de diseño de sistemas digitales en ingeniería, que apoya la enseñanza semipresencial y el aprendizaje activo, introduciendo aspectos novedosos en el hardware y a nivel de aplicación. Los usuarios pueden acceder a una placa remota de entrenamiento que está basada en un FPGA (*Field Programmable Gate Array*), pueden ejecutar pruebas mediante un módulo analizador lógico, e interactuar con otros usuarios en línea a través de los servicios Web 2.0 que se integran al sistema. El principal aporte de este trabajo se centra en la plataforma *hardware*, para la que se concibe, diseña e implementa un analizador lógico con base en criterios de operación de usuarios académicos de la ingeniería electrónica. Así mismo, se introduce en la placa un FPGA y sus subsistemas de control para su disponibilidad total como recurso configurable. El recurso del sistema para la fase de verificación en el ciclo de diseño lo realiza el analizador lógico, que se conecta con el FPGA, pero la configuración de sus parámetros es realizada por el usuario. El soporte para ello se implementa en hardware y representa una variación importante de las arquitecturas típicas para estos sistemas en el contexto de sistemas embebidos. Los parámetros técnicos y de usuario se consideraron en el desarrollo de la aplicación *software* del sistema informático propuesto, en la que se integraron un gestor de aprendizaje (*Moodle*) y un gestor de contenido (*Joomla*).

ABSTRACT

Blended learning was strengthened in education in recent years, because of the technological trends and developments in the Information and Communications Technology ICT. Some software platforms are required for the learning places virtualization. These platforms must be consistent with teaching models, which are suggested from each field. Experiential teaching on engineering is important and involves experiential labs, simulations and verifications between others methodological topics.

Remote laboratories allow students to practice the experiments related to specific fields. Many research works are still being developed, which encourages the implementation of alternative teaching/learning models. This is a challenge for engineering education because its highly practical contents and activities are difficult for carrying in distance learning. This work presents an architecture for remote laboratory of Digital Systems Design focused on supporting blended teaching and active learning. Learners may access to a remote training board based on a Field Programmable Gate Array (FPGA); then, they may execute tests by a logic analyzer module, and interact with other online users by integrated Web 2.0 services. The main contribution of this work is in the hardware platform, which is conceived, designed and implemented with an internal logic analyzer, regarding some operating criteria based on academic users of electronics engineering. Also, an FPGA is introduced in the training board with its control subsystems. The logic analyzer is the verification resource which is available to debugging. It's connected to the FPGA, but its setup is performed by remote users. Both technical and users parameters were considered in the developing of the software application for the proposed system, where a learning management system (LMS – moodle) and a content management system (CMS – Joomla) were integrated.

RECONOCIMIENTOS

La concepción y el desarrollo de esta tesis doctoral hubiesen sido imposibles sin el apoyo, la orientación, la confianza y tutoría del Doctor *Álvaro Bernal Noreña*, director del *Grupo de Arquitecturas Digitales y Microelectrónica de la Universidad del Valle*. Quiero expresar a él mis más sinceros agradecimientos por su acompañamiento en todo mi proceso de formación; por sus gestiones, consejos y comprensión; por su profesionalismo y personalidad.

Agradezco también al Doctor *Rubén Darío Nieto*, profesor del mismo grupo de investigación. Su disposición permanente, sus asesorías, consejos y sus palabras de aliento fueron esenciales durante mi permanencia en el grupo y desarrollo de mi investigación. Su compañerismo, confianza y amistad han sido, son y serán muy valiosas para mí.

Un agradecimiento muy especial al Doctor *Wilhelmus Van Noije*, profesor titular de la *Universidade de São Paulo*, quien brindó aportes y orientaciones importantes durante mi pasantía en el *Laboratório de Sistemas Integráveis*. Gracias también por su gestión y disposición, por permitirme participar del curso *Introdução ao projeto de sistemas VLSI em CMOS*, por facilitarme los medios y espacios requeridos durante mi estadía.

Gracias a todo el equipo del *LSI* en la *USP*, en especial a *José, Hugo, Lizbeth, Paulo, Bruno, Alejandro* y *Silvana*, por su compañerismo y su acompañamiento para conmigo, mi esposa y mi hija durante nuestra estadía en São Paulo.

Quiero hacer un reconocimiento especial también a *Colciencias* y a la *Universidad del Quindío*. Su financiación y apoyo fueron imprescindibles para mi formación doctoral. Gracias a todos mis colegas en la *Uniquindío*, que siempre han estado muy pendientes de mi evolución: *Alexander*, *Jorge Iván*, *Wilmer Diego*, *Francisco*, …. En especial agradezco a *José Fernando Echeverry* y a *Jaiber Cardona* que me apoyaron incondicionalmente para la aprobación de mi comisión de estudios.

Extiendo mi agradecimiento a todos mis compañeros del grupo de arquitecturas digitales y microelectrónica. A *Edison Ferney*, *Jorge Eliecer*, *Duván Fernando*, *Sergio Andrés* y *Nathan*, quienes hicieron aportes muy valiosos dentro de los proyectos de investigación y sembraron compañerismo, amistad y profesionalismo. También agradezco a *Nathaly*, *Oscar Darío*, *Andrés*, *René* y los *Anderson*, por su amistad y los gratos momentos.

Finalmente, hago un reconocimiento muy especial a mi familia; su perseverancia y apoyo incondicional fueron imprescindibles para la continuidad y finalización de este proceso. El amor, la paciencia y comprensión de mi esposa; el afecto, la sonrisa y lo abrazos de mi hija me impulsaron permanentemente. El acompañamiento y las oraciones de mi madre, quien en vida me vio emprender este proceso de formación; ahora me acompaña y guía desde el Cielo. Las palabras y confianza de mi padre, quien siempre ha creído en mí. Los mensajes de aliento de mis hermanos, que siempre han estado ahí…

Alexander Vera Tasamá

1. INTRODUCCIÓN

En la última década, múltiples trabajos centran los esfuerzos en reflexionar sobre la puesta en marcha de nuevos paradigmas formativos y en alcanzar el umbral de lo que debería ser una verdadera sociedad del conocimiento. Según [1], es evidente la necesidad de un modelo para espacios de aprendizaje *on-line*, lo que sugiere cambios en los procesos educativos. La adecuación a los nuevos medios tecnológicos, no como simples mediadores, sino como una parte destacada en los procesos de enseñanza-aprendizaje con objetivos, recursos, contenidos al servicio del desarrollo de capacidades y habilidades tanto personales como sociales, es el centro de estudio de trabajos como los desarrollados en [1], [2]. Estos trabajos buscan nuevos modelos conceptuales para el aprendizaje y la aplicación de estrategias didácticas con enfoque en el aprendizaje activo en los cursos básicos de ingeniería, dando mayor relevancia a la instrucción con participación, el desarrollo de competencias y la estimulación de la autonomía del estudiante.

Los laboratorios en línea se han convertido en un apoyo muy útil para los componentes prácticos y experienciales de los métodos de enseñanza en todo el mundo, desde que se tiene el soporte técnico y la disponibilidad tecnológica para su implementación. Existen dos clases principales de estos laboratorios en línea: Laboratorios remotos y laboratorios virtuales [3], donde la instrumentación para la ejecución de las prácticas estará disponible físicamente en forma real o será simulada respectivamente. La aplicación de ambos esquemas en la educación en ingeniería es evidente, por la importancia del aprendizaje experiencial, bien sea con entornos simulados o reales, que garanticen el ejercicio del ciclo de diseño siempre que soporten procesos de depuración o verificación funcional de los sistemas que el usuario desarrolla.

1.1 Contexto de la Tesis

El crecimiento de las sociedades del conocimiento e información y el consecuente desarrollo de las TIC ha promovido la legislación de varias naciones en un contexto tecnológico. Colombia no es ajena a esta tendencia [4], lo que le permitiría ampliar las posibilidades de contribución en este campo y aprovechar los recursos que se generen. Como consecuencia, es imperativo el aporte de cada región desde el campo tecnológico a diferentes sectores, entre ellos la educación.

Con la masificación de la Internet y en general de las NTIC, se han realizado estudios que evalúan el impacto de este fenómeno sobre la educación y las ventajas que se tienen en particular sobre las didácticas en la educación superior. Según *California State University – Northridge*, la experiencia de la educación virtual entrega resultados positivos frente a las didácticas tradicionales en el aula [5], usando técnicas similares pero, mientras la segunda es presencial, la primera es más autónoma y dirigida. Por otra parte, el Departamento de Educación de los Estados Unidos de América, ha realizado una revisión sobre las investigaciones desarrolladas alrededor del aprendizaje en línea entre los años 1996 y 2008 [6], mostrando principalmente que, además del crecimiento en la población estudiantil que se acoge a esta modalidad de formación, los resultados positivos se reflejan de manera especial cuando se incorpora la experiencia e interactividad en las aplicaciones instaladas sobre la plataforma.

En Colombia, alrededor de octubre de 1998 se funda el primer colegio virtual de Iberoamérica [7], con una propuesta pedagógica-metodológica basada en investigaciones de la Universidad de Harvard, en teorías constructivistas y la *Taxonomía de Bloom*. Así mismo, se han desarrollado laboratorios virtuales disponibles en *websites* para diferentes áreas de formación en Colombia y el mundo [8][9][10][11], lo que muestra la tendencia mundial a eliminar la necesidad de coincidencia espacio-temporal de educandos y docentes para múltiples campos del saber.

Desde el inicio de la educación en ingeniería y tecnología, los laboratorios han tenido un rol central en la formación de los ingenieros y tecnólogos [12], [13]. Las prácticas de laboratorio tradicionales son fundamentalmente un recurso para la modalidad educativa presencial; sin embargo, los laboratorios virtuales/simulados y los de experimentación remota están jugando un papel cada vez más importante [12], estableciendo un canal alternativo y complementario para el desarrollo de habilidades, pero que debe acompañarse de la orientación y tutoría del maestro. Los experimentos en el laboratorio definen un recurso que se torna esencial para estimular el aprendizaje práctico de la construcción de productos, procesos y sistemas, el conocimiento disciplinario y el aprendizaje social, como se plantea en [14]; por lo que deberían diseñarse con base en métodos de aprendizaje activo, estableciendo un modelo coherente con las didácticas propias del proceso de formación, de manera que maestros y estudiantes generen espacios significativos de aplicación y conexión de conceptos en escenarios diversos.

En los contextos nacional e internacional, se evidencia una fuerte tendencia a consolidar los laboratorios remotos en las diferentes áreas de formación para complementar la educación virtual, dada la importancia que tienen las didácticas basadas en experiencias y proyectos a nivel de educación superior, en particular de ingeniería. En la actualidad, se han reportado múltiples trabajos de investigación y desarrollo que implementan el soporte *hardware* y *software* de laboratorios de experimentación remota para la educación en diferentes disciplinas de la ingeniería [3], [15], [16] En general, estos aportes muestran la capacidad de las plataformas propuestas para albergar montajes modulares, según los tópicos y objetivos de las prácticas; así mismo, algunos discuten sus metodologías de implementación y aceptación en la comunidad académica donde se desarrollan.

Sin embargo, más allá del soporte físico y capacidad de los laboratorios, son necesarios el diseño e implementación de recursos didácticos suficientes para dotar los espacios de trabajo que definen los *WebLabs* en ingeniería electrónica, de manera que apoyen y estimulen el aprendizaje práctico para que el estudiante aprenda a diseñar, implementar y operar productos, procesos y sistemas. Esto debe hacerse obedeciendo a didácticas experienciales que involucren estrategias de aprendizaje basado en proyectos, colaborativo y cooperativo [17].

La iniciativa CDIO (*Conceiving, Designing, Implementing, Operating*) constituye un marco importante que proporciona a los estudiantes una educación enfatizada en los fundamentos de ingeniería en los contextos de concebir – diseñar – implementar – operar sistemas y productos del mundo real [14]. La adopción de un programa de estudios (*syllabus*) [18] y los estándares en que se encuentra estructurada la iniciativa establecen un nicho importante para el aporte desde el desarrollo tecnológico a la educación en ingeniería; esto sucede especialmente con los estándares 6 y 8 de CDIO que se enfocan en los espacios de trabajo para el diseño y la implementación, como los laboratorios de experimentación, y los métodos de enseñanza-aprendizaje respectivamente.

La iniciativa CDIO está estructurada en la adopción de 12 estándares [14], enfocados a 6 aspectos: Contexto (Estándar 1); Aspectos Curriculares (Estándares 2, 3 y 4); Experiencias y Espacios de trabajo para el diseño y la implementación (Estándares 5 y 6); Métodos de enseñanza y aprendizaje (Estándares 7 y 8); Habilidades de los docentes (Estándares 9 y 10); Evaluación (Estándares 11 y 12). Esta iniciativa ha sido

aceptada ampliamente a nivel internacional, lo que explica la vinculación de más de 120 instituciones de los 5 continentes al consorcio. Es necesario resaltar que ACOFI (Asociación Colombiana de Facultades de Ingeniería) ha manifestado su interés por promover esta iniciativa para la formación en ingeniería a nivel nacional.

En este contexto educativo de la ingeniería electrónica, existe una carencia de soluciones para laboratorios remotos que articule la operatividad técnica de los mismos con el carácter administrativo de la información que involucra la implementación de didácticas de aprendizaje experiencial, mediante la integración de gestores de contenido y aprendizaje. Esta primera hipótesis está sustentada en la revisión bibliográfica que se discute en el capítulo 2.

Frente a los laboratorios virtuales, los laboratorios de acceso remoto garantizarían una experiencia más próxima a situaciones de reales de ingeniería, pues permiten la teleoperación de un prototipo o planta física y una consecuente retroalimentación de resultados, cuya presentación puede realizarse en múltiples facetas con base en recursos de programación a nivel de aplicación, como bibliotecas y marcos de trabajo de software (*frameworks*). Estos mismos recursos son considerados por los laboratorios virtuales, pero su base de operación es sobre un modelo computacional que generalmente simula o emula un sistema. Por estas razones, la investigación expuesta en este trabajo se dirige sobre un sistema teleoperado, cuya concepción, diseño e implementación física se realizan con base en la operación por parte de un usuario académico partícipe de un proceso de formación en sistemas digitales.

La revisión bibliográfica sugiere también que las plataformas hardware de laboratorios remotos para este campo de formación tienen limitaciones a nivel de hardware para el cumplimiento del ciclo de diseño. En particular, los analizadores lógicos requeridos como instrumentos esenciales para la fase de verificación ocupan un área significativa si se usan dentro del mismo recurso configurable disponible para el usuario y, en ocasiones, simplemente se conecta al sistema un instrumento externo. Esta hipótesis relevante se discute también en el capítulo 2 y se desarrolla en los capítulos 4 y 5.

1.2 Contribución de la Investigación

Dado el carácter social de la educación y la necesidad del fortalecimiento permanente de este sector desde el punto de vista de optimización en el uso de los recursos, los resultados de este proyecto tienen un impacto importante en la medida que el sistema propuesto sea adoptado por instituciones de educación superior y/o media técnica, facilitando el acceso a laboratorios que incorporan TICs y soportan didácticas del aprendizaje activo, con una reducción en los costes de inversión en equipos de laboratorio. Sin embargo, como será discutido en el capítulo 2, la pertinencia sobre las disciplinas específicas y la implementación de las herramientas o recursos que se colocan a disposición de los usuarios en el aula son aspectos que abren aún múltiples nichos de investigación y desarrollo.

Como resultado de este análisis, desde la disciplina de la ingeniería electrónica para la formación de competencias en el diseño de sistemas digitales, en esta investigación se proyecta el desarrollo de un sistema informático en este contexto, que permita la implementación y depuración de funciones lógicas electrónicas. Este sistema informático, disponible en http://digitestlab.univalle.edu.co, facilita el uso de didácticas experienciales de laboratorio y desarrollo de habilidades en diseño, implementación y operación de sistemas con base en la programación y prueba remota de funciones lógicas en dispositivos programables, mediante una plataforma *hardware* o placa de experimentación conectada a la Internet.

Los principales aportes de esta investigación se enmarcan en el desarrollo tecnológico derivado del sistema informático, envolviendo esencialmente un aspecto a nivel de software, como soporte metodológico, y dos contribuciones a nivel de hardware, orientadas a garantizar los procesos de implementación y verificación de circuitos lógicos establecidos por los usuarios desde un sitio remoto. Las contribuciones de la investigación se describen a continuación:

- Integración e implementación de un gestor de contenido y un gestor de aprendizaje en la misma aplicación Web (Ver Figura 1.1) dentro del sistema informático con perfiles de administrador y de usuario. Esta integración facilita la organización de contenidos académicos, planificación de actividades y de acceso a la placa de experimentación. Así mismo, integra servicios Web 2.0, entre los que se destacan foros, redes sociales, chat y video, lo que facilita la implementación de estrategias de aprendizaje colaborativo y dinámicas de grupo mientras se ejecuta una práctica de laboratorio. En el capítulo 3 se describen los detalles y contexto de esta contribución, así como la propuesta de diagrama de flujo para la operación del sistema. Es conveniente resaltar que la mayoría de los gestores de aprendizaje incorporan soporte de servicios Web 2.0, razón por la cual resulta muy útil la explotación de sus recursos al integrarse con un gestor de contenido.
- Desarrollo de una placa de experimentación con un sistema embebido basado en FPGA, con soporte *hardware* para su configuración total o parcial desde una estación remota e interfaces para depuración configurable de 8, 16 o 32 bits a través de un analizador lógico descrito en alto nivel (*Verilog*) (Ver capítulos 4 y 5). El principal recurso disponible en el sistema informático para la programación de funciones lógicas de usuario es el FPGA, cuya área queda disponible totalmente para su diseño, como consecuencia de eliminar la necesidad de compilar y sintetizar un analizador lógico embebido en su proyecto para los procesos de depuración. La placa de experimentación propuesta en la sección 4.2 es administrada por un microcontrolador de 16-bits en el que reside un sistema operativo de tiempo real (*RTOS*), dando soporte a la implementación de funciones, tareas o módulos adicionales. Este recurso hardware constituye una de las contribuciones más importantes de esta investigación ya que las unidades que lo componen han sido diseñadas y dispuestas de manera que sean viables la programación y depuración de los circuitos de usuario, garantizando también la disponibilidad del 100% del FPGA para el cliente remoto. Así mismo, algunas funciones como el RTOS fueron concebidas para facilitar la proyección de la plataforma hardware, lo que permitirá actualizaciones posteriores de su *firmware* e incorporación de otras funciones y/o circuitos.
- Diseño de la arquitectura de un analizador lógico (*Logic Analyzer Core*), que integra además de sus bloques esenciales, un módulo para la aplicación de estímulos sobre el sistema bajo prueba y un subsistema basado en comandos para su configuración y operación, en función de los principales requerimientos de depuración de los estudiantes de ingeniería electrónica en el desarrollo de sus habilidades en diseño. Usualmente, los analizadores lógicos se utilizan como instrumentos de verificación que se conectan físicamente a los nodos de interés en el circuito bajo prueba, con el fin de monitorear los niveles lógicos con base en condiciones de disparo. Sin embargo, algunos sistemas utilizan analizadores lógicos como "núcleos – *cores* - embebidos" que consisten en unidades que se sintetizan junto con las unidades a verificar en dispositivos configurables, especialmente FPGAs. Ambas alternativas de uso para este mecanismo de depuración tienen ventajas y desventajas si se consideran aspectos de coste, alcances, potencia, y tiempos de verificación. En el contexto educativo que impacta este trabajo, se consolida una contribución que alcanza un equilibrio entre la reducción de los tiempos de verificación para los usuarios del entorno académico, como también de los costos de la plataforma que lo involucra. Así mismo, el consumo de potencia se mantiene en tiempo de operación activo para los alcances proyectados desde el área

y disciplina de formación. En el capítulo 5 se describen los detalles de este aporte, que se constituye como un elemento relevante de la plataforma hardware que se expone en el capítulo 4.

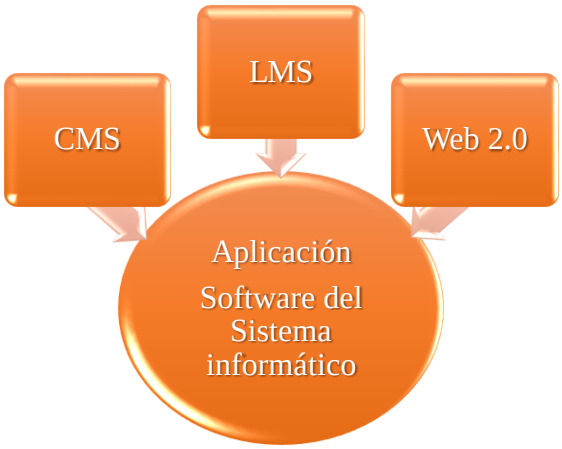


Figura 1.1 Integración de componentes software

1.3 Estructura del Libro

Este libro de investigación está conformado de la siguiente manera:

- Capítulo 2: Plataformas de Experimentación Remota en Ambientes Educativos de Ingeniería Electrónica
  En este capítulo se presentan algunos conceptos fundamentales de las plataformas virtuales para experimentación y una revisión de los principales antecedentes al respecto para el campo de la ingeniería electrónica. Así mismo, se exponen algunos conceptos y tendencias a considerar en el diseño del sistema informático, del cual se presenta también la arquitectura general.
- Capítulo 3: Desarrollo de la Componente Software del Sistema Informático
  Este capítulo describe en detalle las consideraciones y procedimientos de diseño de la componente software del sistema informático. Por tanto, se definen las especificaciones y requisitos del sistema en función de la aplicación Web, de los usuarios y de las metodologías de enseñanza soportadas.
- Capítulo 4: Desarrollo de la Componente Hardware del Sistema Informático
  En este capítulo se presenta el diseño e implementación de la plataforma hardware, también llamada placa de experimentación. Inicialmente se describen los aspectos técnicos esenciales de los componentes que hacen parte de la plataforma hardware, se muestra su modelo general, su diagrama esquemático y diseño del PCB. Finalmente, se expone la alternativa de implementación de soporte para el acceso simultáneo multiusuario al recurso hardware del sistema informático, considerando la reconfiguración parcial del FPGA.
- Capítulo 5: Diseño del Analizador Lógico Integrado
  Este capítulo presenta detalles de la arquitectura desarrollada para el analizador lógico de la placa de experimentación. En la primera parte se exponen generalidades de la configuración y funcionamiento de los analizadores lógicos como instrumentos, seguido de la descripción de los principales antecedentes en el contexto de los sistemas embebidos. Posteriormente, se describen los detalles del diseño de cada uno de los componentes de la arquitectura propuesta para el analizador lógico.
- Capítulo 6: Resultados y Conclusiones

Los principales resultados y conclusiones de la implementación del sistema informático se presentan en este capítulo, junto con las conclusiones, publicaciones y descripción de los proyectos de investigación, tesis de pregrado y posgrado derivados de este trabajo.

2. PLATAFORMAS DE EXPERIMENTACIÓN REMOTA EN AMBIENTES EDUCATIVOS DE INGENIERÍA ELECTRÓNICA

Desde el desarrollo de las TIC se aportan soluciones para ampliar el acceso a la experimentación con laboratorios remotos y virtuales. Así se conseguirán simultáneamente dos objetivos didácticos: (1) Realizar prácticas relacionadas con la asignatura ampliando la disponibilidad de los laboratorios y (2) formar estudiantes en el uso de las TIC [19]. Los laboratorios pueden clasificarse en función de dos criterios: (1) La forma de acceder a los recursos (local o remota) para propósitos de experimentación y (2) la naturaleza del sistema físico (real o virtual), con lo que los entornos de experimentación quedarían clasificados de acuerdo con la Tabla 2.1 [20].

Tabla 2.1 Clasificación de laboratorios según el tipo y forma de acceso [20]

Acceso / **Lab.**	**Real**	**Virtual**
Local	Laboratorios presenciales con plantas reales	Laboratorios presenciales con plantas simuladas y/o emuladas
Remoto	Teleoperación de una planta real	Laboratorio remoto con plantas simuladas y/o emuladas

La principal característica que diferencia a un laboratorio remoto de uno virtual es que detrás del laboratorio remoto hay *hardware* real. La persona que hace uso de ese laboratorio durante una sesión tiene el control físico de todos los recursos *hardware* involucrados en el experimento que está utilizando. Un laboratorio virtual emula el comportamiento del experimento mediante *software*. Utilizar un laboratorio remoto es por tanto una experiencia mucho más cercana a un uso real como en un laboratorio presencial, por lo que es capaz de sustituir a éste sin afectar negativamente la labor del usuario. Su principal desventaja es el coste que pueda implicar, puesto que los recursos utilizados deben existir físicamente. Sin embargo, esto representa una ventaja para la educación semipresencial y para mejorar la accesibilidad y experimentación de las personas con algunas limitaciones físicas, ya que existiría disponibilidad plena del experimento (aunque puede ser controlado por el profesor), eficiencia máxima en el tiempo de uso y el mantenimiento necesario es notablemente menor [21].

La idea básica de un laboratorio remoto es que un usuario se conecte a través de internet con un computador o dispositivo móvil desde un lugar A, a un experimento real que se lleva a cabo en un lugar B [22]. El núcleo de estos laboratorios remotos es, por lo tanto, el conjunto de instrumentos de propósito general y/o especializado, interconectados a un sistema de computadores con conexión a internet [23][24].

Por otra parte, como herramienta para la enseñanza se encuentran las plataformas virtuales, las cuales constituyen un conjunto de estructuras, políticas, técnicas, estrategias y elementos de aprendizaje que se integran en la implementación del proceso de enseñanza-aprendizaje, dentro de las instituciones educativas,

las figuras 2.1 y 2.2 presentan las principales características de las plataformas virtuales y los elementos que se deben considerar en éstas para la educación [25].

Figura 2.1 Principales características de las plataformas virtuales [25]

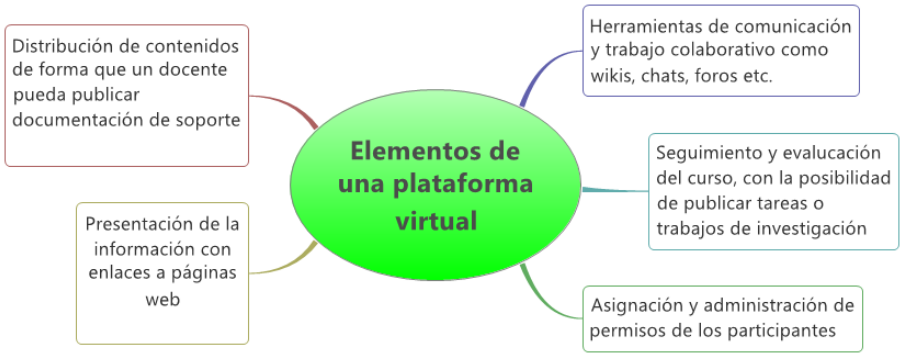


Figura 2.2 Elementos que una plataforma virtual debe considerar [25]

El desarrollo de laboratorios encontrados en el campo de la ingeniería electrónica es muy variado, ya que se han desarrollado interfaces de usuario con base en lenguajes de programación como HTML, PHP o JAVA y también en software avanzado como LABVIEW; en algunos casos, se implementa el uso de cámaras web para que el usuario pueda ver en tiempo real lo que sucede con el experimento. Para el transporte de la información remota como un protocolo de internet es comúnmente usado el TCP/IP ya que éste garantiza la confiabilidad de la información transportada del cliente al servidor y viceversa.

2.1 Antecedentes Principales en Ingeniería Electrónica

En educación superior, programas académicos de ingeniería, como electrónica, comunicaciones e informática han incorporado ya aplicaciones para el soporte de laboratorios virtuales como en [8]–[10], lo que contribuye a formación de un mayor número de estudiantes por espacio académico. Nada reemplaza la experiencia presencial en el aula, pero es difícil garantizar que los estudiantes participen siempre en este tipo de encuentros [26], razón por la que se han presentado ya modelos que usan TIC para mejorar las

experiencias de aprendizaje en modalidades presencial y distancia [27]. En niveles que implican aprendizaje experiencial resulta necesaria la vinculación de laboratorios remotos y técnicas para una adecuada implementación [16], que contribuyan el desarrollo de competencias en el campo específico, puesto que además de la experimentación es importante también el procedimiento de medida y manejo de instrumentación [26].

Actualmente, la enseñanza en particular de los Sistemas Digitales a nivel de ingeniería electrónica requiere de la realización práctica de ejercicios orientados a la simulación, implementación y prueba de circuitos lógicos ligados a un contexto específico [28]; por esta razón, surgen desarrollos en hardware y software orientados a la explotación de beneficios de las tecnologías relacionadas, como su reconfigurabilidad y modularidad, que facilitan la aplicación de ese tipo de didácticas para un aprendizaje exitoso en este campo.

Al respecto, en la Facultad de Informática de la Universidad Complutense de Madrid se propuso durante el año 2005 el desarrollo de un entorno hardware para la ejecución de aplicaciones de propósito general sobre FPGA [29], donde la implementación se hace sobre una placa de *prototipado* con FPGA de la familia *Virtex II de Xilinx*, incorporando módulos de entrada por teclado, salida VGA, planificación de tareas, memoria SDRAM, interfaz de red y BIOS (Ver Figura 2.3 y Figura 2.4). Sin embargo, aunque la configuración de la FPGA puede hacerse desde los módulos de entrada, las pruebas post-implementación de cualquier descripción o función lógica se harían mediante el uso de equipos externos como el analizador lógico. Así mismo, para su configuración remota es importante una interfaz de usuario amigable que guíe adecuadamente al educando y que sea personalizada por el instructor.

Una placa didáctica basada en FPGA fue propuesta en la misma institución para el año 2006 [30], orientada a prácticas locales de laboratorio con dispositivos de E/S básicos, apropiados para ese tipo de estrategias, como *displays*, banco de LEDs, pulsadores, teclado matricial, puertos PS/2, VGA, interfaz IDE, zumbador y parlante. En dicho proyecto se propuso trabajar con una FPGA de la familia *Spartan* II de *Xilinx* y a nivel de programación de interfaz de usuario, se hace sobre plataforma Windows en Visual C++.

No obstante los trabajos anteriores representan un aporte importante a nivel de las didácticas de enseñanza de los sistemas digitales, en la medida que facilitan la interacción del estudiante con el diseño, simulación e implementación de éstos, existe la limitación del trabajo local, al menos para las fases de implementación y test. El acceso remoto a estos recursos ha sido objeto de investigación en otros proyectos como aparecen en[11], [31]–[33], en los que se plantean arquitecturas Cliente-Servidor para el acceso a distancia a un ambiente de aprendizaje integrado que soporte la experimentación remota de sistemas. En [31] el sistema se desarrolla específicamente para cursos de diseño digital avanzado y procesamiento de señal usando plataformas FPGA, permitiendo a los estudiantes acceso total a ellas, a equipos de laboratorio y software licenciado, usando una aplicación con interfaz gráfica de usuario, basada en LabVIEW, a través de un navegador Web en un PC remoto. La instrumentación contemplada corresponde a analizador lógico, osciloscopio digital y generador vectorial de señales. Por otra parte, con [33] se desarrolla, mediante el proyecto DIESEL (*Distance Internet-Based Embedded System Experimental Laboratory*), un modelo de laboratorio de acceso remoto general para sistemas embebidos, como se muestra en la Figura 2.5, en el que las distintas estaciones experimentales de trabajo establecen una red LAN con el servidor administrador de laboratorio, el cual a su vez se encuentra conectado a Internet. En este modelo, cada estación incorpora un hardware y software adicional para la conexión y configuración de los instrumentos de prueba (*test*) conectados mediante interfaz GPIB, así como las tarjetas principales de propósito específico que se comunican con interfaz RS-232 o paralela.

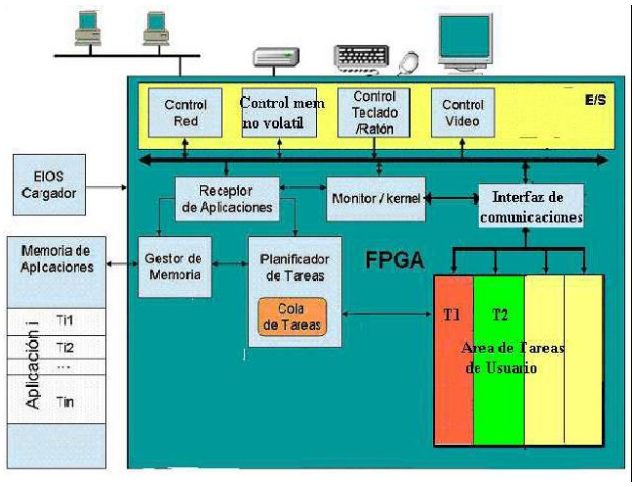


Figura 2.3Diagrama general del proyecto planteado en [28]

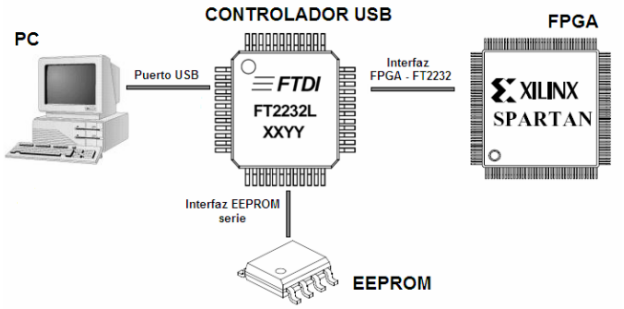


Figura 2.4 Diagrama general de bloques e interfaces de la placa didáctica propuesta en [29]

La complejidad y el poco tiempo en el mercado de los sistemas embebidos hacen que se requieran el uso de técnicas automatizadas durante la especificación, implementación y pruebas de dichos sistemas. Debido a los costos y sus restricciones temporales, la aplicación de soluciones de hardware específico a menudo es necesaria, haciendo del codiseño de hardware y software un tema importante para la automatización del diseño de sistemas embebidos [34]. Esto implica que podría concebirse la instrumentación necesaria para el test en laboratorios remotos como hardware modular embebido. Por ejemplo, las actuales herramientas de desarrollo de *Altera Corporation* [35] brindan el soporte de un analizador lógico embebido a través de una aplicación llamada por ellos *SignalTap*; no obstante, el utilizarla implica el consumo de recursos de la FPGA configurada con la aplicación de usuario.

Es evidente que los principales desarrollos propuestos, que han aportado en el campo de los laboratorios remotos en ingeniería electrónica, han centrado su atención fundamentalmente en dos líneas: la plataforma hardware y el modelo de integración sobre el que se colocan las aplicaciones. El proyecto *WebLab*-DEUSTO [36], uno de los más recientes, ha trabajado enfáticamente en la segunda de estas líneas, dando suficiente soporte a la primera para extenderse a cualquier disciplina en especial de la ciencia e ingeniería.

Este proyecto es basado en una arquitectura distribuida, como se muestra en la Figura 2.6, especificando sus propios protocolos *WebLab* (*Direct, UnixSocket, TCP Socket, Pickle over SOAP*) con el uso de *Phyton*. Dicha arquitectura la componen esencialmente los microservidores de experimentos, servidores de laboratorio, servidores de acceso (*login*), servidores de núcleo y las bases de datos implementadas en *MySQL*. Con este trabajo se han realizado algunos experimentos con dispositivos *Xilinx* CPLD XC9572, FPGA *Spartan*, *Microchip* PIC 18F97J60, conexión GPIB a un osciloscopio y soporte para el proyecto VISIR (*Virtual Instrumental Systems in Reality*) [37], un laboratorio remoto *OpenSource* desarrollado en el *Blekinge Institute of Technology*. El proyecto *WebLab*-DEUSTO representa un gran aporte en el desarrollo de plataformas software y de comunicación para el soporte de los laboratorios remotos en diferentes disciplinas de la ingeniería, dado el importante trabajo realizado a nivel de redes y su validación con diferentes experimentos con hardware, lo que implica mucho más trabajo por realizar a este nivel en otras áreas de la ciencia y la ingeniería.

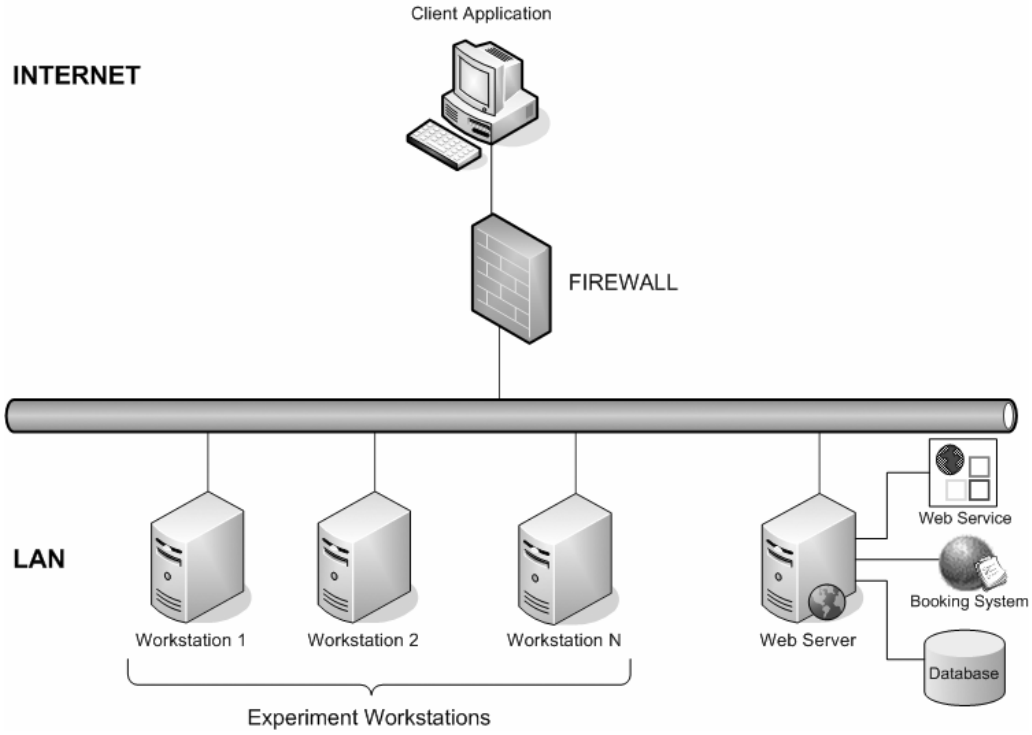


Figura 2.5 Arquitectura para el laboratorio de acceso remoto dado en [31]

El laboratorio presentado en [26] consiste en una arquitectura cliente/servidor, que soporta el test de circuitos y la conexión de fuentes de poder e instrumentos. Estos últimos cuentan con una interfaz antigua GPIB (*General Purpose Instrument Bus*), que permite la interconexión entre computadores e instrumentos. Los programas del servidor están escritos en *LabView 6i* y los de los clientes en *Visual Basic 6*; sin embargo, para la administración de estos recursos disponibles para los clientes, se requiere de múltiples módulos acoplados a cada computador, según la cantidad de instrumentos utilizables, lo cual puede disminuir también la eficiencia del sistema en cuanto a tiempos de respuesta o ancho de banda, dada la topología de bus usada.

Una arquitectura implementada en el curso *Computer Integrated Manufacturing Systems* en el *Engineering Management Department de la University of Missouri-Rolla*, se plantea como una infraestructura genérica para ambientes de aprendizaje en educación a distancia, con el uso de un controlador lógico programable

cuyo control y programación se soporta a través de un servicio Web [38]. Esta arquitectura se constituye de tres niveles (Ver Figura 2.7): El primer nivel lo conforman los computadores cliente, con terminales Windows 2000, Navegadores o Web Browsers, donde se desarrollan y ejecutan los programas de PLC sobre el modelo físico. En el segundo nivel se encuentra el controlador del sistema, con un PC con *Windows 2000*, *RSLogix* (paquete de programación de PLC), *RSLinx* (dispositivos y aplicaciones software), *Emulator* 500 (compilar programas PLC sin conexión), *Wonderware* (modelo animado del sistema); además en este nivel se encuentra otro PC, usado como Sistema de Soporte, con funciones primarias de servidor HTTP. El tercer nivel tiene el modelo físico del laboratorio de PLC, y lo constituyen inicialmente una banda transportadora equipada con sensores, varias luces y un cilindro de aire; además, el PLC SLC 500 de *Allen Bradley* se usa para el control del modelo físico. Como la mayoría de laboratorios remotos reportados en la bibliografía, presenta un conjunto de cámaras para suministrar video en vivo durante la ejecución de las aplicaciones, lo que se constituye en un elemento de retroalimentación muy útil para el estudiante o usuario remoto. Sin embargo, la instrumentación disponible para el estudiante está sujeta a los módulos que se puedan acoplar al PLC y que sean emulados a través del software de aplicación.

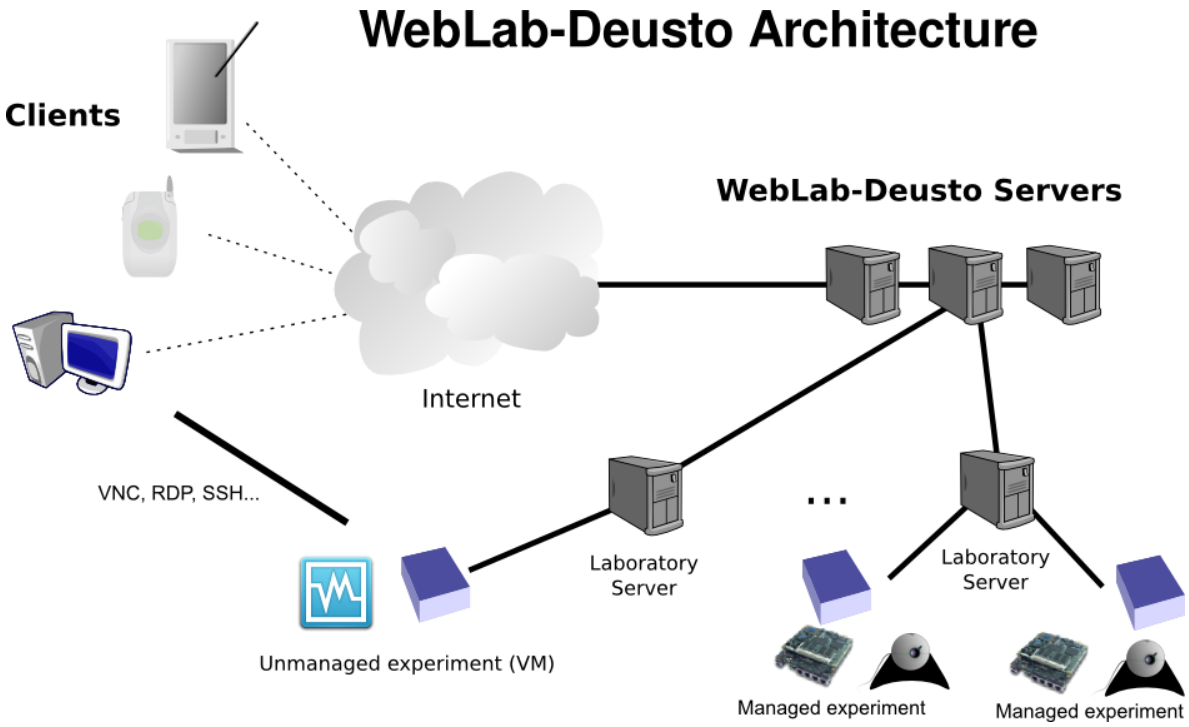


Figura 2.6 Arquitectura de *WebLab*-DEUSTO [36]

Otra propuesta bien interesante y bastante citada en la bibliografía corresponde a la del proyecto *RMCLab*, de la *University of Patras*. El sistema desarrollado se basa en arquitectura cliente-servidor (Ver Figura 2.8) y consiste de: cliente, cliente instructor, servidor de aplicación, servidor de recursos e infraestructura de laboratorio, incluyendo la instrumentación y los módulos de hardware [39]. El servidor de recursos cuenta con interfaces configurables hacia las señales de laboratorio (experimentos analógicos y digitales), basadas en bus LPT, e instrumentos con base en interfaz RS232. Potencialmente, este servidor podría soportar interfaces PCI, USB, entre otras. Múltiples tipos de circuitos pueden ser soportados en la plataforma de este servidor de recursos, tanto analógicos pre-configurados, como digitales o mixtos; por esta razón, se cuenta para este desarrollo con una *motherboard* capaz de manejar hasta 64 tarjetas, donde cada una tiene incorporado un FPGA y módulos auxiliares, como se muestra en la Figura 2.9. Cada tarjeta de estas incluye también un PLD para su direccionamiento y configuración del FPGA.

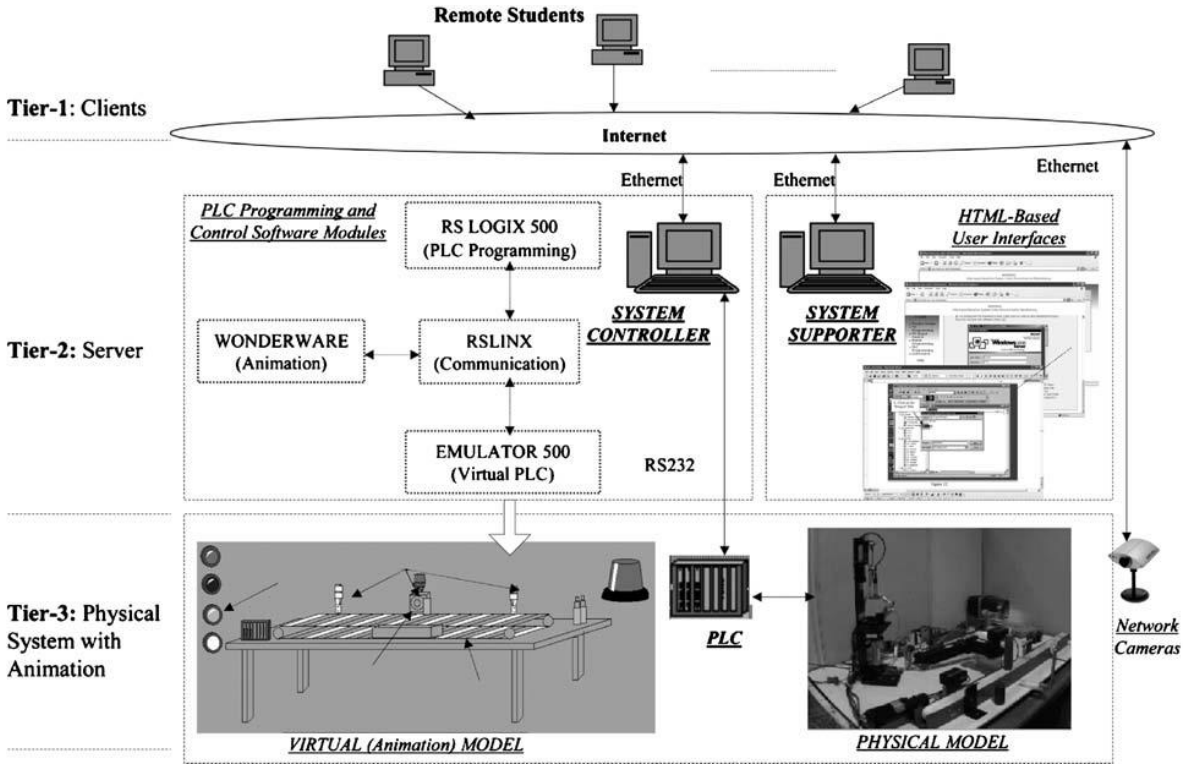


Figura 2.7 Arquitectura del sistema laboratorio de PLC basado en Web [38]

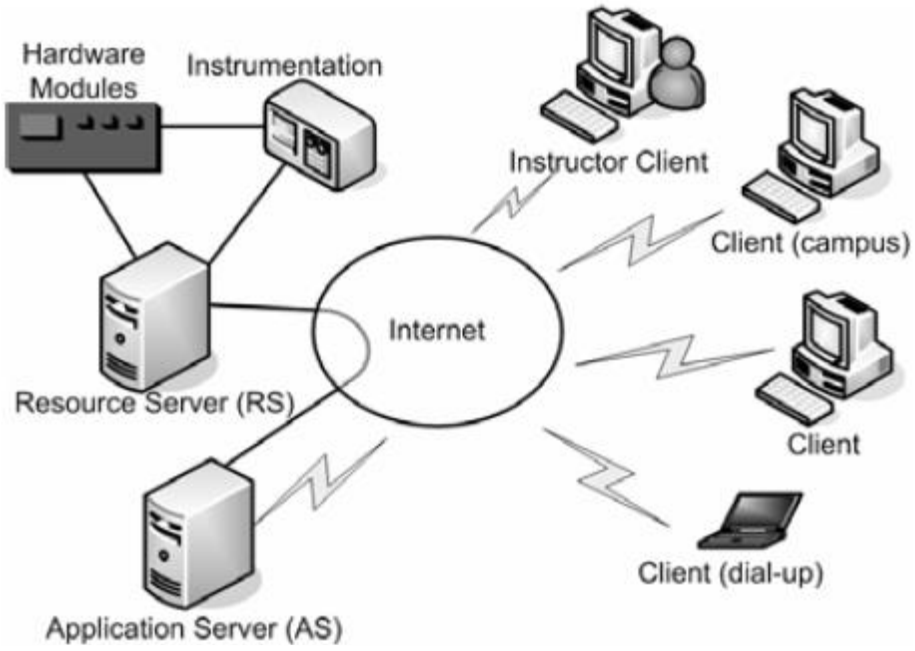


Figura 2.8 Arquitectura General RMCLab [39]

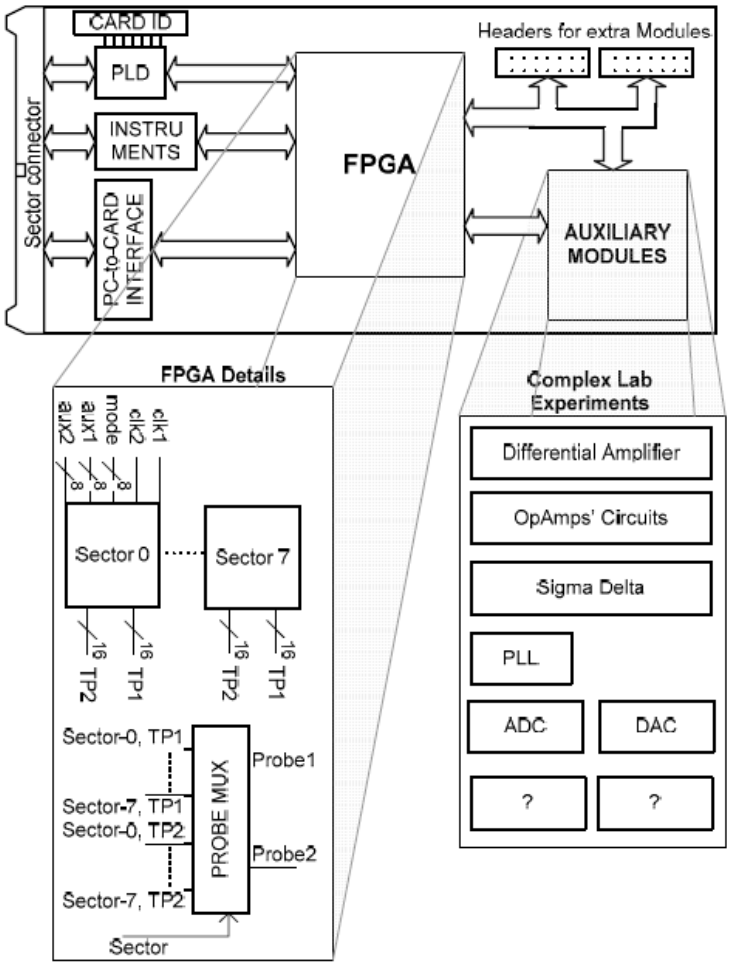


Figura 2.9 Arquitectura hardware para cada una de las 64 tarjetas en RMCLab [39]

Como se puede apreciar, en esta última propuesta que se constituye en la más completa en términos de desarrollo a nivel de hardware, se soportan múltiples áreas de la ingeniería electrónica, dado que cada una de las tarjetas que hacen parte del "*rack*" contiene *hardware* re-configurable y dispositivos de E/S específicos, según su función dentro del rango de experimentos pre-configurados. También, debe prestarse especial atención a que existe una interfaz para conexión de instrumentos, lo cual es esencial en el momento de realizar las pruebas del diseño implementado. No obstante, dado que el sistema tiene soporte para tantas tarjetas, pero cada una de ellas tiene interfaces para conexión al bus e instrumentos, si se suministra la suficiente potencia para la operación de todas, existe la posibilidad de tener múltiples instrumentos trabajando también separada y simultáneamente; esto quiere decir que, como no se referencia la posibilidad de instrumentos compartidos, debe considerarse una cantidad suficiente de cada uno de estos (voltímetros, osciloscopio, generador de funciones, analizadores lógicos, etc.) para interactuar uno a uno con cada tarjeta.

Otros trabajos publicados, orientados específicamente a las áreas de sistemas digitales, conciben laboratorios remotos con arquitecturas similares a las expuestas arriba, pero usando plataformas *hardware* basadas en FPGA con diferentes fines. En *Northern Illinois University*, se presentó en el año 2007 el proyecto FPGA *e-Lab*, el cual desarrolla un sistema compuesto por hardware y un "Protocolo de Laboratorio" [40], basados en un *kit* de entrenamiento de FPGA Spartan-3E, *hardware* de adquisición de

datos de *National Instruments* e instrumentos como osciloscopio y generador de funciones; mientras que, en lo correspondiente al *software* usan el LabView, suministrado por NI. Con el trabajo expuesto en [41] se muestra el uso de una plataforma similar para un laboratorio de sistemas electrónicos digitales con aplicaciones industriales, en la que, usando una tarjeta de entrenamiento FPGA XCV800, se configuran bloques funcionales interconectados desde el cliente; de manera que, a través de un entorno basado en el proyecto *opensource Ptolemy II* (desarrollado en Universidad de California), se soporta concurrencia de actores e interconexión de bloques para simulación e implementación. Por otra parte, en [42][43] se presenta el desarrollo de un sistema de laboratorio para el diseño y experimentación de circuitos digitales, basándose en el proyecto TDeLMS (*TopDown eLearning Management System*) y en el entorno integrado de desarrollo FPGA *Advantage de Mentor Graphics Corp*.

En la Universidad de *Bahrain* se ha presentado también la arquitectura de un hardware para laboratorio remoto, basado en FPGA, orientado al diseño de circuitos electrónicos digitales[44]. Para este trabajo han usado una tarjeta de desarrollo de *Diligent Spartan 3E* de *Xilinx,* conectada a través de puerto USB a un PC; sin embargo plantean una red local de veinte (20) unidades (PC-tarjeta), a la cual los estudiantes pueden ingresar desde un sitio remoto y hacer su simulación e implementación con ayuda de la herramienta *Xilinx ISE Foundation*.

En los trabajos anteriores [40]–[44], se presentan topologías relacionadas y ligeramente similares, destacando que sus plataformas hardware involucran generalmente arquitecturas reconfigurables para programar bloques funcionales, conforme a la descripción del cliente o a librerías de soporte del desarrollador. Varios trabajos adaptan a sus modelos arquitecturales, tarjetas de desarrollo o de entrenamiento ofertadas y distribuidas comercialmente por los fabricantes y proveedores con trayectoria en el campo, como por ejemplo *Xilinx* y *Altera*. Así mismo, cada una de los proyectos presentados considera la adopción de un software entorno integrado de desarrollo para la edición, compilación y soporte de aplicaciones de acuerdo con los servicios ofertados.

Por otro lado, en la Universidad Estadual de Campinas se presenta una arquitectura para laboratorio de acceso remoto con aplicaciones educativas orientado a la enseñanza de ingeniería electrónica [45][46]. En este proyecto se plantea el desarrollo de una placa de control, una de experimentación y unos bloques electrónicos como elementos constituyentes del subsistema de hardware, basándose en un microcontrolador PIC16F877, un circuito integrado con el protocolo USB FT245 y otros componentes lógicos para interfaces y E/S. Este *hardware* se interconecta a través de un sistema de buses y se comunica con un PC mediante enlace USB. Este PC, en conjunto con una placa controladora, cumple el papel de servidor GPIB, permitiendo el soporte para la conexión de un osciloscopio HP54503A a través de esta interfaz.

Otro de los trabajos más completos reportados en la bibliografía es el de la Universidad de Campinas [47], pues el desarrollo que se hace con base en la plataforma REDLART, muestra un esfuerzo significativo en la concepción desde la arquitectura para aplicaciones hasta la interacción con la plataforma *hardware*. Este trabajo especifica muy bien cada una de las capas que conforman el soporte para los servicios que debe ofertar el laboratorio, según la Figura 2.10, donde la capa superior *WebLab* es la que soporta finalmente las aplicaciones mencionadas. Sin embargo, aunque gran parte del desarrollo enfocado en este trabajo hace referencia al modelado de los esquemas de rutinas, servicios y protocolos adoptados, la plataforma REDLART (Ver *Figura 2.11*) es la que implementa, con base en FPGA, *cores* de procesamiento, almacenamiento, empaquetamiento/ desempaquetamiento e interfaces con módulos de E/S. No obstante, el trabajo omite el soporte explícito de instrumentación para pruebas post-implementación por parte del cliente o usuario remoto, diferentes a las convencionales que corresponden a las empleadas por sus autores durante los procesos de diseño e implementación.

Figura 2.10 Arquitectura propuesta para aplicaciones de Laboratorio Remoto basado en REDLART [47]

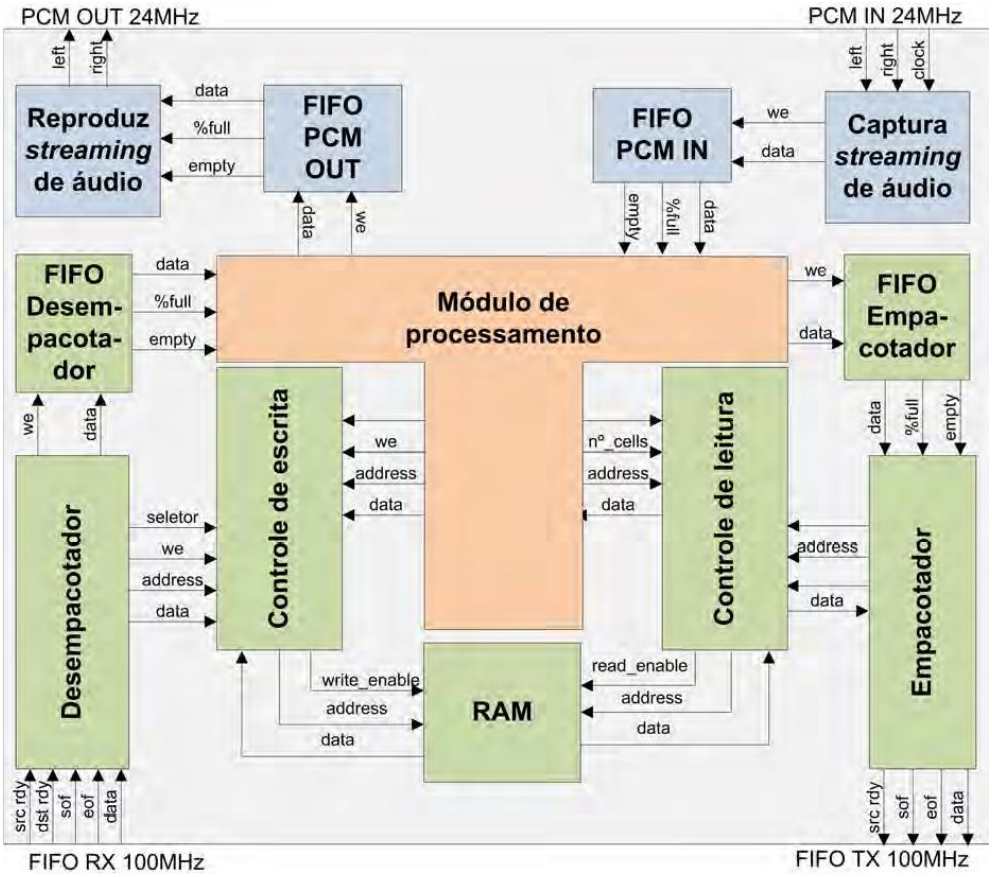


Figura 2.11 Arquitectura detallada de plataforma REDLART [47]

En la *Technical University of Kosice* [31] de la republica de Eslovaquia, los autores presentan un sistema de laboratorio remoto que permite el desarrollo de experimentos controlados a través de internet mediante una interfaz web, este sistema ha sido desarrollado para cursos de sistemas digitales avanzados y procesamiento de señales utilizando FPGA, permitiendo a los estudiantes acceso total a los equipos de laboratorio y a software especializado licenciado como LABVIEW, que permite la interface gráfica con el

usuario. Para el desarrollo de las prácticas remotas consideran que la instrumentación adecuada corresponde a un analizador lógico, osciloscopio digital y un generador vectorial de señales.

Uno de los proyectos que representan un antecedente inmediato para este trabajo es el *Desarrollo De Un Sistema De Monitoreo Y Control Electrónico Remoto Via Internet Basado En Lenguaje Java*, realizado en la Universidad del Valle [32]. El autor presenta el desarrollo de un sistema *software* que está orientado al monitoreo y control remoto de un sistema *hardware* para la integración de un laboratorio de electrónica a internet. El laboratorio tiene una interfaz Web que permite que sean cargados remotamente los parámetros de programación y obtener medidas de las señales de salida del sistema previamente programados.

Entre los más recientes aportes a este campo se destacan los esfuerzos por adoptar lineamientos estándar en la infraestructura de los laboratorios remotos. Las tendencias ratifican a los FPGA como base de las plataformas *hardware* configurables. Algunos autores proponen arquitecturas genéricas de *hardware*, *software* reconfigurables [48], compartiendo y reusando instrumentos y otros módulos pertinentes a la experimentación, para reducir los costos de implementación del laboratorio. Así mismo, estos trabajos sugieren dos enfoques de implementación donde su diferencia radica fundamentalmente en la implementación de un servidor embebido SoC (*System on Chip*) en el FPGA, o un micro servidor Web independiente que establece un protocolo de comunicación con la plataforma basada en FPGA. No obstante, los autores realizan una implementación minimizada donde configuran un *core* de un generador de funciones en un FPGA que se conecta mediante bus de expansión a una tarjeta de entrenamiento comercial y operan desde una aplicación Java que corre sobre una PDA.

La adopción del estándar IEEE 1451.0, que define un conjunto de interfaces de comunicación para conectar transductores inteligentes a sistemas basados en microprocesadores, instrumentos y redes, permitiría reducir los costos de implementación y constituiría un mecanismo viable para los laboratorios modulares reconfigurables. Los instrumentos y módulos [49] que harían parte de un laboratorio remoto con tales características serían parcial o totalmente configurados sobre arquitecturas como FPGAs dedicadas a esas funciones específicas; sin embargo, los *cores hardware* que se implementarían sobre los dispositivos reconfigurables deben obedecer al modelo de referencia que plantea el estándar.

El concepto de laboratorios remotos modulares [50] y el paradigma "Laboratorio como servicio" (LaaS – *Laboratory as a Service*) propone actualmente un modelo técnicamente muy viable que facilita el mantenimiento y el reúso de los componentes del laboratorio, pero su alto nivel de abstracción y virtualización merece especial atención ante la implementación de didácticas de aula, puesto que el enfoque altamente técnico de la propuesta contrasta con la intención de aportar en la construcción del estándar IEEE P1876 (*Standard for Networked Smart Learning Objects for Online Laboratories*). No obstante, en este enfoque se resalta la importancia de los gestores de aprendizaje (LMS – *Learning Management Systems*) como herramienta que permite la disponibilidad de laboratorios remotos como servicio en los cursos *online*.

2.2 Integración de las TIC en la Enseñanza

Las tecnologías en educación han existido desde los inicios de la misma, por las diversas aulas se han visto pasar televisores, radios, materiales didácticos, videos, proyectores, etc. Pero actualmente existe la facilidad de usar diversidad de medios tecnológicos; los cuales contribuyen en la a veces difícil tarea de capturar la atención de los estudiantes, reducir el tiempo de comprensión, liberar al profesor en tareas repetitivas y sobre todo poner a disposición de quien necesite los contenidos a través del uso de herramientas sociales, con lo que el conocimiento muchas veces se encuentra simplemente a unos cuantos *clicks* de distancia [51].

Las Tecnologías de la Información y la Comunicación (TIC) han ido alcanzado un auge acelerado en diversos sectores de la población , incorporándose rápidamente en la vida social, convirtiéndose su dominio en un importante elemento de la cultura. Las TIC están presente en todas o en casi todas las esferas del desempeño humano, en las diversas áreas del conocimiento, en particular en la educación superior, tanto en la docencia como en la investigación y la gestión escolar y administrativa [52]; por lo tanto, se denominan Tecnologías de la Información y las Comunicación al conjunto de tecnologías que permiten la adquisición, producción, almacenamiento, tratamiento, comunicación, registro y presentación de informaciones, en forma de voz, imágenes y datos contenidos en señales de naturaleza acústica, óptica o electromagnética. Las TICs incluyen la electrónica como tecnología base que soporta el desarrollo de las telecomunicaciones, la informática y el audiovisual [53].

La insuficiencia de recursos educativos en muchas instituciones educativas latinoamericanas es bien conocida. En particular, la escasez de materiales es una de las más serias limitaciones para la formación de niños y jóvenes de los sectores menos favorecidos económicamente. Esa carencia podría resolverse con una dotación mínima de computadores con acceso a Internet de banda ancha en las bibliotecas escolares. La gran cantidad de libros, revistas, periódicos, diccionarios, enciclopedias, mapas, documentos, videos, muchísimos de ellos gratuitos y con capacidad de multimedia, justifican una inversión inicial en dotación e instalación de equipos y un gasto de sostenimiento cuyo valor sería marginal si se lo compara con el gasto educativo de cualquier país latinoamericano. El acceso a Internet permitiría, además, una cantidad de experiencias educativas nuevas como visitas a museos de arte y de ciencias, acceso a laboratorios virtuales o remotos, viajes virtuales a ciudades o regiones remotas, utilización de software educativo interactivo, etc.

Ese esfuerzo de dotación general a las bibliotecas escolares traería importantes cambios a las instituciones educativas, abriría las puertas de un nuevo mundo para sus estudiantes y ayudaría a mejorar la calidad de la educación de América latina [54]. Existen dos razones muy importantes por las que los gobiernos deben ir mucho más allá de dotar las bibliotecas escolares con acceso a la Web (ver Figura 2.12).

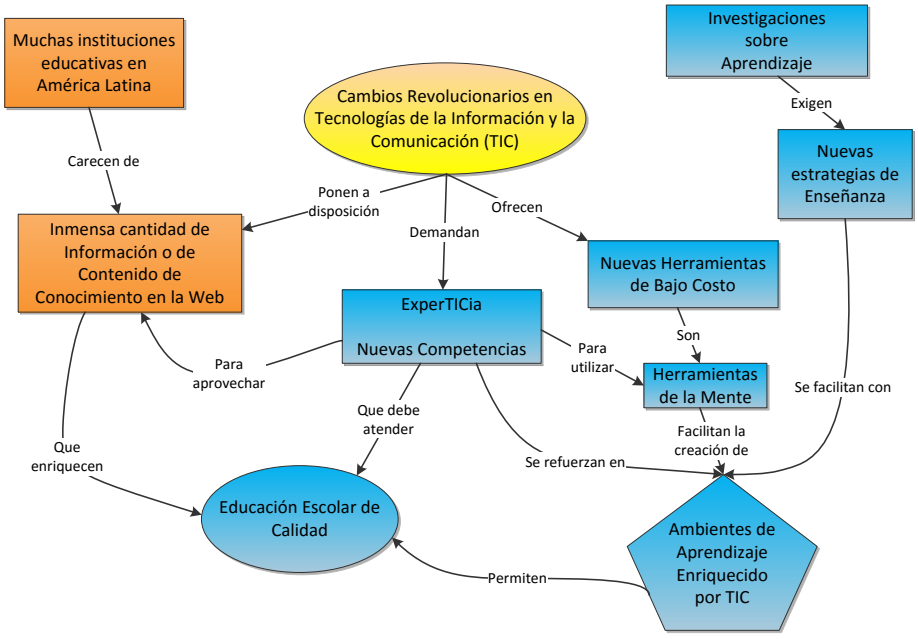


Figura 2.12 Justificación de las TICs en la educación

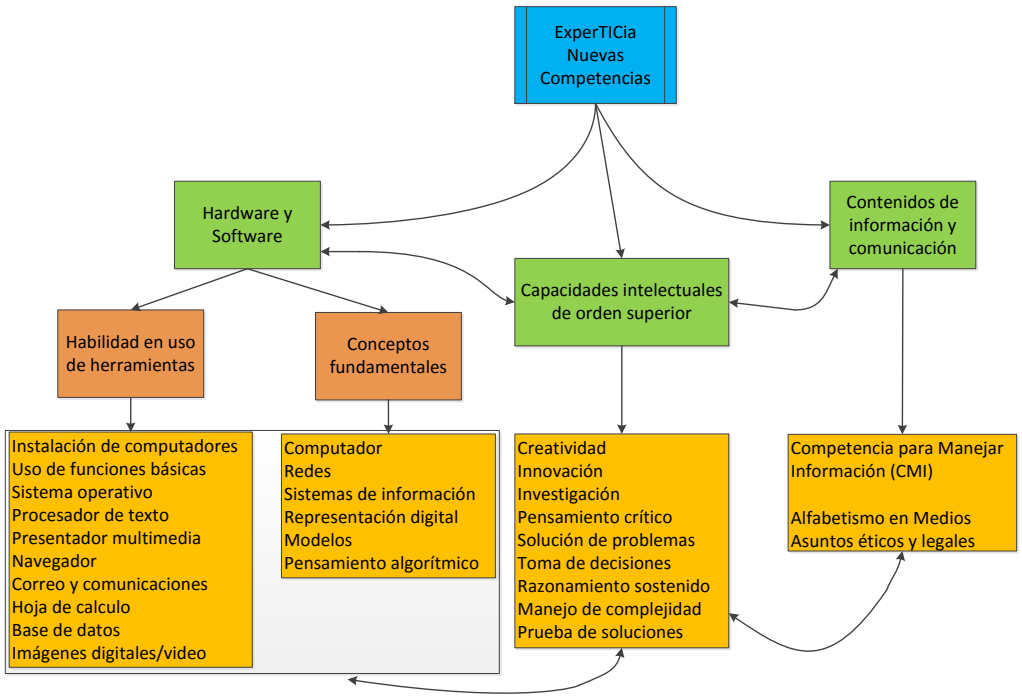


Figura 2.13 ExperTICia en las nuevas tecnologías

En [54] se propone llamar experTICia (Figura 2.13) a la condición de una persona competente en las nuevas demandas de formación originadas en la revolución de las TIC, demandas que, como ya se dijo, deben ser atendidas por cualquier sistema escolar de calidad contemporáneo. La experTICia incluye unas competencias relacionadas con el hardware y el software; otras relacionadas con los contenidos de la información y las comunicaciones; y un tercer tipo que enlaza las dos anteriores con capacidades intelectuales de orden superior. Las primeras implican un conocimiento de los conceptos fundamentales de las TIC y la habilidad en el uso de sus diversas herramientas. El conocimiento de los conceptos fundamentales de las TIC y las habilidades en el uso del hardware y del software componen la primera parte de la experTICia. La segunda, está relacionada con el uso y la producción de los contenidos de la información, tanto en la Web como en los medios digitales en general.

2.3 Servicios Web 2.0

Con la Web 1.0 las funciones de un usuario de internet se limitaban a consultar o buscar información específica en los sitios web, cuyos creadores eran expertos en el tema de diseño web estática. La evolución de la Web permite que ahora los mismos usuarios construyan o participan en la construcción de los contenidos. De acuerdo con [55], la Web no tiene una frontera clara sino un núcleo gravitacional; es decir, puede ser vista como un sistema de principios y prácticas que conforman todo un sistema solar de sitios, que muestran todos estos principios a una distancia variable de ese núcleo.

La filosofía Web 2.0 considera espacios virtuales auto-organizados, abiertos, adaptativos, ágiles, accesibles y fáciles de usar, que cuentan con servicios de soporte colaborativo y posibilitan que los usuarios compartan, opinen y creen nuevos contenidos [56]. En [57] se define la Web 2.0 como un espacio o un escenario en el que convergen usuarios, servicios, medios y herramientas. Así, se establece un terreno en el que estas relaciones tejen redes sociales, destacándose la participación, la posibilidad de interactuar y de conversar.

Considerando la filosofía de participación, retroalimentación y colaboración de la Web 2.0, en la que resulta fundamental el compartir recursos y producir contenidos que otros usuarios puedan reutilizar, se puede afirmar que la Web 2.0 se constituye actualmente en una potente herramienta de apoyo en los procesos de enseñanza –aprendizaje. Integrar a los estudiantes en comunidades virtuales en las que éstos puedan discutir, practicar, aportar y compartir lo aprendido, representa una experiencia muy enriquecedora para la adquisición y conservación del conocimiento [58].

Según [59], la filosofía de la Web 2.0 se resume en una actitud y no en una tecnología, pues el usuario es contribuyente y editor de los contenidos. En este sentido, la Web 2.0 se consolida como una tendencia cuyo funcionamiento es cada vez más participativo y bidireccional [60]. Con ello, los alumnos no se limitan simplemente a visualizar o leer contenidos, sino que se involucran con la producción de sus propios contenidos y los publican fácilmente. Cada vez más profesores, alumnos e investigadores se relacionan y comparten conocimientos y experiencias usando el concepto Web 2.0.

En los contextos colaborativos, se incrementa la posibilidad de que los estudiantes puedan resolver problemas reales que se les presenten [61], lo que les permite desarrollar competencias no solo cognitivas y enriquecer el proceso de aprendizaje. Así, cuando docentes y estudiantes trabajan bajo entornos Web 2.0, serán entes activos del internet, capaces de crear, publicar y compartir sus conocimientos [58].

Con base en [62], la Figura 2.14 resume las principales ideas marco de la Web 2.0 y sus principales servicios para garantizar la interacción entre usuarios.

2.3.1 Servicios Web 2.0 en Educación

Desde la perspectiva de las TIC, el uso y desarrollo actual de herramientas y servicios basados en "software social", Web 2.0 plantea una nueva visión y oportunidades para la innovación de la educación y la gestión del conocimiento a nivel organizacional y personal. El software social: *blogs*, *wikis*, marcadores sociales (*social book-marking*), servicios de sindicación de contenidos basados en RSS, *podcasts*, repositorios sociales de videos, fotos, presentaciones, archivos, entre otros. Esto representa un soporte robusto para los procesos y prácticas educativas abiertas y, en general, para las universidades [63]. A continuación, se presentan algunas generalidades de las herramientas de la Web 2.0 que aplican a los procesos educativos.

2.3.1.1 Redes Sociales

La capacidad de comunicación y de poner en contacto a las personas que tienen las redes ha provocado que un gran número de personas las esté utilizando con fines muy distintos. Se utilizan para encontrar personas y entablar diálogos, para debatir sobre los temas más variados, apoyar causas de todo tipo, organizar encuentros de amigos, ex-compañeros de estudios o para dar a conocer congresos y conferencias [64].

Involucrándose en este fenómeno social, en el ámbito educativo los usuarios comparten conocimientos sobre una determinada materia o disciplina en las redes sociales, pueden mostrar sus trabajos y poner a disposición su experiencia [65].

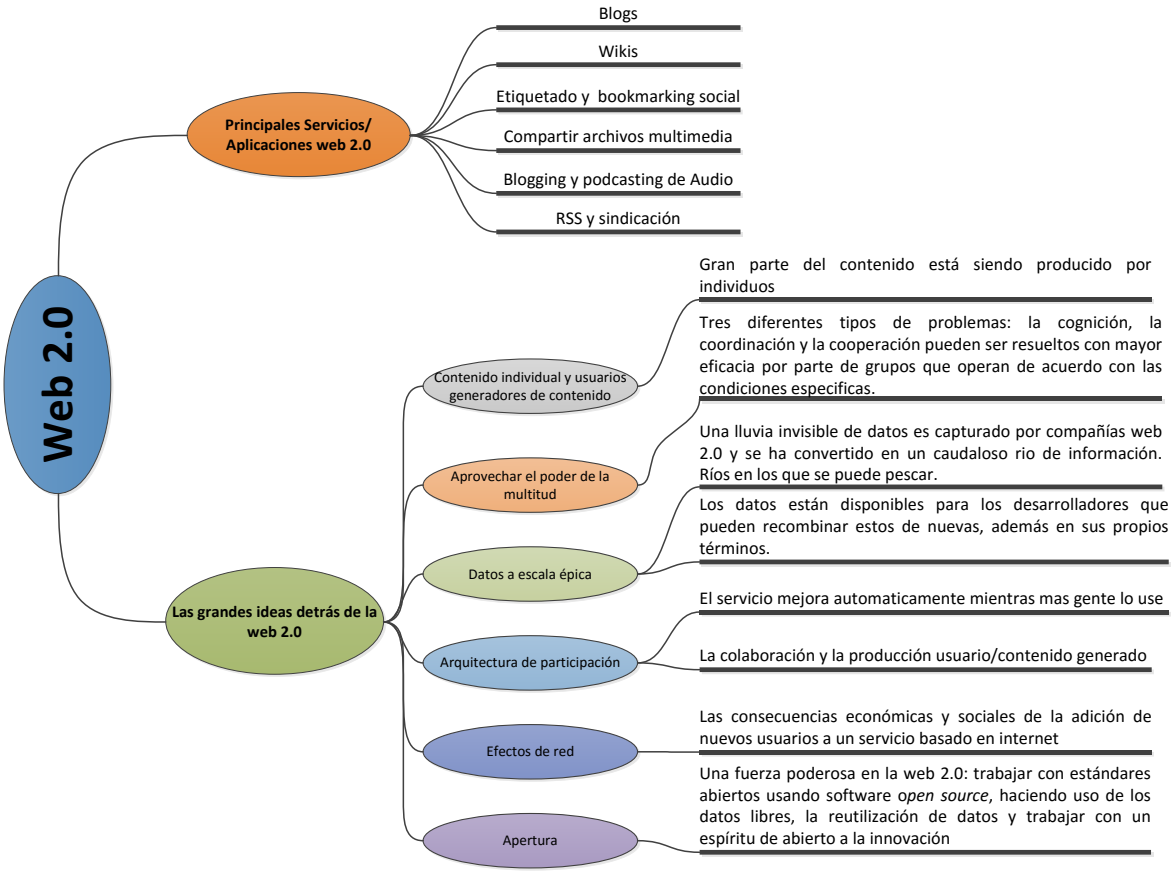


Figura 2.14 Ideas de la Web 2.0 y sus principales servicios

La tecnología sobre la que se sustentan las redes sociales permite a sus usuarios compartir todo tipo de datos e información en múltiples formatos, como audio, texto y video. De esta manera, los estudiantes pueden crear nuevos contenidos, mezclar los contenidos existentes, agregar comentarios, etiquetar diferentes contenidos, compartirlos con sus compañeros de aula o bien con estudiantes de otras partes del mundo, entrando a un ciclo permanente de retroalimentación [58].

Entre las principales redes sociales que han surgido en internet se pueden destacar: *Facebook*, *Twitter*, *FriendFeed*, *Linkedln*, *Hi5*, *LiveJournal*, *MySpace* y *Google+*.

Los principales beneficios que ofrece *facebook* para el ámbito educativo son: su alto grado de integración, tanto para alumnos como para académicos; incrementa la capacidad del estudiante para identificar información sobresaliente en los debates académicos; incrementa el aspecto social entre pares y proporciona a los departamentos escolares una herramienta de comunicación de eventos y noticias de la propia institución educativa [66].

2.3.1.2 Wikis

Los Wikis son herramientas de colaboración, simples, flexibles y potentes. Se pueden utilizar para múltiples cosas, desde repositorios o listas de enlaces Web hasta la creación de enciclopedias. En estructura y lógica es similar a un *blog*, pero en este caso, cualquier persona puede editar los contenidos, aunque hayan sido

creados por otra persona. Además, los *wikis* permiten consultar las diferentes versiones o modificaciones que ha sufrido el texto hasta alcanzar su versión definitiva.

La *Wikipedia* es el *Wiki* más grande en internet. Esta enciclopedia ha sido construida o editada por voluntarios. Cuando un usuario inicia un artículo de un tema de interés, el resto de la comunidad puede añadir contenidos, editar el trabajo de otro o añadir otra página de subcontenidos. Con un *Wiki*, se facilita el desarrollo de contenidos de una asignatura con el aporte de todos; así, el libro de texto dejaría de ser la herramienta base (a veces, única) [67].

Si bien es una muy buena herramienta donde se pueden elaborar documentos en conjunto, presenta una desventaja en el riesgo de no controlar la información presentada para un contenido determinado. Sin embargo, si los errores se detectan, estos pueden ser corregidos también en conjunto. Entre otras herramientas *wikis* se destacan *WikiTaller*, *AulaWiki21* y *WikiHow*.

2.3.1.3 Blogs

Un *blog* es una herramienta Web que permite publicar artículos (denominados "entradas" o "*posts*"). Un *blog* funciona como un diario en línea y puede incluir texto, imágenes y sistemas de búsqueda y navegación a otros blogs [68], además de permitir que los usuarios comenten los artículos. Los *Blogs* ofrecen muchas posibilidades de uso en procesos educativos. Por ejemplo, para estimular a los alumnos en: escribir, intercambiar ideas, trabajar en equipo, diseñar, visualizar de manera instantánea de lo que producen, etc. La creación de *Blogs* por parte de estudiantes ofrece a los docentes la posibilidad de exigirles realizar procesos de síntesis y composición, ya que al escribir en Internet deben ser puntuales y precisos, en los temas que tratan [69].

Los docentes pueden utilizar los *Blogs* para acercarse a los estudiantes de nuevas maneras, sin que necesariamente se vea limitada su interacción. Esto se puede hacer publicando materiales de manera inmediata y permitiendo el acceso a información o recursos necesario para realizar proyectos o tareas. También ofrecen la posibilidad de enriquecer los contenidos con herramientas de multimedia como audio, video, imágenes entre otros.

La facilidad con que se crean y alimentan los *Blogs* los hace muy llamativos porque gracias a los asistentes y las plantillas prediseñadas, solo hay que atender a los contenidos y materiales a publicar. Esto permite que cualquier docente o estudiante pueda crear recursos y contenidos de temas educativos sin necesidad de instalar aplicaciones o de tener conocimientos de programación.

2.3.1.4 Etiquetado y *Social Bookmarking*

Una etiqueta o *tag* es una palabra clave que se le adiciona a un objeto digital; por ejemplo, a un sitio Web, una fotografía o un clip de video, para describirlo, pero no como parte de un sistema formal de clasificación sino de nuevas maneras que posibilitan a cualquier persona encontrar información. Por su parte, "*Social Bookmarking*" es una forma en la que los usuarios de Internet almacenan, organizan (etiquetan), comparten y buscan páginas Web de interés para ellos. En un sistema de este tipo, las personas guardan enlaces a páginas Web que desean recordar y/o compartir que generalmente son públicos pero, dependiendo de las características del servicio, pueden guardarse en forma privada, compartirse únicamente con personas o grupos específicos, compartirlos solo dentro de ciertas redes, o en combinación de público y privado. La mayoría de los servicios en línea de este tipo permiten ver los enlaces guardados cronológicamente, por categoría o etiqueta, mediante un buscador o, incluso, al azar [70]. Algunos sitios que facilitan el "*Social Bookmarking*": *Blinklist*, *Del.icio.us*, *Digg*, *LibraryThing* y *RawSugar*.

2.4 Sistema Informático Propuesto

2.4.1 Definición

Un sistema informático puede definirse como el conjunto de recursos que interactúan entre sí para el almacenamiento, transmisión y procesamiento de información, con base en conocimientos y técnicas relacionadas con el propósito del mismo.

Según [71], la palabra "informática" tiene origen francés (*Informatique* [72]). Fonológicamente, la informática combina elementos tanto de ***INFOR****mación* y *auto****MATICA***. Este término fue acuñado por *Dreyfus*, en Marzo de 1962, refiriéndose a la aplicación de los computadores al almacenamiento y procesamiento de la información [73]. El diccionario de la Real Academia Española [74] registra el significado de esta palabra como: "*Conjunto de conocimientos científicos y técnicas que hacen posible el tratamiento automático de la información por medio de computadoras*".

Así mismo, este diccionario define "sistema" como: "*Conjunto de cosas que relacionadas entre sí ordenadamente contribuyen a determinado objeto*". En [75] se define como una "*Composición integrada por personas, productos, y procesos que proveen la capacidad de satisfacer una necesidad indicada o un objetivo*", lo que involucra varios agentes o recursos en procesos de interacción permanente para el cumplimiento de un propósito.

Como se trata de procesamiento de información, existen recursos físicos y lógicos que se deben relacionar para el cumplimiento de tareas específicas. Los recursos lógicos como el software se regulan mediante instrucciones con base en el razonamiento humano que define la base de conocimiento del sistema. Estos recursos requieren de los medios físicos para la realización de procesos de captura, tratamiento y presentación de la información [71]. Así, un sistema informático cuenta con recurso humano, de software y de hardware para el cumplimiento del propósito para el cual se diseña.

2.4.2 Arquitectura del Sistema Informático Propuesto

El sistema informático propuesto en este trabajo presenta tres servidores, el primero es el servidor Web, que se encarga de controlar la interfaz de usuario y una base de datos MySQL; el segundo servidor es de control o de *hardware* y se encarga de interactuar con la tarjeta de experimentación por medio del protocolo de comunicación USB; por último, una cámara IP actúa como un servidor de video. En la Figura 2.15 se observa la conexión de los servidores utilizados, la conexión entre éstos se realiza mediante la red interna de la Universidad del Valle y como protocolo de comunicación está el TCP/IP.

2.4.2.1 PC Servidor Web

El servidor Web usado en el sistema informático es un computador con sistema operativo *Windows 7* con arquitectura de 32 bits, que tiene instalado el paquete software ***wampserver*** que habilita el PC como servidor; además, contiene la base de datos *MySql* para el manejo de usuarios. Este PC tiene un contenedor de *Servlets* y un servidor *FTP* para posibilitar la comunicación con los demás elementos de la red. El servidor está conectado a la red interna de la Universidad del Valle. El servidor Web maneja las peticiones *http* de los clientes y, dependiendo de éstas, las redirecciona al servidor de video o al servidor de *hardware*.

Dependiendo del paquete software utilizado como servidor Web, éste puede ser multiplataforma e independiente de la arquitectura.

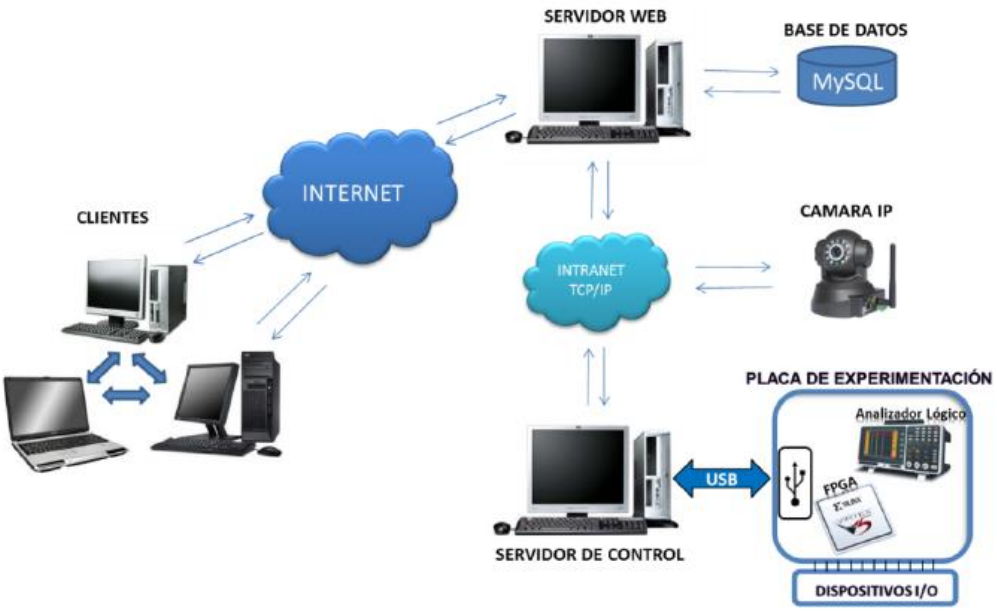


Figura 2.15 Arquitectura del sistema informático de acceso remoto

2.4.2.2 PC Servidor de control o de Hardware

El servidor de *hardware* es un PC corriendo un sistema operativo *Windows* 7 de 32 bits. Es el encargado de recibir peticiones desde el servidor *hardware* y hacer llegar éstas hasta la tarjeta de experimentación a través de *USB*. Por ello, este servidor contiene los *drivers* necesarios para el *hardware*, un contenedor de *Servlets* para el manejo del mismo y envío de las respuestas pertinentes al servidor Web y un servidor *FTP* para el envío de archivos hacia el mismo servidor. Este servidor se encuentra conectado a la red interna de la Universidad del Valle.

La biblioteca utilizada para establecer la comunicación USB con el *hardware* puede ser implementada en arquitecturas de 32 o 64 bits y en diferentes plataformas; por lo tanto, la aplicación desarrollada puede ser independiente tanto de la plataforma como de la arquitectura.

2.4.2.3 Cámara IP (servidor de video)

La cámara *IP* es un elemento muy importante para un laboratorio de acceso remoto, dados los componentes visuales que el *hardware* posee, esta cámara responde a las peticiones que el servidor Web le haga. También está conectada a la red interna de la Universidad del Valle.

3. DESARROLLO DE LA COMPONENTE SOFTWARE DEL SISTEMA INFORMÁTICO

La componente software del sistema informático propuesto constituye la base del procesamiento lógico de la información desde los sistemas computacionales e infraestructura de red que hacen parte de su arquitectura (Ver sección 2.4.2). El tratamiento de información se realiza fundamentalmente sobre las capas de aplicación, red y enlace de datos, considerando respectivamente los procesos de interfaz de usuario, comunicación de estaciones remotas, servidores y placa de experimentación; esta última para efectos de configuración y operación. La base de conocimiento para la integración del diseño de interfaz de usuario, las políticas de administración del recurso y la disponibilidad de procesos y servicios para el usuario la constituye esencialmente el soporte metodológico para la implementación de didácticas de aprendizaje activo y espacios de trabajo centrados en el estudiante.

Este capítulo describe el proceso de desarrollo de la componente software del sistema informático, en el que se destaca, como una de las contribuciones este trabajo, la integración de un conjunto de servicios sobre la misma plataforma para el soporte del aprendizaje activo en la enseñanza de sistemas digitales.

3.1 Gestores de Contenido

La tendencia mundial en la elaboración o diseño de páginas Web es la utilización de los sistemas gestores de contenido (*CMS*) los cuales, son programas que permiten crear una estructura de soporte *framework* para la creación y administración de contenidos, principalmente en páginas Web; por lo tanto, un CMS consiste en una interfaz donde se controla una o varias bases de datos en las que se aloja el contenido del sitio. Permitiendo manejar de manera independiente el contenido y el diseño del portal web, una característica importante del CMS es que posibilita que el sitio pueda ser rediseñado y alterar la información presentada sin necesidad de cambiar el formato del contenido nuevo, además de que facilita la edición y publicación desde múltiples usuarios, según las políticas de administración [76]. En [77] se plantean 4 ítems básicos que los gestores de contenidos permiten (ver Figura 3.1).

Figura 3.1 Elementos básicos de un gestor de contenido [77]

Desde la perspectiva de un usuario administrador, un CMS es un sistema para gestionar de forma uniforme, accesible y cómoda, un sitio web dinámico con actualizaciones periódicas y sobre el que pueden trabajar

una o más personas, cada una de las cuales tiene una función determinada; por otro lado, para un usuario del sitio Web, el CMS es un sitio web dinámico con apariencia e interfaz uniforme con un diseño centrado en el usuario y que permite llevar acabo fácilmente las tareas para las que ha sido diseñado [78].

Existen gran variedad de gestores de contenidos, en [79], se mencionan 20 de los más prometedores, entre los que se encuentran *Wordpress* y *Joomla*.

En muchas páginas Web dedicadas a realizar estadísticas sobre CMS mencionan a *wordpress* como el gestor de contenido de mayor uso por el público en general; sin embargo, esta situación se presenta debido a que este *CMS* es principalmente para el desarrollo fácil de *blogs*; por otro lado, *Joomla* está enfocado principalmente al desarrollo de páginas web, y en los años recientes su popularidad y uso ha venido en aumento convirtiéndose en una herramienta potente para desarrollos muy profesionales pero de forma fácil.

Joomla es un sistema gestor de contenidos reconocido entre la comunidad de diseñadores web, que posibilita construir sitios web y poderosas aplicaciones en línea. Muchos aspectos incluyendo su facilidad de uso y extensibilidad han hecho a *Joomla* el software más popular para sitios web. Una de las características más importantes es que es una solución de código libre disponible gratuitamente para todo el mundo y tanto en inglés como en español [80]. Entre las aplicaciones más poderosas en las que *Joomla* es usado se encuentran: sitios Web o portales corporativos, Intranets y extranet corporativas, revistas en línea, periódicos y publicaciones, comercio electrónico y reservas en línea, aplicaciones gubernamentales, pequeños sitios Web de negocios, sitios Web académicos y páginas familiares o personales.

Las principales ventajas que ofrece este sistema gestor de contenido se listan a continuación [81]:

- **Posibilidad de modificar el código fuente:** los programadores o diseñadores pueden acceder a lo más profundo del código y modificarlo según las necesidades de la organización.
- **Más de tres mil extensiones:** la gran mayoría de libre uso, que permiten ampliar las posibilidades y características de *Joomla*. Si el núcleo de *Joomla* no suple alguna necesidad específica, con alguna extensión seguramente se logrará.
- **Instalación en servidores Linux, Mac y Windows:** no hay limitantes, si se tiene un servidor en *Windows*, podrá instalarse sobre Apache.
- **Velocidad de carga:** a diferencia de otras plataformas, *Joomla* permite una carga muy rápida de sus páginas gracias al sistema de caché.
- **Cumplimiento de estándares web:** la más reciente versión de *Joomla* se acerca al ideal de cumplimiento de los estándares del W3C. Gracias a su sistema de plantillas es posible separar la presentación del contenido y marcar semánticamente los documentos.

3.2 Gestores de Aprendizaje

Considerando las ventajas que ofrece la introducción de herramientas Web 2.0 a la educación (ver Figura 3.2) [82], resulta importante integrar elementos como *quices*, tareas, foros, chats, encuestas y calificaciones entre otras, que aportan interactividad a la aplicación permitiendo la constante realimentación docente-estudiante e intercambio de información estudiante-estudiante; por lo tanto, es una necesidad evidente la vinculación de elementos o servicios Web 2.0 al gestor de contenido, lo cual puede explorarse significativamente con la integración de un LMS (*Learning Management System*) al CMS (*Content Management System*).

Un LMS es una aplicación de servidor web que se emplea para administrar, distribuir y controlar las actividades de formación virtual (*e-Learning*) de una organización acorde a un contenido temático establecido. Las principales funciones del LMS son: gestionar usuarios, recursos y contenidos así como

materiales y actividades para la formación o enseñanza de un material determinado; administrar el acceso, controlar y hacer seguimiento del proceso de aprendizaje, realizar evaluaciones, generar informes de avances, y gestionar servicios de comunicación como foros de discusión y videoconferencias. En cuanto a su puesta en funcionamiento, un LMS requiere soporte de una base de datos que permita el registro y monitoreo de actividades [83].

Figura 3.2 Principales características de la Web 2.0 [82]

Moodle es una alternativa de gestor de aprendizaje a las soluciones comerciales como *Blackboard* y *WebCT*, y se distribuye gratuitamente bajo licencia *Open Source*. El entorno de aprendizaje de esta herramienta está basado en los principios pedagógicos constructivistas, con un diseño modular que hace fácil agregar contenidos que motivan al estudiante [84].

La perspectiva constructivista de *Moodle* implica activamente a un estudiante en su aprendizaje para que le dé significado logrando que este tipo de enseñanza motive que el alumno pueda analizar, investigar, colaborar, compartir, construir y generar, basándose en lo que ya sabe.

3.3 Requisitos del Sistema

Los laboratorios virtuales y remotos permiten el acercamiento a estudiantes, que por diferentes motivos no pueden acceder a prácticas y simulaciones, hacia un espacio real de experimentación. Los planteamientos reportados en la bibliografía consideran aspectos que comúnmente forman parte de las aplicaciones cliente/servidor en cuanto a interfaces de usuario con contenidos técnicos y los controles y herramientas estrictamente necesarios. Sin embargo, para que la experiencia de aprendizaje, en particular para sistemas digitales, se acerque suficientemente a los beneficios que se obtienen en una práctica real presencial, deben considerarse además de los nuevos conceptos introducidos en Web 2.0 (Ver Capítulo **Error! Reference ource not found.**), servicios de programación y test de las funciones electrónicas bajo prueba, garantizando también el acceso al recurso *hardware*. Con esto, resulta necesario diseñar una aplicación que brinde todos estos servicios a través de una única plataforma, de manera que se optimice desde el diseño electrónico y de software hasta las metodologías de enseñanza/aprendizaje que tendrían lugar a través de ella como recurso. Así, con base en la población que se beneficiaría de este tipo de sistemas se plantean los siguientes cuestionamientos que sirven como base para la definición de los requisitos que debe cumplir la aplicación:

- ¿Qué aspectos técnicos y metodológicos deben considerarse en la presentación de una aplicación Web para experimentación remota de sistemas digitales, en procura de garantizar el desarrollo de los procesos de enseñanza/aprendizaje en el área?
- ¿Cómo administrar de forma eficiente y automática los recursos de la aplicación con el fin de garantizar el acceso a múltiples usuarios que intentan hacer uso de la plataforma hardware?
- ¿Qué consideraciones deben tenerse en cuenta a nivel de enlace de datos para garantizar el acceso al recurso físico de experimentación, con las condiciones y parámetros adecuados y preestablecidas por el usuario a nivel de aplicación (programación FPGA y Test)?

Con la especificación de requerimientos se brinda una garantía de la funcionalidad de la aplicación y su usabilidad en general. En ésta se realiza una descripción general del proyecto, así como los requisitos que este debe cumplir.

La aplicación Web desarrollada con el nombre ***Laboratorio remoto para el aprendizaje en línea de sistemas digitales***, permite a un usuario el acceso a una tarjeta de entrenamiento (*Plataforma hardware*) que se encuentra en una ubicación remota. Así mismo, el usuario tiene acceso a la información, trabajos asignados y herramientas que le proporcione un docente administrador de la plataforma.

3.3.1 Descripción general

La aplicación pretende proporcionar a los usuarios el soporte para el manejo de una placa para experimentación que involucra *hardware* reconfigurable y *hardware* para análisis de señales, posibilitando que éstos prueben sus diseños en esta tarjeta electrónica. Los usuarios también deben poder acceder a información relevante al tema expuesto en el sitio web e interactuar con otros usuarios a través de las herramientas que el portal proporcione.

La aplicación debe suministrar información o acceso a contenidos autorizados dependiendo del tipo de usuario que acceda al sitio web.

3.3.2 Funciones de la Aplicación

La aplicación debe poseer niveles acceso dependiendo el usuario que ingrese, por lo tanto deben existir los siguientes tipos de usuarios.

- Usuarios no registrados
- Usuarios Estudiantes
- Usuarios Docentes Administradores

Los principales elementos de *hardware* que involucra la tarjeta para experimentación se listan a continuación:

- FPGA
- Analizador Lógico
- Administrador de tareas (Microcontrolador)
- *Leds*, *Displays* y *LCD*

3.3.2.1 Usuarios no registrados

Para este tipo de usuarios, la aplicación debe considerar:

- Información general
- Localización y contacto: Se especificará la ubicación del grupo de investigación y de la Institución así como la información de contacto de los mismos.

- Enlaces: Se pondrán a disposición del usuario enlaces para redirigirse a la página oficial del grupo de investigación y de la Universidad del Valle.

3.3.2.2 Usuarios estudiantes

Para este tipo de usuarios, la aplicación debe considerar:

- Citas: Se le presentará a los estudiantes la opción de solicitar una cita para hacer uso de los recursos del laboratorio, es decir del manejo del hardware.
- Perfil: Cada estudiante podrá administrar su propio perfil de usuario modificando los campos que se le presenten.
- Configurar FPGA: Al hacer uso del Hardware se le proporcionará al usuario la posibilidad de programar la FPGA con cualquier diseño que previamente haya creado en una herramienta externa a esta aplicación.
- Configurar analizador Lógico: El hardware posibilita capturar salidas de un diseño determinado bajo prueba y para esto se debe dar una configuración específica del analizador lógico; por lo tanto, se presentarán al usuario los campos necesarios para introducir dicha configuración.
- Ver y analizar resultados: Finalizado el proceso se le presentarán al usuario los resultados obtenidos con lo que éste deberá hacer el respectivo análisis de los mismos.
- Información del curso: Se pondrá a disposición de los estudiantes todo el material requerido que contribuya en el aprendizaje de los sistemas digitales desde manuales, tutoriales, ejemplos, material de clase y las guías de laboratorio.
- Cámara Web: Dado que el hardware posee elementos visuales como *leds*, *displays* 7 segmentos y LCD se incorporará una cámara web que posibilite que el usuario vea en tiempo real lo que acontece en el laboratorio.
- Subir archivos: Los estudiantes tendrán la opción de subir archivos que pueden corresponder a tareas propuestas en el curso

3.3.2.3 Usuarios docentes administradores

Para este tipo de usuarios, la aplicación debe considerar:

- Registrar Alumnos: Los docentes tendrán la capacidad de inscribir los alumnos que harán uso del sitio web.
- Perfil: Cada docente podrá administrar su propio perfil de usuario modificando los campos que se le presenten.
- Eliminar Alumnos: Dado que los cursos tienen una duración de tiempo limitada, los alumnos cambiarán cada cierto tiempo, con lo que se verá la necesidad de eliminar los usuarios que no usarán más la aplicación.
- Listado de Alumnos: Los profesores tendrán a disposición el listado de los usuarios dependiendo del curso al que pertenezcan.
- Listado de Profesores: Se listarán los docentes que estén registrados en la aplicación para dictar el curso.
- Poner notas: El sitio permitirá que los profesores asignen notas a los trabajos que los estudiantes suban a la aplicación.
- Anuncios: La aplicación permitirá que los docentes envíen correos o anuncios importantes a los estudiantes.
- Crear Foros: Se posibilitará que los docentes inicien foros para discutir algún tema o particular o simplemente para retroalimentación.

- Crear Chats: Igualmente que con los foros se dispondrá de un chat para interacción entre los usuarios del sitio.
- Crear tareas, talleres y trabajos: Los docentes podrán proponer trabajos que el estudiante debe desarrollar.

3.3.3 Características de los usuarios

Los usuarios que no hayan sido registrados no tendrán acceso, sólo podrán ingresar a la página de inicio donde se dará una información general de las funciones del sitio web y alguna información de contacto, adicionalmente estará el formulario de acceso con el que ingresan los usuarios registrados.

Los usuarios registrados pueden dividirse en Docentes y Estudiantes, estos últimos son los que más interactúan con el sitio ya que tendrán a disposición muchos recursos, entre ellos el laboratorio que es el centro de la aplicación, también tendrán acceso a material de apoyo para el proceso educativo y la posibilidad de interactuar con otros compañeros gracias a las herramientas disponibles como foros y chats. Finalmente los docentes son los encargados de gestionar todos los recursos del sitio.

Los usuarios deberán tener conocimientos mínimos de informática y de navegación en internet para tener un exitoso acceso y manejo del sito, adicionalmente tener algún conocimiento de programación en *VHDL* y del manejo de la herramienta *ISE DESIGN* de *XILINX* que si bien son programas externos de la aplicación web, se requiere el uso de éstos para generar los archivos que se cargarán en la misma para la configuración de la FPGA.

3.3.4 Restricciones Generales

Un usuario de la aplicación requiere de un computador bien sea portátil o de escritorio con acceso a Internet y con el software que se lista a continuación

- ISE DESIGN de Xilinx
- JDK de JAVA versión 7 de Oracle
- Navegador de Internet (preferiblemente Mozilla)

Dados los requerimientos planteados en 3.3.2 y teniendo definido el uso de un gestor de contenido para la construcción del sitio web, se requieren herramientas adicionales para dar completa funcionalidad a la aplicación; por lo tanto a continuación se definen algunas de las herramientas a usar.

3.3.5 Herramientas para el desarrollo de aplicaciones cliente/servidor

En el desarrollo de las aplicaciones propuestas en este proyecto, se buscan herramientas que permitan dar al usuario gran interactividad; por lo tanto, es necesario hacer uso de los modelos existentes. El primer modelo es la programación del lado cliente, el cual tiene la propiedad de ejecutar las aplicaciones requeridas por el usuario en el navegador que dispone éste para acceder a la información; el segundo modelo es la programación del lado servidor y tiene la propiedad de ejecutar las aplicaciones requeridas directamente en el servidor para posteriormente visualizar los resultados obtenidos de la ejecución en el lado cliente.

3.3.5.1 Programación del lado cliente

En el lado cliente se cumplen los siguientes eventos:

- El navegador del usuario envía una petición al servidor de aplicaciones.
- El servidor envía una respuesta que contiene código que el navegador es capaz de interpretar.
- El navegador interpreta el código enviado por el servidor y realiza una determinada acción.

La programación del lado del cliente tiene como principal ventaja permitir que la ejecución de una aplicación se delegue al cliente, con lo cual se evita recargar al servidor de trabajo. El servidor sólo envía el código, y es tarea del navegador interpretarlo. La gran desventaja de esta metodología es que el código que el servidor envía se vería afectado por lo que pueda realizar (o no realizar) el navegador. Debido a las incompatibilidades existentes y a la posibilidad que el usuario controle a su voluntad las acciones a ejecutar, la programación del lado del cliente no es muy recomendable y debe limitarse a un código altamente estándar que pueda interpretarse de cualquier forma en cualquier navegador, lo cual obliga a ejecutar la gran mayoría de las aplicaciones y servicios de un sitio web del lado del servidor.

Para este trabajo en particular se usa una herramienta que se ejecuta del lado del cliente, los *Applet* de *Java*, que se definen como componentes de software que corren en el contexto de otro programa, por ejemplo un navegador Web. El *Applet* debe correr en un contenedor, que lo proporciona un programa anfitrión, mediante un Plug-In.

A diferencia de un programa, un *Applet* no puede correr de manera independiente, ofrece información gráfica y a veces interactúa con el usuario, típicamente carece de sesión y tiene privilegios de seguridad restringidos. Un *Applet* normalmente lleva a cabo una función muy específica que carece de uso independiente y está formado por una clase o un conjunto de clases escritas en lenguaje Java y que está insertado en una página Web. Cuando un usuario carga la página en la que se encuentra el *Applet*, este se ejecuta localmente, es decir en la máquina del cliente donde se está ejecutando el navegador web y no de forma remota [85] (ver Figura 3.3).

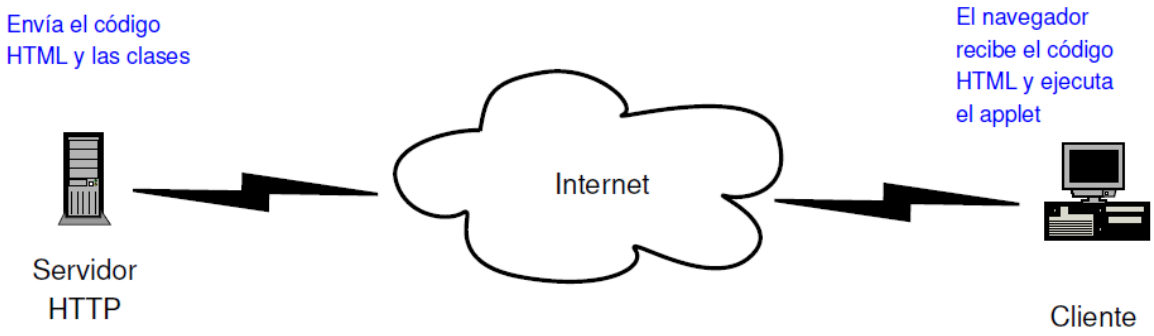


Figura 3.3 Ejecución de un *Applet* Java [85]

La ejecución del código en el lado cliente implica que la seguridad sea un factor determinante. Los *Applet* están sometidos a unas restricciones de seguridad por defecto. Los gestores de seguridad de un navegador verifican, entre otras cosas, que los *Applet*:

- No puedan leer los archivos locales
- No puedan escribir en los discos locales
- No puedan establecer conexiones a otras máquinas, salvo al servidor que contiene el *Applet*
- No ejecuten programas locales
- No puedan obtener información privada de los usuarios

3.3.5.2 Programación del lado del servidor

En el lado cliente se cumplen los siguientes eventos:

- El navegador del usuario envía una petición al servidor de aplicaciones.
- El servidor ejecuta una aplicación que realiza una determinada acción.
- El servidor envía el resultado de dicha aplicación al cliente.
- El navegador muestra el resultado recibido desde el servidor.

Programar del lado del servidor tiene como ventaja que cualquier cosa puede hacerse sin tener en cuenta el tipo de cliente o el software existente en el equipo cliente, ya que la aplicación se ejecuta en el servidor, que es un ambiente controlado. Una vez ejecutada la aplicación, el resultado que se envía al cliente puede estar en un formato "normalizado" que cualquier cliente debería mostrar. La desventaja reside en que el servidor se sobrecarga de trabajo, ya que además de servir páginas, es responsable de ejecutar aplicaciones que en muchas ocasiones requiere alto consumo de recurso computacional. A menudo, esto redunda en requisitos de hardware mayores a medida que el servidor ejecuta más servicios.

Los lenguajes del lado servidor más utilizados para el desarrollo de páginas y con los cuales se trabaja son PHP y *Java*.

- **PHP** (*Hypertext Preprocessor* – Preprocesador de Hipertexto), es un lenguaje "*open source*" (de código abierto) interpretado de alto nivel, especialmente pensado para desarrollos Web y que puede ser embebido en páginas HTML. La mayoría de su sintaxis es similar a C, *Java* y *Perl*. La meta de este lenguaje es permitir escribir a los creadores de páginas Web, páginas dinámicas de una manera rápida y fácil, aunque se puede hacer mucho más con PHP. Podría ser considerado como el lenguaje análogo al ASP y es utilizado en plataformas Unix, Linux y Windows.
- ***Java*** es un lenguaje de programación desarrollado por *Sun*, en un intento de resolver simultáneamente todos los problemas que se le plantean a los desarrolladores de software por la proliferación de arquitecturas incompatibles, tanto entre las diferentes máquinas como entre los diversos sistemas operativos y sistemas de ventanas que funcionaban sobre una misma máquina, añadiendo la dificultad de crear aplicaciones distribuidas en una red como Internet. Si bien su uso se destaca en el web, sirve para crear todo tipo de aplicaciones (locales, Intranet, Internet, e incluso móviles).

Para desarrollar aplicaciones del lado del servidor existen 2 formas de realizarlo: el protocolo CGI y módulos dados al servidor Web.

3.3.5.2.1 Protocolo CGI

El protocolo CGI (*Common Gateway Interface* – Interfaz de Entrada Común) fue creado para establecer un protocolo estándar de comunicación entre el servidor web y cualquier lenguaje de programación, de forma tal que desde el lenguaje "x" puedan recibirse datos que el usuario envía usando el método "POST" o "GET" desde un navegador, y además el resultado de la aplicación sea derivado por el servidor al navegador. Típicamente, para recibir datos se usa alguna biblioteca o módulo del lenguaje elegido que implementa el protocolo CGI. Para enviar datos, simplemente se envían a la salida estándar desde el lenguaje elegido, y el servidor se encarga de re-direccionar esto al navegador. Para ejecutar una aplicación CGI el servidor en general procede de la siguiente manera:

- El navegador realiza una petición y envía información al servidor por medio de las directivas "GET" o "POST", a través de variables ambiente.
- El servidor re-direcciona su salida estándar al navegador.
- El servidor crea un proceso, que tiene la salida estándar re-direccionada.
- El servidor ejecuta la aplicación deseada en el proceso creado.
- Se ejecuta la aplicación.

- Finaliza el proceso

3.3.5.2.2 Adicionar un módulo al servidor Web

La tecnología más reciente para la ejecución de aplicaciones consiste en anexar "módulos" a un servidor que permiten a este interpretar un determinado lenguaje (Ej. *PHP*, *ASP*, *Perl*, *Phyton*). De esta forma se logra eficiencia ya que el servidor no necesita crear un nuevo proceso por cada aplicación que ejecuta. Las aplicaciones son portables puesto que son desarrolladas en un lenguaje estándar que no depende del servidor, lo que ofrece confiabilidad; si se produce algún tipo de error, debido al lenguaje en que están diseñados, el "módulo" empleado garantiza que dichos errores no comprometan el funcionamiento del servidor.

Entre los lenguajes disponibles para desarrollar aplicaciones en Internet, se han seleccionado el lenguaje *Java* y los *Servlets*, debido a algunos beneficios que presentan con respecto a otros lenguajes.

Un *Servlet* es un programa que se ejecuta en el contenedor Web de un servidor de aplicaciones. Los clientes pueden usarlo utilizando el protocolo *HTTP*. Comparativamente, así como un *Applet* es cargado por un navegador Web, un *Servlet* es cargado y ejecutado por un contenedor Web.

Un *Servlet* acepta peticiones de un cliente, procesa la información relativa a la petición realizada y devuelve los resultados que pueden ser mostrados mediante *Applet* y páginas HTML. Los *Servlet* también pueden realizar tareas como comunicarse con otro *Servlet* o facilitar el acceso a bases de datos. No obstante, entre sus principales se pueden destacar:

- Al estar escritos en *Java*, son independientes de la plataforma.
- Son seguros y portables debido a que se ejecutan bajo la máquina virtual de *Java*, al mecanismo de excepciones y al uso del administrador de seguridad de *Java*
- No requieren soporte de *Java* en el explorador del cliente, ya que operan del lado del servidor y envían los resultados en *HTML*; no obstante, se pueden utilizar otras interfaces de cliente como aplicaciones de java o *Applet*.

Java proporciona el soporte necesario para escribir *Servlets* a través de los paquetes *javax.servlet* y *javax.servlet.http*; por lo tanto, un *Servlet* es un objeto de la clase de la API *Java Servlet* que implementa *GenericServlet* y *HttpServlet*.

El tipo de *Servlet* más básico puede ser un objeto de la una clase derivada de *HttpServlet* que encapsula los métodos *init* para iniciar el *Servlet*, *doPost* y/o *doGet* para responder a las peticiones de los clientes.

El método de un *Servlet* en el que se encuentra la mayor parte de su funcionalidad es *public void service(HttpServletRequest request, HttpServletResponse response).* Cada vez que se realiza una petición se llama a este método, que recibe dos parámetros: uno para obtener información de la petición y un flujo de salida para escribir una respuesta. Análogamente, se tienen los métodos *GET* y *POST,* que realizan la implementación de respuesta de métodos de comunicación del protocolo *http 1.1*. Estos métodos se citan a continuación:

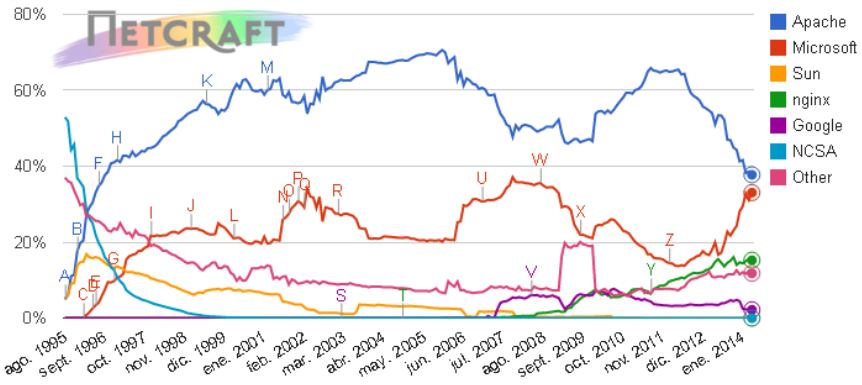
- *public void doGet(HttpServletRequest request, HttpServletResponse response)*
- *public void doPost(HttpServletRequest request, HttpServletResponse response)*

Durante la solicitud/ejecución de un *Servlet* se cumplen los siguientes pasos:

- El cliente Web realiza una petición utilizando un explorador. Si se supone que la página Web del cliente posee un formulario para interactuar con el usuario, los datos recogidos en ese formulario son enviados también como parte de la petición.
- Utilizando como medio Internet, la petición es trasladada al servidor especificado por la URL escrita en el navegador.
- El servidor recoge la petición, la analiza, se da cuenta de la necesidad de ejecutar el *Servlet* y delega esta acción al contenedor de *Servlets*.
- El contenedor ejecuta el *Servlet*.
- Los datos producidos como resultado de la petición, y que tienen que ser enviados al cliente, se trasladan en forma de página Web al servidor Web.
- El servidor envía la respuesta al cliente a través del internet.
- El cliente recoge la página Web enviada y la muestra al usuario.

3.3.5.3 Servidor *Apache Http Server*

Apache es el servidor Web más utilizado actualmente en el mundo (ver Figura 3.4), superando sus competidores, tanto gratuitos como comerciales [86], y distribuyéndose bajo una licencia especial *Apache Software License*. En esta licencia se menciona que los binarios y el código fuente se pueden usar y distribuir de forma libre pero bajo las condiciones mencionadas en la licencia; por ello, se le llama comúnmente software de código abierto. Esto permite que cualquier persona con conocimientos básicos de programación pueda modificarlo y ajustarlo a sus necesidades e igualmente pueda contribuir en su desarrollo con la *Apache Software Foundation*, que es el ente encargado de la implementación y de la referencia de las especificaciones para el servidor *Apache*.



Developer	March 2014	Percent	April 2014	Percent	Change
Apache	354,956,660	38.60%	361,853,003	37.74%	-0.87
Microsoft	286,014,566	31.10%	316,843,695	33.04%	1.94
nginx	143,095,181	15.56%	146,204,067	15.25%	-0.31
Google	20,960,422	2.28%	20,983,310	2.19%	-0.09

Figura 3.4 Estadística de los servidores Web más usados en el mundo hasta Abril de 2014 [86]

El propósito de este tipo de software es transmitir páginas en formato *HTML* a los navegadores cliente que las soliciten, utilizando para ello el protocolo de comunicación *HTTP*. Actualmente, el servidor *Apache*

tiene capacidad para servir páginas tanto de contenido estático, como dinámico, a través de herramientas que facilitan la actualización de contenidos mediante bases de datos, archivos u otras fuentes de información.

3.3.5.4 *Apache Tomcat* – Un contenedor de *Servlet*

Tomcat es un proyecto de la organización *Apache*, distribuido libremente por *apache.org*. Es una herramienta de administración que permite utilizar y manipular *Servlets* y JSP (*JavaServer Pages*), así como listar, instalar, recargar, desarrollar y remover aplicaciones orientadas a Web.

Tomcat es un contenedor de *Servlets* con un entorno JSP, que alberga una o varias aplicaciones Web y constituye una estructura de directorios, en los que se encuentran todos los archivos necesarios para la ejecución de la aplicación Web. Así, el contenedor es parte del servidor Web o de aplicaciones y se encarga de la conectividad con la red, captura de requerimientos *HTTP*, gestión del ciclo de vida de *Servlets*, manejo de errores, suministro de seguridad, manipulación de las páginas JSP, entre otras funciones. El contenedor de *Servlets* se puede definir como un *Shell* de ejecución que maneja e invoca *Servlets* y JSP por cuenta del usuario.

3.4 Diseño de la Aplicación Web para Experimentación Remota

Con base en los requerimientos especificados en la sección 3.3, se ha desarrollado una aplicación cliente/servidor con las siguientes características: una página Web construida mediante un gestor de contenidos, la inclusión de herramientas Web 2.0 por la integración de un gestor de aprendizaje, el uso de *Servlets* de *Java* para extender las funciones del servidor y lograr establecer sincronía entre los dos servidores usados y el protocolo de comunicación con el hardware para experimentación. Entre las funciones adicionales consideradas se encuentran alertas al usuario en procesos de configuración, inserción de funciones de enlace con utilidades de *faceboook* y algunos *scripts* de diferentes lenguajes de programación Web como *php, JavaScript* y *html.*

3.4.1 Modelo Propuesto

El modelo propuesto plantea tres tipos de usuarios: una categoría denominada *USUARIOS DOCENTES*, en la que se encuentran uno o más profesores que administran el sitio, adicionando o eliminando contenidos, creando salas de chats, planteando discusiones en foros, asignando notas, entre otras funciones de administración. Por otra parte, están los *USUARIOS ESTUDIANTES*, que hacen uso de los contenidos ofrecidos por los docentes, así como del laboratorio.

El laboratorio remoto involucra hardware que puede ser utilizado por un usuario a la vez. Para garantizarlo, se deben crear citas para el acceso de los estudiantes al recurso hardware.

Cuando un estudiante ingresa el sitio Web, el sistema verifica a través de la aplicación si el usuario actual ha solicitado una cita previamente; también se verifica la hora correcta de la cita, para permitirle el ingreso a partir de la misma. Cuando el estudiante ingresa al sitio en la hora de la cita solicitada, éste puede hacer uso del hardware disponible para la experimentación en sistemas digitales; sin embargo, cuando un usuario no tiene cita, éste puede hacer uso de todas las herramientas que el sitio ofrece, excepto del acceso a la tarjeta de experimentación. El diagrama de flujo que se presenta en la Figura 3.5 muestra en detalle este procedimiento.

3.4.2 Gestión de Usuarios

El sitio Web es de carácter educativo por lo que es indispensable la inscripción de muchos estudiantes que hacen parte de los cursos dictados en la página Web. También puede resultar necesaria la inscripción de

más de un docente administrador, dado que pueden generarse muchos cursos diferentes; por lo tanto, al existir dos tipos de usuarios existe la necesidad de clasificar los contenidos presentados a cada grupo de ellos, así como los permisos que se les otorga; por ejemplo, un docente debe tener la posibilidad de adicionar información relevante a los estudiantes como guías de laboratorio, tareas, entre otras funciones, mientras que un estudiante debe solo poder visualizar este contenido para posteriormente desarrollarlo. En el gestor de contenidos es posible crear categorías y otorgar permisos de administración o de usuario registrado. Por tanto, se crearon dos categorías: ***DOCENTES*** y ***ESTUDIANTES.*** Los usuarios que se clasifican en la primera tienen permisos de *público*, *Gestor* y *Administrador*, mientras que la segunda posee solo permiso de *público* y de *Registrado*.

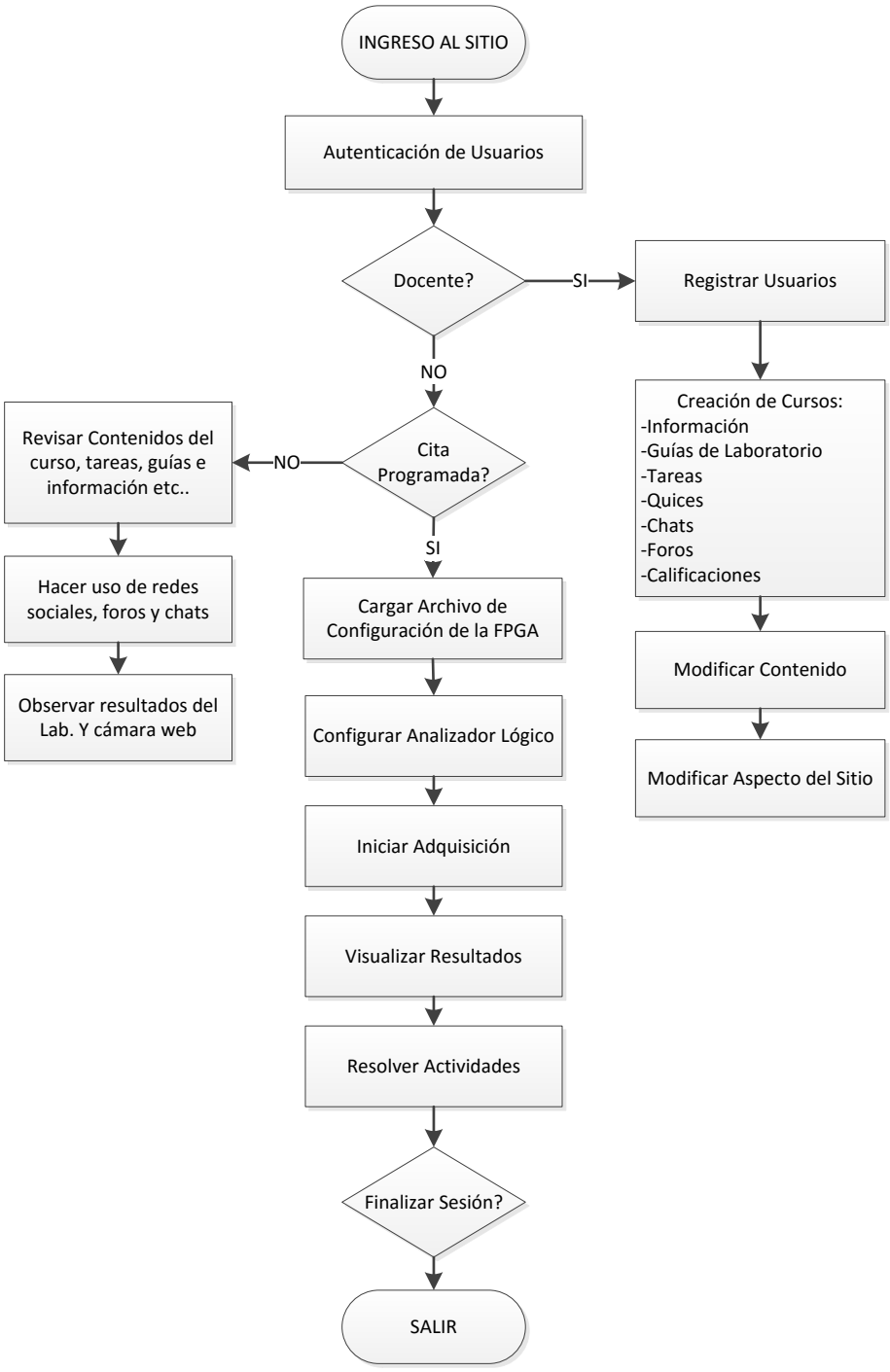


Figura 3.5 Diagrama de flujo principal de la aplicación software del Laboratorio Remoto

Con las categorías creadas, se adicionan los usuarios que correspondan a cada una. En un CMS, como *Joomla*, la vinculación de nuevos usuarios resulta muy sencilla para un administrador del sitio, ya que se hace desde el *backend* (ver Figura 3.6) ingresando datos como el nombre del estudiante, nombre de Usuario,

el correo electrónico, una contraseña inicial por defecto (modificada posteriormente por el propio usuario) y la categoría a la que el usuario será inscrito, si es docente o estudiante (ver Figura 3.7).

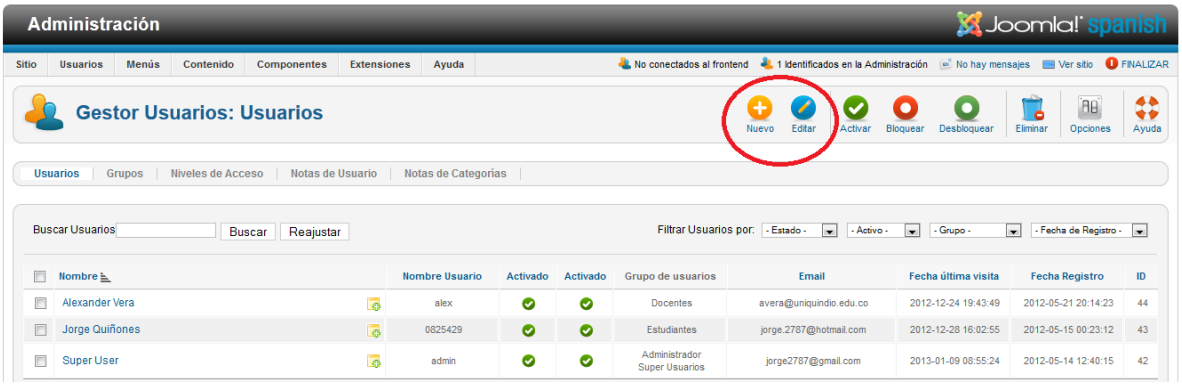


Figura 3.6 Gestión de usuarios desde Joomla

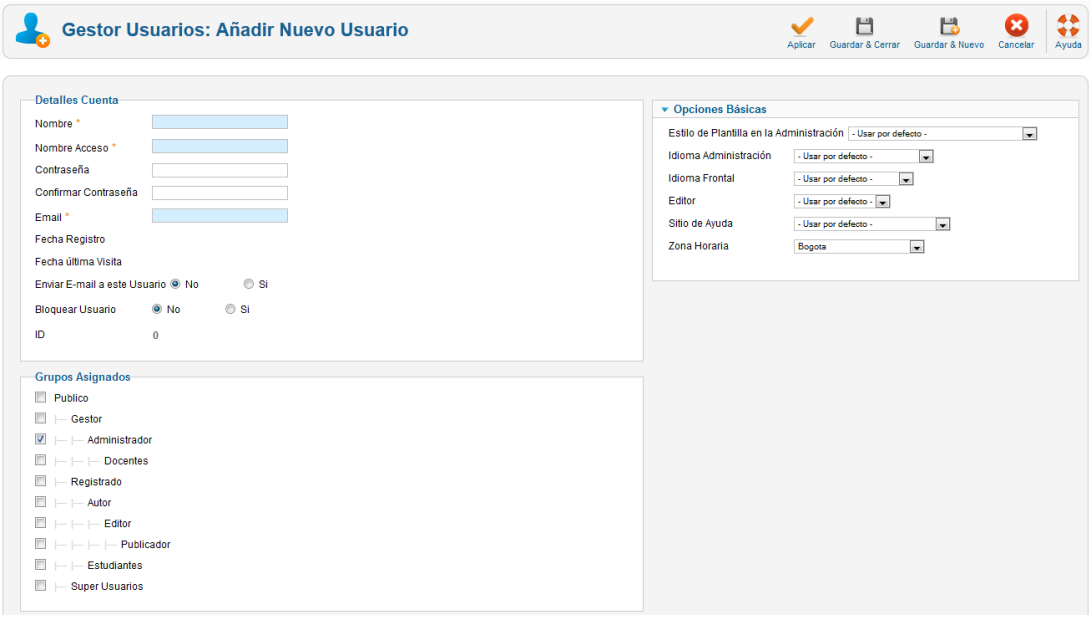


Figura 3.7 Adicionar usuarios y asignación de permisos

Los usuarios docentes puedan adicionar otros usuarios desde el *frontend*, debido a que puede resultar ambigua la administración desde el *backend*, lo que le facilita el manejo de la página al docente (ver *Figura 6.2*). Cuando un usuario ha sido adicionado, existe la opción de enviar un correo a cada estudiante inscrito con la contraseña asignada, con el fin de que éste ingrese y cambie esa contraseña, mediante la edición del perfil (ver *Figura 6.3*), por una más segura que solo él conozca. Una vez el usuario realiza el cambio de contraseña, ésta se modifica en la base de datos del sitio, utilizando el algoritmo de criptografía ***MD5*** para el cifrado de las contraseñas.

3.4.3 Gestión de Citas

El hardware utilizado en el laboratorio no admite acceso concurrente de múltiples usuarios, por lo que se ha implementado un sistema gestor de citas que permite que cada usuario registrado solicite una de estas al sistema, el cual evalúa disponibilidad de fechas y de horarios.

Inicialmente, el usuario registrado ingresa al sitio Web en el enlace "*laboratorio*", que lo direcciona a un *script* en *php*, cuya función es lanzar una consulta a la base de datos para verificar si el usuario actual tiene o no una cita asignada. Si en la base de datos no aparece ningún registro del usuario, se presentará una página en la que se informa que nunca ha solicitado una cita en la aplicación (ver *Figura 6.4*). Por otra parte, si existe un registro en la base de datos de una cita del usuario actual, pero no es coherente con la fecha y hora actual se presenta un mensaje indicando la situación (ver *Figura 6.5*) y posibilita al usuario para pedir una cita futura. Finalmente, si la consulta en la base de datos indica que el usuario tiene una cita en la fecha actual pero la hora no coincide entonces se presenta el aviso respectivo al usuario indicándole la fecha y hora de su cita (ver *Figura 6.6*).

En el caso en que el usuario no tenga ninguna cita y quiera solicitar una para poder acceder al laboratorio remoto, está disponible el enlace "***CITAS*",** desde el que se presenta al usuario un calendario que le permite escoger un mes y un día especifico que se ajuste a su calendario de actividades (ver *Figura 6.7*). Una vez el usuario ha seleccionado la fecha más conveniente para él, debe seleccionar la hora a través de una página que contiene las horas de disponibilidad del laboratorio (ver *Figura 6.8*).

Cuando el usuario ha seleccionado la fecha y hora de su cita, ésta se registra en la base de datos para su posterior uso con los datos personales del mismo (datos que no son modificables por el usuario, se le presentan como información) como *nombre, usuario* y *correo* (ver *Figura 6.9*). Adicionalmente, el sistema envía un correo electrónico de recordatorio al usuario que solicitó la cita.

Si el usuario que ingresa en un momento dado tiene asignada una cita en la franja horaria actual, este puede hacer uso del sistema sin restricciones; sin embargo, si quien ingresa al laboratorio lo hace faltando treinta minutos para finalizar su cita, el gestor de citas no le permitirá el ingreso. Esto se hace para evitar posibles conflictos con el siguiente usuario agendado para usar el laboratorio. En la Figura 3.8 se presenta el diagrama de flujo del funcionamiento del gestor de citas.

3.4.4 Gestión de Contenidos

El sitio Web ha sido construido con un sistema gestor de contenidos (*CMS*) que permite administrar la información presentada al usuario de forma rápida y sencilla. En el núcleo de *Joomla* aparece como contenido los artículos, categorías, enlaces, contactos y *feeds*, también los módulos son utilizados para crear contenidos. Adicionalmente, existe un gestor de multimedia para enriquecer el mismo con imágenes, archivos y videos [87].

Para la construcción de la aplicación Web los principales contenidos utilizados son los artículos y los módulos. Para el caso de los artículos, se adicionan los textos, enlaces o imágenes que se deseen tal como se hace en un editor de textos como *Open Office Write* o *Microsoft Word* (ver Figura 3.9). No obstante, mediante la vinculación de un componente externo al gestor de contenido es posible insertar en los artículos *scripts*, bien sea en *php*, *html* o *JavaScript*, para así lograr artículos con contenido dinámico como subida de archivos, consultas a bases de datos entre otras funciones.

Los módulos son muy útiles y flexibles debido a sus múltiples funciones, entre las que se destacan la creación de menús para formularios de acceso y los vínculos a los que un usuario puede acceder en el sitio Web.

3.4.5 Integración Web 2.0

Considerando la evolución que ha tenido el internet en cuanto a la forma de presentar la información en una página Web y, consiguientemente, la creación de innumerables herramientas que pueden contribuir en procesos educativos, resulta importante vincular algunos de los elementos de la Web 2.0 en la aplicación.

Esto redundaría en motivación para la vinculación de los estudiantes a los procesos de aprendizaje y desarrollo de competencias experienciales, abordando los temas ofertados.

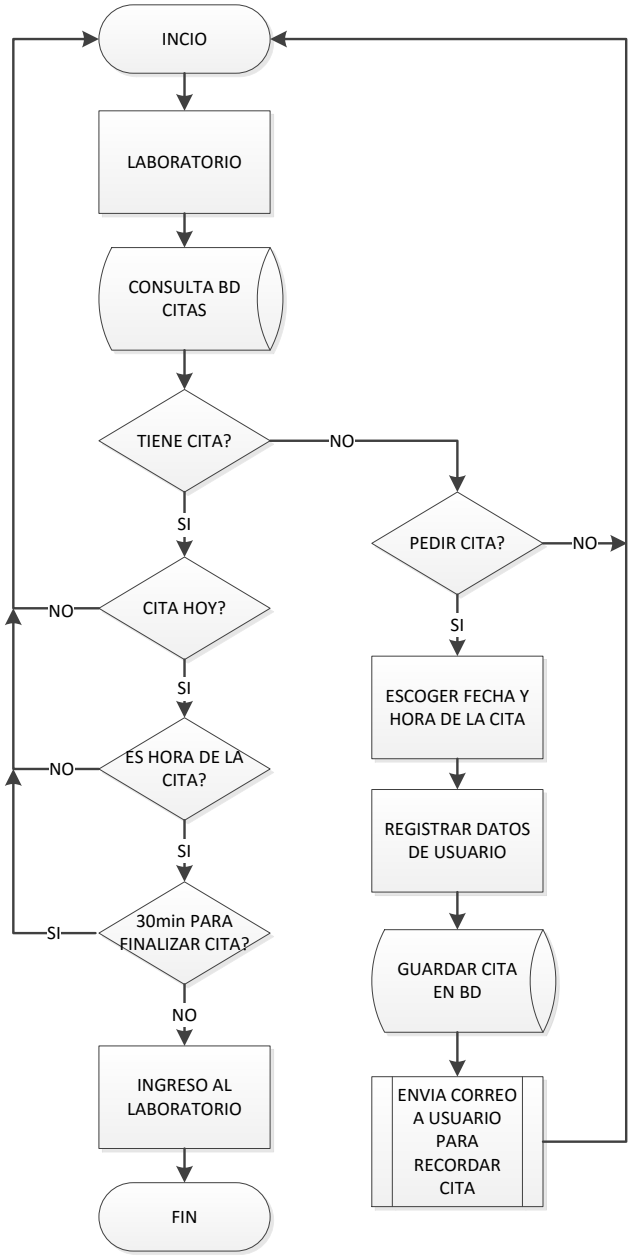


Figura 3.8 Diagrama de Flujo del Gestor de Citas

Actualmente, los usuarios de internet se caracterizan por una constante interacción con el resto de internautas, a través de foros, blogs, chats, comentarios entre otros. Por ello, resulta importante la vinculación de estas herramientas en la página Web, ya que incentiva el trabajo en grupo, dadas las características interactivas.

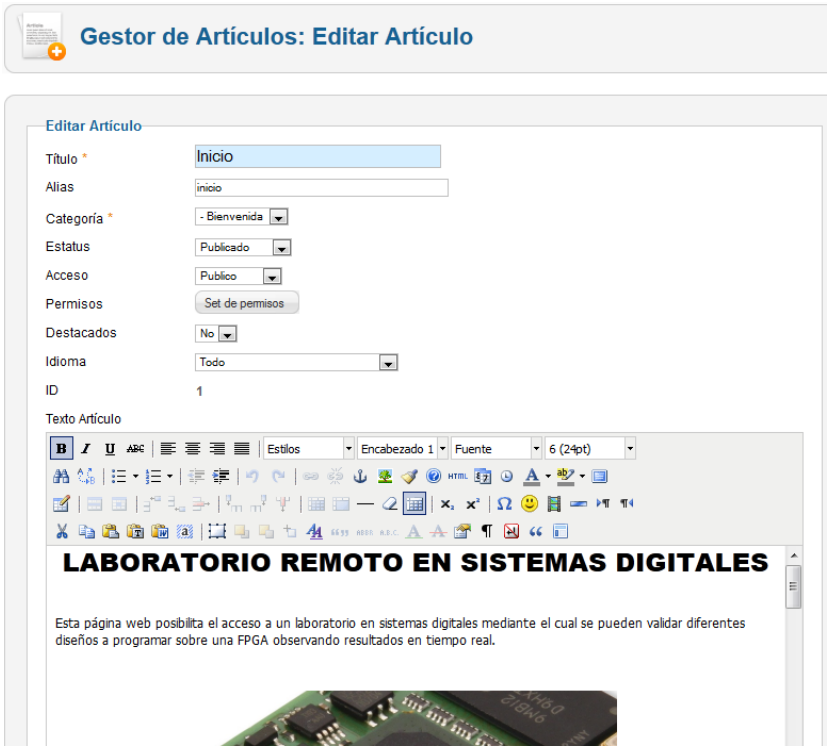


Figura 3.9 Gestor de contenidos de Joomla

Teniendo en cuenta que las herramientas Web 2.0 ofrecen ventajas importantes en el contexto educativo (ver Figura 3.2), éstas se han vinculado a la aplicación involucrando además a la red social *Facebook*. Las principales características de esta poderosa red social es la posibilidad de interactuar, compartir recursos, participar y colaborar; por lo tanto, en un proceso de aprendizaje puede ofrecer las siguientes ventajas [88]:

- Posibilidades de comunicación inmediata entre los participantes
- Facilidad para el intercambio de recursos
- Facilidades para la publicación de material creado para cumplir con los objetivos de aprendizaje
- Posicionamiento de uso en los alumnos.

Facebook ofrece una extensa documentación en su página *Facebook DEVELOPERS* para todos los desarrolladores que deseen incluir las funciones de la red social en determinada aplicación, ya sea para sitios Web, para aplicaciones móviles o aplicaciones para el mismo *Facebook* (ver Anexo 2.A).

Para integrar exitosamente *Facebook* en la aplicación Web, es necesario tener una cuenta en la red social. En esta cuenta se debe crear una aplicación cuya creación se facilita desde la sección *DEVELOPERS*, a la aplicación se le da un nombre, un correo de contacto y es muy importante entregar la *URL* en la que se encuentra alojado el sitio (ver Figura 3.10 superior).

En la Figura 3.10 inferior se observa que una vez la aplicación ha sido creada, se entrega un código de identificación ***APP ID***, que es un requisito indispensable para la integración de la página Web con funciones de *Facebook*. La herramienta *developers* entrega un bloque de código de acuerdo con los requerimientos de enlace a la red social; por ejemplo, para insertar el botón de iniciar sesión, el acceso a dejar comentarios o un *LIKE (Me Gusta)*. Se observa que se especifica el campo en el que se debe ingresar el ***APP_ID*** así como el domino de la página Web.

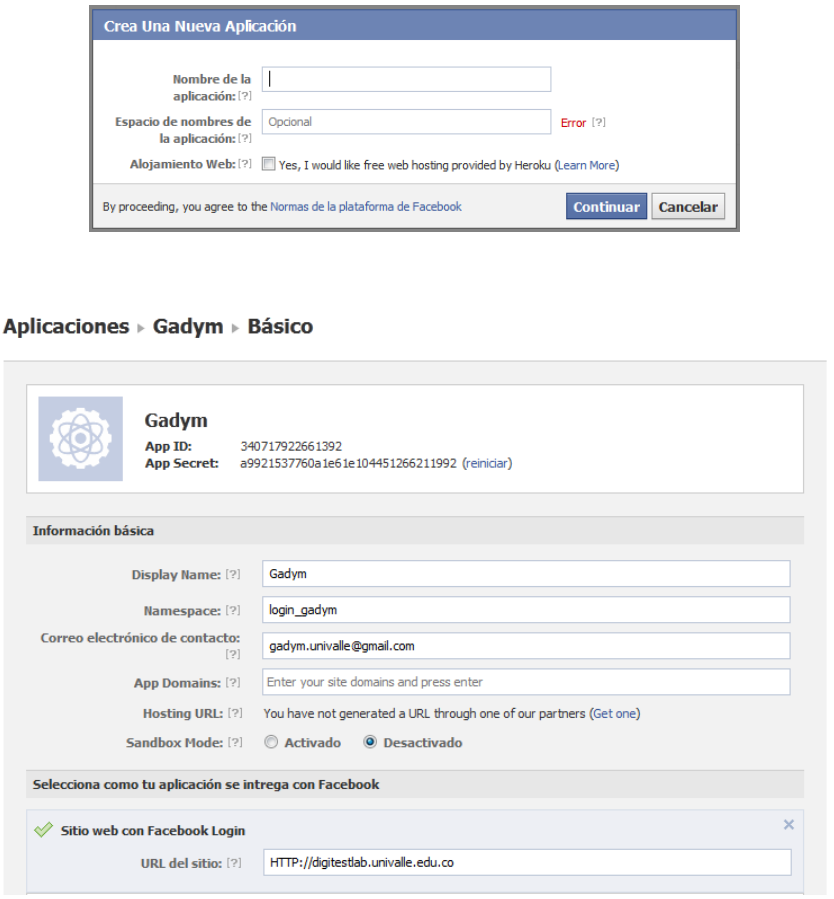


Figura 3.10 Creación de una Aplicación en Facebook

En la Figura 3.11 se observan las funciones de *Facebook* que han sido integradas a la aplicación desarrollada, con las que un usuario puede iniciar sesión con su cuenta personal, dejar comentarios, dar "*Me Gusta"* a determinados contenidos o *subscribirse* a la página en *Facebook* del grupo de investigación ***GADyM***.

Figura 3.11 Funciones de Facebook implementadas en la aplicación

3.4.6 Integración CMS – LMS

Tomando en consideración lo expuesto en la sección 2.2, herramientas como chats, foros, wikis resultan importantes y convenientes en un proceso de enseñanza actual. Por esta razón, estas herramientas son integradas, aunque no de igual forma que *Facebook*, a través de un sistema gestor de aprendizaje o *LMS*. Para la vinculación de estas herramientas se requiere inicialmente integrar el *LMS* con el gestor de contenidos y posteriormente adicionar los elementos *Web 2.0* que aportan interactividad a la aplicación permitiendo la constante realimentación entre todos los usuarios de la aplicación ya sean estudiantes o docentes.

Para que el gestor de aprendizaje *Moodle* pueda establecer comunicación con el gestor de contenidos *Joomla* deben realizarse una serie de pasos de configuración en ambas herramientas, esto con el fin de que mediante la extensión ***joomdle*** pueda gestionarse contenido del *LMS* para que sea visible en el *CMS*, la configuración básica para lograr este objetivo se describe a continuación

3.4.6.1 Instalación de *Joomla* y *Moodle*

Inicialmente para la instalación tanto de *Joomla* como de *Moodle* deben crearse las bases de datos que contendrán toda la información requerida por los mismos (ver Figura 3.12). Esto se hace mediante la herramienta del paquete *WampServer* ***phpMyAdmin***.

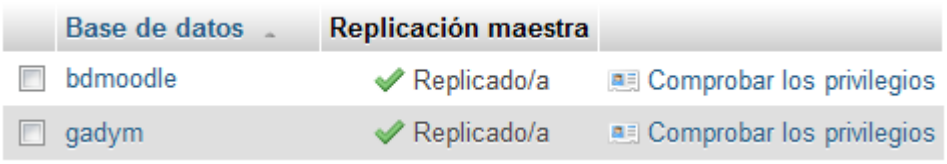


Figura 3.12 Bases de datos requeridas para la Instalación de los gestores

Una vez las bases de datos se han generado, la instalación de *Joomla* y *Moodle* resulta muy intuitiva, solo se debe prestar especial atención a la conexión con las bases de datos respectivas indicando el nombre de las mismas, el usuario utilizado y la contraseña indicada a la hora de la creación de estas.

Una vez instalados los gestores de contenido y de aprendizaje en el servidor, se procede con la configuración de ambos para que *Joomla* se comunique con *Moodle* y viceversa; para esto, primero se procede con la instalación del paquete *joomdle*, que contiene componentes, módulos y *plugins* reconocibles por ambos gestores, la instalación de éste en *Joomla* se hace desde su *Gestor de extensiones*. Una vez instalada la extensión, se ejecuta el proceso de configuración inicialmente desde *Joomla* para conseguir la integración (ver Anexo B).

En cuanto a la configuración en *Moodle*, inicialmente se debe instalar el *plugin joomdle*, la instalación del *plugin* se realiza mediante la copia del archivo que entrega el autor de *joomdle* en la carpeta *AUTH* de este gestor y, una vez instalado, se procede con su habilitación (ver Anexo C).

Finalizada la etapa de configuración de los gestores de contenido y de aprendizaje, se procede con la configuración del paquete *joomdle* para la integración entre *Moodle* y *Joomla*. La información que se debe proporcionar durante el proceso de configuración son la ruta donde se encuentra instalado *Moodle* para *Joomla* y la ruta en la que se encuentra instalado *Joomla* para *Moodle*. Esto es parte importante en la integración de ambas herramientas; sin embargo, no es suficiente ya que se requiere de la creación de un usuario denominado *usuario conector*.

Para la creación de un usuario conector, debe adicionarse un *servicio Web* con determinados permisos y funciones que posteriormente se le confieren a este usuario. Una vez creado el servicio y el usuario, se crea lo que *Moodle* denomina un *token*, que entrega una ficha con la que se logra que el *LMS* identifique a *Joomla*.

La ficha que *Moodle* entrega al *usuario conector* debe insertarse en la configuración de ***joomdle*** sobre *Joomla*. Otro aspecto importante es que los *métodos de conexión* deben coincidir en ambos gestores; las opciones son *cURL* y *file_get_contents*, cualquiera que se utilice debe corresponder en ambas herramientas (ver Figura 3.13 y Figura 3.14).

Figura 3.13 Configuración de Moodle para la Integración con el CMS

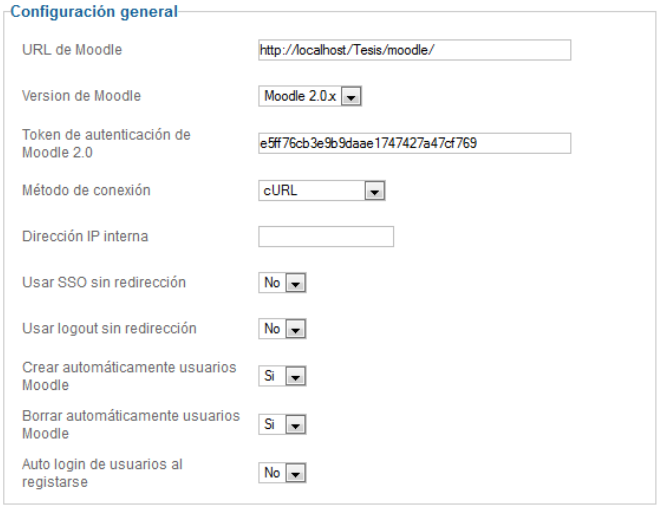


Figura 3.14 Configuración de *Joomla* para la Integración con el LMS

Cuando el estado de todas las comprobaciones realizadas es correcto, desde ese momento es posible ver contenido de *Moodle* en *Joomla*; es decir, la integración ha sido realizada exitosamente.

Una ventaja que ofrece *joomdle* para inspeccionar si la configuración se realizó correctamente, es la comprobación del sistema que se puede hacer desde *Joomla*. Esta comprobación indica el estado de todas las funciones que deben estar configuradas correctamente para la integración de los gestores. Entre varias de estas funciones se encuentra la que mayores inconvenientes puede presentar, que es la de servicios Web de *joomdle*, que requiere la configuración mencionada previamente.

Un aspecto que resulta negativo en esta integración, es que con las plantillas que dan la apariencia del sitio que trae *Moodle* por defecto, la visualización para el usuario no resulta adecuada ya que lo que se visualiza es toda la pantalla del *LMS* pero inmersa en el área del contenido principal de *Joomla* (ver Anexo D).

Para corregir este problema de visualización, se decidió trabajar con la plantilla *estándar* del *Moodle* pero editando algunos de los archivos **.php* de esta plantilla con el fin de eliminar los módulos de las barras laterales de la página, pero solo para los usuarios estudiantes y no para administradores. Con las modificaciones pertinentes, el resultado es considerablemente mejor para el momento en el que un usuario navega por la aplicación (ver Anexo D).

En cuanto a la administración de usuarios, tanto *Moodle* como *Joomla* manejan una tabla de usuarios registrados en sus respectivas bases de datos de forma independiente, pero mediante la extensión *joomdle* es posible integrar los usuarios de ambas bases de datos, ya sea copiar los usuarios de *Joomla* a *Moodle* o el caso contrario mediante las opciones enmarcadas en el recuadro de la Figura 3.15.

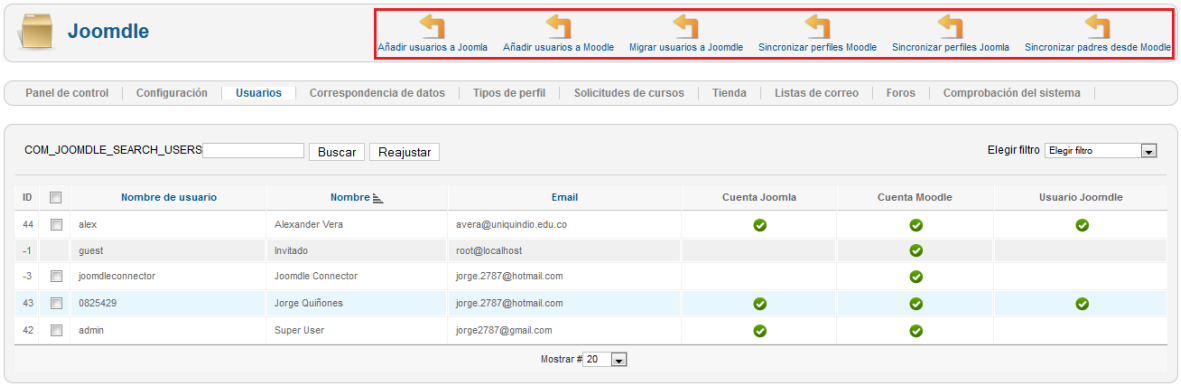


Figura 3.15 Sincronización de Usuarios de ambas Plataformas

Finalmente, con la configuración exitosa de la integración, con las mejoras aplicadas para una navegabilidad amigable del usuario y con sincronización en ambas plataformas, es posible adicionar el contenido deseado para ofertar un curso de un tema determinado. En la aplicación, el curso creado es del área de Sistemas Digitales (ver *Figura 6.14*).

3.4.7 Protocolo de comunicación Hardware

La aplicación involucra una tarjeta electrónica con todo el *hardware* requerido para la experimentación en sistemas digitales, como por ejemplo una FPGA, un Analizador Lógico y un Microcontrolador (*uC*). Este último es el encargado de la administración de todos los elementos de la tarjeta y de la comunicación con el servidor a través del protocolo USB; por lo tanto, es a través de éste que se manejan las configuraciones tanto de la FPGA como del Analizador Lógico y también el envío de resultados desde la tarjeta hacia el servidor.

El *uC* es un *PIC18F4550* que soporta comunicación USB y el fabricante *Microchip* entrega un *framework* [89] para el desarrollo de diferentes aplicaciones con él. Por lo tanto, la parte fuerte del desarrollo de comunicación se encuentra del lado del servidor, ya que desde ahí se debe manipular el protocolo USB para lograr comunicación con el *hardware*. Al realizar el proceso de búsqueda y selección de opciones para manejar puertos USB desde un *PC* se obtuvo una biblioteca llamada *libusb*. Se trata de una librería de *C* que proporciona un fácil acceso a las aplicaciones de los dispositivos USB en varios sistemas operativos y es un proyecto de código libre. Adicionalmente, esta biblioteca puede ser utilizada con los lenguajes de programación presentados en la Figura 3.16 [90].

Language	Details
Ada	Libusb 0.1 bindings for Ada
C(/C++)	Provided by the libusb library itself
C#	libusbdotnet for C#, DotNet and Mono
Common Lisp	Common Lisp bindings to libusb-0.1
Factor	Factor libusb-1.0 binding
Go	Go libusb-1.0 wrapper
Haskell	Low level, high level
Java	Java libusb(-0.1)/libusb-win32 wrapper
Java	Java libusb-1.0 bindings
Lua	lua libusb-1.0 binding
node.js	node.js libusb-1.0 binding
OCaml	OCaml libusb-1.0 binding
Perl	Perl bindings for libusb-0.1
Python	pyusb for Python
Python	partial python wrapper using swig
Python	python-libusb1 pure-python wrapper. Supports asynchronous API, all transfer types, all platforms supported by libusb-1.0 + FreeBSD
Ruby	Ruby bindings for libusb-0.1
Ruby	Ruby bindings for libusb-1.0

Figura 3.16 Lenguajes que soportan la biblioteca para Comunicación USB

Dado que el servidor que está conectado al *hardware* es un PC, con sistema operativo Windows de 32 bits, y que de acuerdo con las necesidades de extender las funciones del servidor Web para la comunicación con el *hardware* se hace uso de los *servlets* de Java, la biblioteca seleccionada es *Java libusb(-0.1)/libusb-win32 wrapper*. Esta biblioteca proporciona las clases *Java* necesarias para acceder a *libusb/libusb-win32* través de la interfaz nativa de *Java*. La clase *ch.ntb.usb.LibusbJava* carga la librería compartida y proporciona la interfaz nativa de acceso a *libusb*.

En *libusb*, la estructura del bus (con los dispositivos, las configuraciones, las interfaces y los *endpoints*) es representada en estructuras de C. Para cada estructura, se crea un objeto de Java (llamado *Usb_xxx*) y se copia la información a este objeto. Esto se hace cuando se llama a *LibusbJava.usb_get_busses()*. La información acerca de los buses y los dispositivos conectados se puede obtener de un árbol como la estructura de objetos de *Java*.

La biblioteca proporciona una sencilla Clase *Device* que representa un dispositivo USB y permite que sea fácil escribir y leer de él. Errores y *timeouts* resultan en excepciones de *Java*.

Esta biblioteca se ha utilizado en dos proyectos de la Universidad de Ciencias Aplicadas de Tecnología NTB (Interfaz de programación y depuración (*Cypress FX2*) [91] y *Board Experimental* (AVR AT90USB1287)) los cuales utilizan transferencias masivas (*bulk*) [92].

Al descargar la biblioteca, que se entrega en un archivo *zip*, se puede encontrar un paquete de *Java* llamado ***ch.ntb.usb-0.5.9.jar*** que contiene el código de soporte para el protocolo USB, además de los archivos ***Libusb.dll*** y ***libusb.sys*** requeridos por Windows para el manejo del puerto. En el diseño de la aplicación se

utilizan las funciones que esta biblioteca establece para identificación de un dispositivo específico, la escritura y lectura de datos, entre muchas otras funciones disponibles.

3.4.7.1 Funciones de la biblioteca *Libusb* de *Java*

El API de la biblioteca USB es muy intuitivo y está diseñado para tener analogías muy cerradas con las de la especificación o estándar del protocolo dado el nombre usado en cada Clase. A continuación se da una breve descripción de las principales funciones usadas de esta biblioteca. Estas funciones comprenden el núcleo del *libusb* y son utilizadas por todas las aplicaciones que la utilicen.

- ***Función void usb_init()***. Tal como su nombre lo indica, esta es la función que establece algunas estructuras internas para iniciar la comunicación, esta debe ser llamada antes de cualquier otra función.
- ***Función int usb_find_busses(void).*** Esta es la función que se encarga de encontrar todos los buses en el sistema, esta retorna el número de cambios desde la última llamada de la función (total de nuevos buses y buses removidos).
- ***Función int usb_find_devices(void).*** Esta función es la encargada de encontrar todos los dispositivos que se encuentren en cada bus, esta debe ser llamada después de la función *usb_find_buses()*, adicionalmente retorna el número de cambios ocurridos desde el llamado previo de la función (total de nuevos dispositivos y dispositivos removidos).
- ***Función usb_get_busses().*** Esta función simplemente retorna el valor de la variable global usb_busses con la que se establece el bus que se va a usar para la transferencia de datos.

El siguiente conjunto de instrucciones son para tratar con un dispositivo USB. Estas instrucciones permiten abrir y cerrar un dispositivo, así como las operaciones estándar como establecer los ajustes de configuración, alternar parámetros, entre otras. Adicionalmente, proporcionan operaciones al nivel del sistema operativo como solicitud y liberación de interfaces.

- ***Función usb_open()***. Esta función es usada para abrir un dispositivo para su uso. Es indispensable llamar esta función antes de intentar realizar algún tipo de operación con el dispositivo, esta retorna un identificador o *handle* que será usado en las comunicaciones futuras con un dispositivo.
- ***Función usb_close().*** La función *usb_close()* cierra una conexión que fue abierta con *usb_open(),* no se puede realizan ninguna operación una vez se ha llamado esta función. Retorna 0 si se ejecuta exitosamente, pero en caso de error retorna un valor menor que 0.
- ***Función usb_claim_interface().*** Esta función solicita la interfaz con el sistema operativo. El parámetro de la interfaz es el valor que se ha especificado en el campo descriptor *bInterfaceNumber*, de la interfaz USB. La función retorna 0 si la interfaz le es entregada, de lo contrario entrega un valor menor que 0. Esta función debe ser llamada antes de intentar realizar alguna operación con la interfaz. Finalmente, ella entrega el código *UBUSY*, si la interfaz no está disponible, o *ENOMEM* cuando no hay espacio de memoria suficiente.
- **Función *usb_release_interface().*** Libera una interfaz que previamente ha sido solicitada por la función anterior. El parámetro de la interfaz es el valor que se ha especificado en el campo descriptor *bInterfaceNumber*. Retorna 0 en acierto o un valor menor que 0 en error.

El siguiente conjunto de funciones le permite a una aplicación enviar mensajes al *pipe* de control de USB por defecto:

- ***Función usb_control_msg().*** Realiza una solicitud de control al pipe de control por defecto en un dispositivo. Los parámetros que reflejan los tipos del mismo nombre en la especificación USB. Devuelve el número de bytes leídos /escritos o un entero menor que cero en caso de error.
- ***Función usb_get_descriptor().*** Esta función solicita un descriptor de un dispositivo identificado por el tipo y el *índex* del descriptor del *pipe* de control por defecto. Retorna el número de bytes leídos del descriptor.

El siguiente par de instrucciones permiten a una aplicación enviar y recibir datos por los *pipes bulk* USB.

- ***Función usb_bulk_write().*** Realiza una solicitud de escritura masiva o *bulk* a un *endpoint* especificado por un parámetro de la función. La función retorna el número de bytes escritos con éxito o un entero menor que cero en caso de error.
- ***Función usb_bulk_read().*** Esta función realiza una solicitud de lectura masiva a un *endpoint* específico. Retorna el número de bytes leídos con éxito o un entero menor que cero en caso de error.

Existen otras funciones que están especificadas en el API de la biblioteca [93], pero se ha optado por especificar las más relevantes para este trabajo, considerando aquellas que se utilizan para establecer la comunicación con un microcontrolador.

3.4.7.2 Implementación del protocolo con la biblioteca *Libusb*

Con la biblioteca y las funciones descritas en la sección anterior, se desarrolló el protocolo que permite el envío y la recepción de datos desde y hacia el servidor de *hardware*. La Figura 3.17 presenta el diagrama de flujo del *software* en *Java* que permite ubicar un dispositivo específico en el bus del PC. La forma de ubicar un dispositivo en el bus es a través de sus parámetros *vendorID* y *ProductID*, que están especificados en la descripción del dispositivo (ver Figura 3.18) y en el driver del dispositivo.

Una vez el dispositivo ha sido identificado en el bus del OS y se encuentra listo para usar, se procede con la ejecución de cada uno de los pasos requeridos indicados en la Figura 3.19. Se puede observar que, en cada proceso ejecutado cuando el dispositivo se encuentra listo, existen unos subprocesos que han sido enumerados del 1 al 4; esto corresponde a los procesos de programar la FPGA, configurar el analizador lógico, enviar los estímulos al diseño bajo prueba y finalmente leer los datos capturados, usando el microcontrolador.

La Figura 3.20entrega una visión un poco más detallada de cada proceso, donde se observa que los pasos del 1 al 3, que son de escritura, tienen una estructura muy parecida y requieren conocer los datos a enviar para solicitar la interfaz; si ésta se encuentra disponible se puede hacer el envío de datos. Una vez que los datos han sido enviados por completo, se libera la interfaz para que el siguiente proceso pueda ejecutarse sin problemas.

Por otra parte, el paso de *Lectura de resultados (4)* es un poco diferente dado que la función ya no es *write* sino *read*; pero de igual forma, se envía el comando respectivo para que se inicie la lectura de datos desde el *hardware* y éstos se van recibiendo y almacenando en un buffer para posteriormente usarlos.

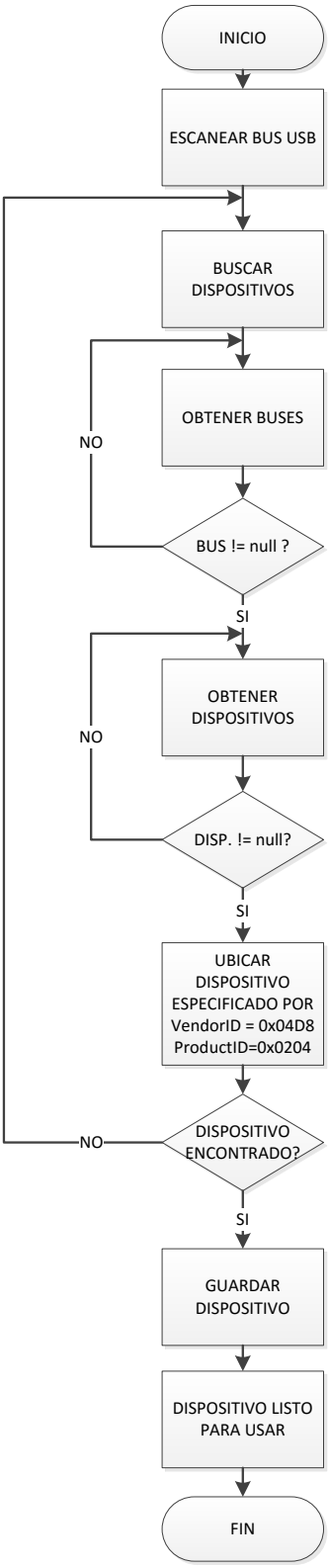


Figura 3.17 Diagrama de flujo para la identificación de un dispositivo USB

Descriptor	
Field	Value
bLength	18
bDescriptorType	1 (DEVICE)
bcdUSB	0x0200
bDeviceClass	0xFF
bDeviceSubClass	0xFF
bDeviceProtocol	0xFF
bMaxPacketSize0	64
idVendor	0x04D8
idProduct	0x0204
bcdDevice	0x00
iManufacturer	0x01
iProduct	0x02
iSerialNumber	0x00
bNumConfigurations	1

Figura 3.18 Parámetros *vendorID* y *productID* del dispositivo USB

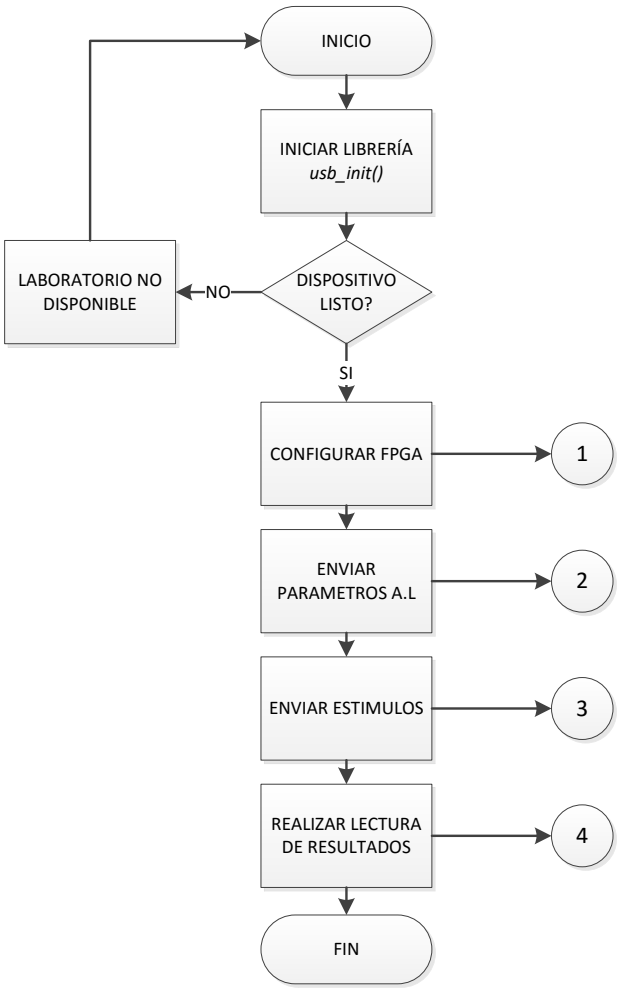


Figura 3.19 Diagrama de flujo de los pasos que se realizan con la biblioteca

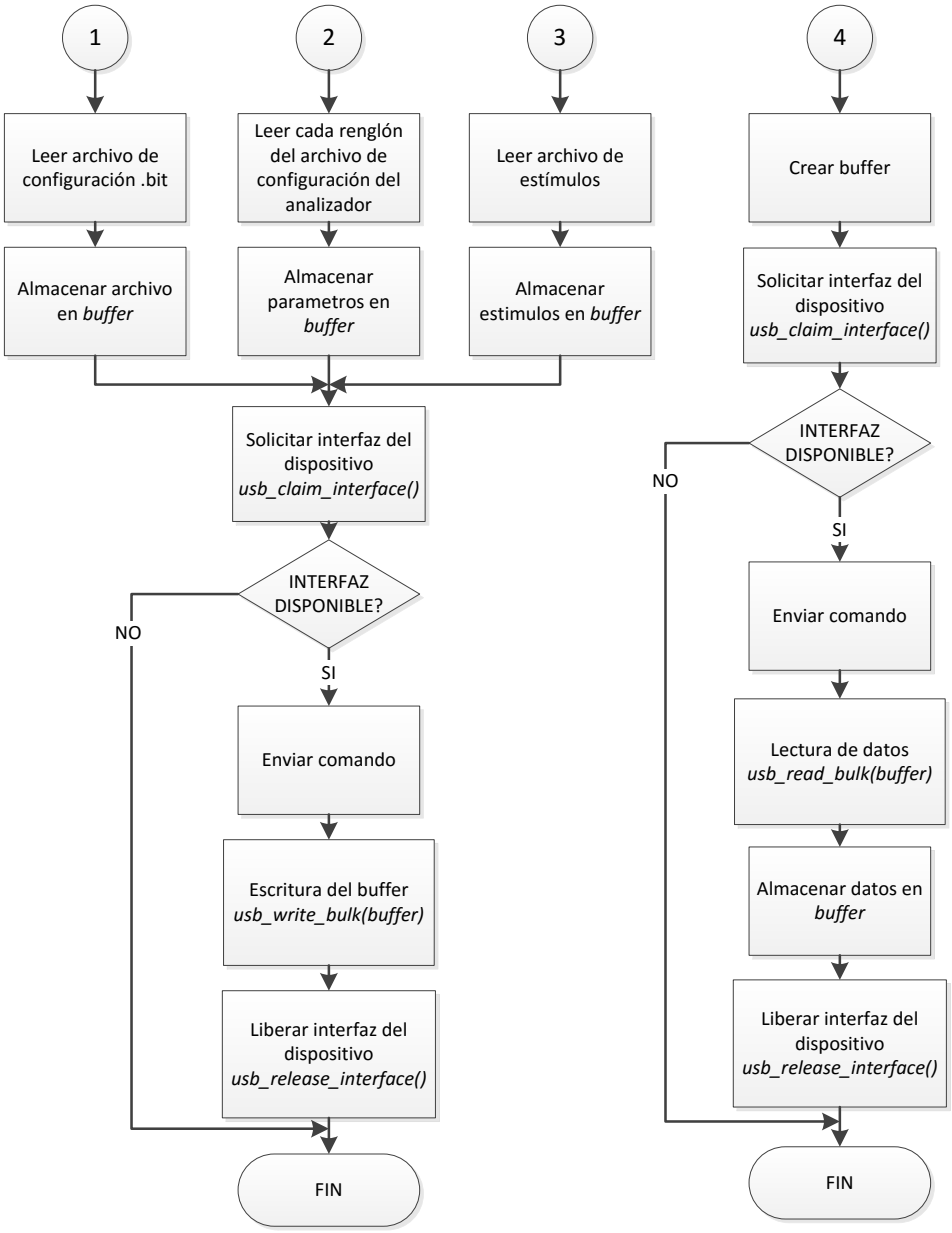


Figura 3.20 Descripción detallada de cada paso a ejecutar

Finalmente, en cuanto a la creación del protocolo, en la Figura 3.21 se presenta el diagrama de clases (ver Anexo 2.E) que explica los métodos involucrados en el mismo. Se puede observar la clase *Dispositivo*, que tiene la información relevante al microcontrolador, como su *vendorID* y *productID*, también la clase *EscaneaBus*, que se explica con el diagrama de flujo de la Figura 3.17; la clase *Lab*, que contiene las funciones para la solicitud y la liberación de la interfaz del dispositivo, así como funciones para el envío de los comandos. Finalmente, se tiene la clase *LabRemoto*, que hace uso de las demás clases y es la que contiene todas las funciones del laboratorio presentadas en la Figura 3.19 y la Figura 3.20. Es importante destacar los mensajes de control establecidos, que se pueden observar en la clase ***LabRemoto.java***: *ObtenerConfiguracionFPGA*(), *resetearFPGA*(), *estadoFPGA*() y *estadoAnalizador*().

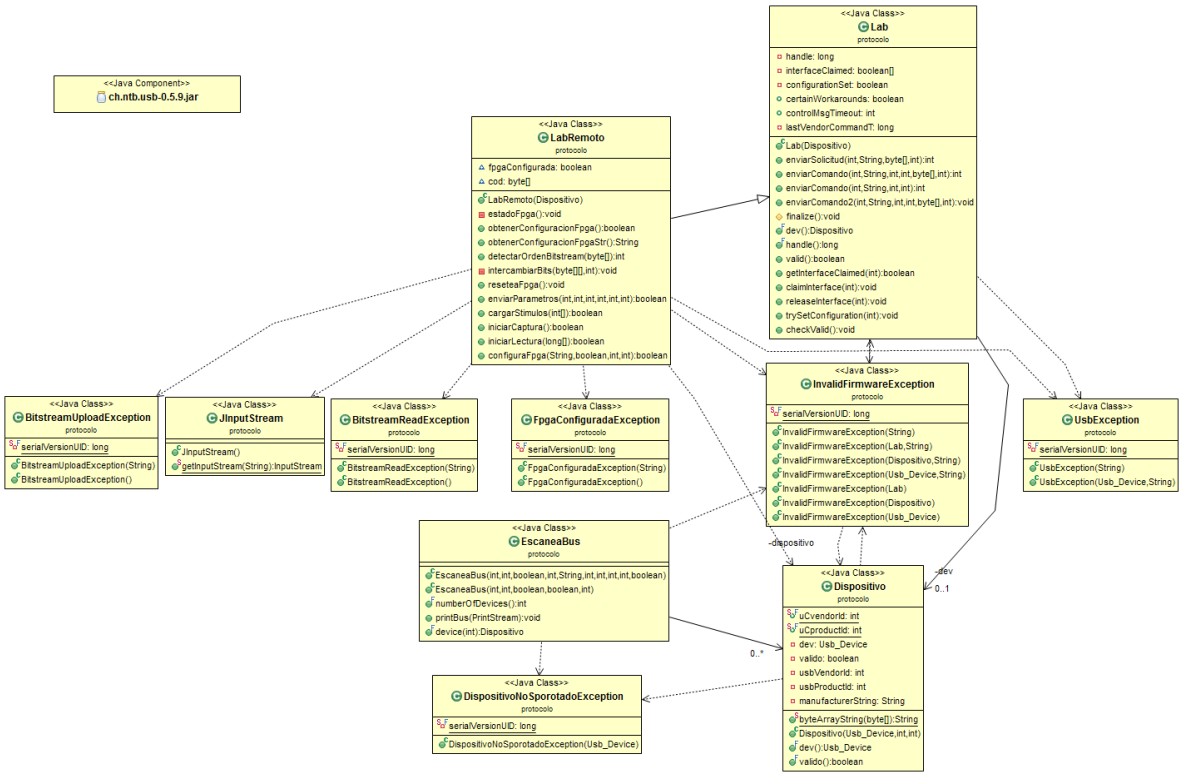


Figura 3.21 Diagrama de clases del protocolo USB creado con la librería *Libusb*

El estándar USB define unos mensajes de control ya establecidos que se utilizan para obtener alguna información de los descriptores como se presenta en la Figura 3.22; sin embargo, para esta aplicación se requiere que se puedan enviar mensajes en los que, al igual que los estándares, la respuesta sea inmediata y no se debe esperar a que alguna interfaz se libere ya que se puede perder sincronía del proceso. Por esta razón, al igual como se establece en la tabla de la Figura 3.22, se crearon cuatro *bmRequestType* para responder a determinadas solicitudes.

En el lenguaje *C* del microcontrolador se crearon en las rutinas que atienden los mensajes de control ya mencionados, funciones para hacer el *sensado* de niveles de voltaje de algunos pines o para que se ejecute una rutina que cambie los niveles de determinados pines. Si se requiere conocer algún estado a través de estos mensajes, se recibe una respuesta inmediata con lo que se refleja su gran aporte. En la Tabla 3.1se presentan los cuatro mensajes de control establecidos en el microcontrolador.

bmRequestType	bRequest	wValue	wIndex	wLenght	Data
1000 0000b	GET_STATUS (0x00)	Zero	Zero	Two	Device Status
0000 0000b	CLEAR_FEATURE (0x01)	Feature Selector	Zero	Zero	None
0000 0000b	SET_FEATURE (0x03)	Feature Selector	Zero	Zero	None
0000 0000b	SET_ADDRESS (0x05)	Device Address	Zero	Zero	None
1000 0000b	GET_DESCRIPTOR (0x06)	Descriptor Type & Index	Zero or Language ID	Descriptor Length	Descriptor
0000 0000b	SET_DESCRIPTOR (0x07)	Descriptor Type & Index	Zero or Language ID	Descriptor Length	Descriptor
1000 0000b	GET_CONFIGURATION (0x08)	Zero	Zero	1	Configuration Value
0000 0000b	SET_CONFIGURATION (0x09)	Configuration Value	Zero	Zero	None

Figura 3.22 Mensajes de control establecidos en el estándar USB

Tabla 3.1 Códigos implementados para los mensajes de control

Comando de control	***Descripción***
0x0C	*USB_GET_AL*
0x0D	*USB_GET_FPGA*
0x0E	*USB_RST_FPGA*
0x0F	*USB_CFG_FPGA*

3.4.8 Herramientas de control, configuración y visualización

Con el protocolo de comunicación establecido entre el servidor de control y el *hardware*, teniendo la capacidad de transferir datos de un lado al otro, se ha diseñado una herramienta tipo *wizard* que va guiando al usuario por cada uno de los pasos de configuración que debe seguir e informa de los posibles errores que se puedan cometer en este proceso. Para la operación del laboratorio, el usuario autenticado que accede al recurso inicia con el *upload* del archivo de configuración del dispositivo reconfigurable FPGA, continúa con la configuración del analizador lógico, la generación de un archivo específico desde la herramienta *ISE Design* de *Xilinx* para la generación de estímulos y, finalmente, debe interpretar los resultados obtenidos en la herramienta de visualización creada. El diseño de cada uno de estos procedimientos se describe a continuación.

3.4.8.1 Configuración FPGA

El laboratorio remoto está concebido para la realización de prácticas en sistemas digitales, en las que múltiples usuarios tendrían acceso a un *hardware* configurable (FPGA) para la programación y verificación de las funciones electrónicas que hayan diseñado. Por lo tanto, la aplicación debe suministrar las interfaces para recibir el archivo de configuración de la FPGA (de extensión **.bit*) para posteriormente configurarla o programarla.

La fase de la aplicación que permite la carga del archivo de configuración de la FPGA ha sido diseñada mediante un *script* en *html*, con el que se genera un formulario al que se le adiciona un campo para hacer

el *upload* de archivos (ver *Figura 6.16*). Existe también un código en *JavaScript* que lanza una alerta si el usuario está cargando algún archivo con extensión diferente de **.bit* (ver *Figura 6.17*). El botón "*ACEPTAR"* lanza una petición *POST* que va dirigida a un *Servlet* específico en el servidor *TOMCAT*, dependiendo de si se ha cargado el archivo correcto. El *Servlet* encargado de atender la petición *POST* se ha nombrado ***ArchivoConfiguracion.java*** cuyo diagrama de clases se puede ver en la Figura 3.23.

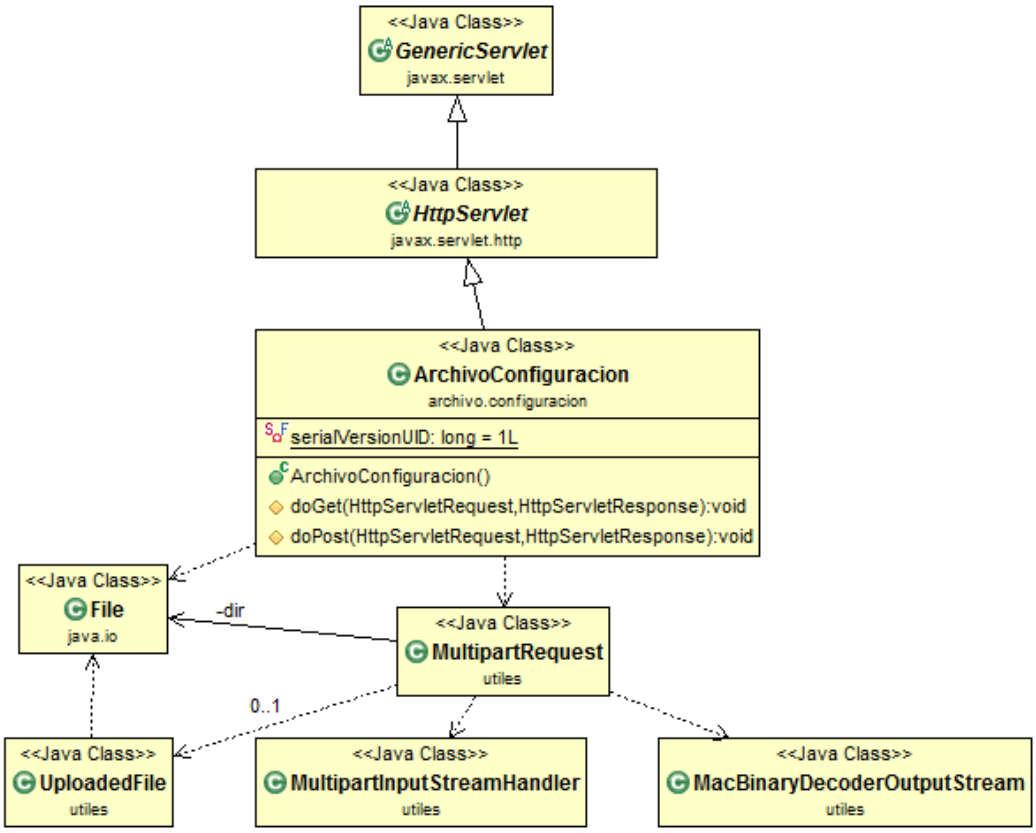


Figura 3.23 Diagrama de clases de la herramienta que permite cargar el archivo de configuración

En la Figura 3.23, se observa que la clase *ArchivoConfiguracion* hereda de la superclase *HttpServlet*; por lo tanto, se puede hacer uso de los métodos *doGet(HttpServletRequest, HttpServletResponse)* y *doPost(HttpServletRequest, HttpServletResponse)* con los que todo el código, que se encuentre en estos métodos, se ejecuta una vez que el *Servlet* se ha llamado a través de una petición *post* o *get*. Para este caso en particular, las peticiones son del tipo *post*, por lo que el método usado es el *doPost()*.

Una vez que el *Servlet* recibe la petición, ésta se atiende extrayendo el archivo subido y el nombre del usuario actual que también se envía en la petición. Extraer parámetros como el nombre de usuario y el archivo de configuración se hace posible gracias a la instanciación o el uso de la clase *MultipartRequest*, con la que se pueden capturar todos los parámetros enviados en la petición *POST*. Otra Clase involucrada en este *Servlet* es la *File*, que puede manejar un archivo de múltiples maneras; no obstante, para este caso la clase se instancia con el fin de renombrar el archivo recibido con el nombre del usuario actual. Esto es muy útil para que el proceso pueda distinguir el archivo que corresponde a cada usuario; así, cuando el archivo se ha renombrado, se copia en una ruta específica del servidor para su posterior uso. Finalmente, el *Servlet* envía una respuesta de redirección a través del atributo *HttpServletResponse*; es decir, se envía la

URL de una página alojada en el servidor y así se avanza en el proceso, siempre que no se presente algún error.

Cuando el *Servlet* se ha ejecutado, el proceso avanza hacia la carga del archivo de extensión **.ucf* (ver *Figura 6.18*). Este archivo es generado por la herramienta *ISE Design* de *Xilinx* y contiene información sobre la asignación de pines del FPGA que el usuario ha designado para las entradas y salidas de su diseño. Luego, se hace una verificación de correspondencia entre los pines asignados por el usuario y los que se encuentran realmente cableados para el análisis.

De igual manera que con el proceso de carga del archivo de configuración, el *Servlet* usado para este archivo (**.ucf*) tiene la misma función; inicialmente se tiene el *script* del formulario en *html* para el *upload* del archivo, la verificación de la extensión con *JavaScript*, el botón de *ACEPTAR* y finalmente, el llamado del *Servlet ArchivoPines*, cuya función es recibir el archivo, renombrarlo con el nombre del usuario actual, copiarlo en una ruta especifica del servidor y por último, redireccionar al siguiente paso del proceso, siempre que no se presenten errores.

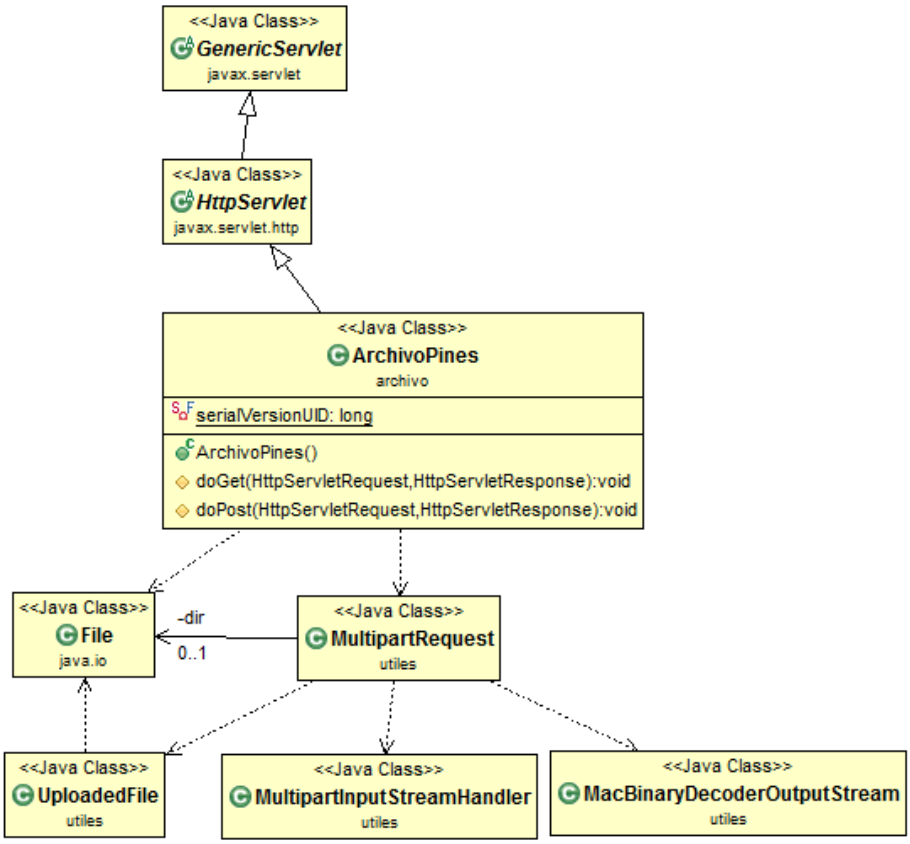


Figura 3.24 Diagrama de clases de la herramienta que permite cargar el archivo de asignación de pines

3.4.8.2 Configuración del Analizador Lógico y Generación de Estímulos (*TestBench*)

Esta etapa del laboratorio considera tres pasos: el *upload* de un archivo que el usuario debe generar y que permite construir los estímulos para el diseño bajo prueba, la configuración de los parámetros del analizador lógico para la captura de datos y la selección del modo de programación de la FPGA. Una vez que los archivos de configuración de la FPGA (**.bit*) y de asignación de pines (**.ucf*) han sido cargados en la

aplicación, se continúa con la configuración del analizador lógico, que requiere una especial atención debido a que el usuario debe ser consciente del significado de cada uno de los parámetros que específica para lograr los resultados esperados (ver *Figura 6.19*).

El usuario debe cargar un archivo de extensión ***.tit*** (*table input test*), que se genera con la herramienta de *Xilinx ISE Design* a partir de ciertas líneas de código específicas que deben ser adicionadas al código *vhdl* de prueba o *testBench.* En este paso el usuario tiene dos opciones, la primera es adicionar él mismo las líneas necesarias para la creación del archivo **.tit* o entregarle al sistema el archivo *testBench* que lleva por extensión **.vhd* y automáticamente este le entregara a modo de descarga su mismo archivo de prueba pero con las líneas adicionales requeridas.

Para generar el archivo **.tit* se ha establecido un formato general de las líneas adicionales que necesariamente deben ir en el *testBench.* En la Figura 3.25 se presenta este formato, donde las únicas variables presentes son los nombres de las entradas de usuario, el resto del contenido de este *process* de *vhdl* debe ir siempre sin importar el diseño a probar.

```
Monitor:process(NombreIn_0,NombreIn_1, .... ,NombreIn_N)
file outfile: TEXT is out "simulacion.tit";
variable outline: LINE;

begin

write(outline, String'(" NombreIn_0:"));
write(outline,NombreIn_0);
write(outline, String'(" NombreIn_1:"));
write(outline,NombreIn_1);
         .
         .
         .
write(outline, String'(" NombreIn_N:"));
write(outline,NombreIn_N);
   write(outline, String'(" tinf:"));
   write(outline, now);
   writeline(outfile,outline);
end process;
```

Figura 3.25 Formato de las líneas que el usuario debe ingresar en el testbench

Lo que los usuarios deben hacer si adicionan por sí mismos este código del *process* en el *testBench*, es realizar la simulación de su diseño para que el archivo que se ve nombrado en el código de la Figura 3.25 como *simulacion.tit* pueda ser creado. Una vez este archivo ha sido creado tras la simulación, puede cargase directamente en la aplicación para seguir con el proceso; sin embargo, si el usuario prefiere cargar el archivo *testBench* para evitar adicionar el código bien sea por desconocimiento o porque prefiere que se haga de forma automática, debe descargar el archivo que se le entrega (ver *Figura 6.20*), adicionarlo como su *testBench* en el *ISE Design* y realizar nuevamente la simulación para poder crear el archivo ***simulacion.tit*** y avanzar en el proceso. El formato del archivo *simulacion.tit* se presenta en la *Figura 6.21*.

El diagrama de flujo de la Figura 3.26 presenta las opciones que el usuario puede tomar dependiendo del archivo que desee subir a la aplicación, también las consecuencias de cada acción tomada durante este proceso.

Con el archivo para la generación de estímulos cargado en la aplicación, se continúa con la configuración del analizador lógico. La página entrega al usuario seis parámetros que no deben ser completados en su

totalidad, ya que dependiendo del tipo de *trigger* que se seleccione se habilitarán los parámetros concernientes a dicha selección.

Por ejemplo en la Figura 3.27 se observan los tres casos posibles de configuración. En *a)* se observa que se ha seleccionado el "Tipo de *trigger*" como "*continuo*" y que únicamente se han habilitado los campos necesarios para este tipo de *trigger* que son el "*número de bits*" y la "*Frecuencia de muestreo*"; estos dos parámetros en realidad son necesarios en cualquier tipo de configuración. Por otra parte, si la selección del *trigger* es pre/post, de igual forma se seleccionan el "número de bits" y la "Frecuencia de muestreo", pero adicionalmente se debe elegir el "modo de Disparo". Si la selección es "Flanco", como se observa en *b)*, se debe también poner el "Número del Canal" cuyo valor está condicionado por el "número de bits"; es decir, si se seleccionan 16 bits, el número de canal debe tener un valor entre 0 y 15, de lo contrario se genera la advertencia correspondiente. Si la selección del "modo de Disparo" es por "Nivel", entonces se debe seleccionar la "Palabra de Disparo" como se puede apreciar en *c)*, cuyo valor está condicionado por el número de bits; si por ejemplo se han seleccionado 8 bits, la palabra de disparo debe tener un valor entre 0 y 255(0 y $2^{\#\text{ de bits}}$-1). Por último, la configuración del analizador lógico requiere la especificación de la frecuencia de muestreo a la que serían capturados de datos.

Finalmente, se tiene un campo que indica el tipo de configuración de la FPGA (ver *Figura 6.22*) y plantea dos opciones, la primera es modo *DIRECTO* y la segunda es *MEMORIA FLASH*, estas opciones se refieren al destino del archivo de configuración de la FPGA en el *hardware*. Si la opción es la primera, entonces el *bitstream* será llevado directamente a la FPGA, pero si se selecciona "Memoria Flash" será almacenado en una memoria tipo *flash*, que se encuentra disponible en la tarjeta de experimentación. Esto con el fin de posibilitar un diseño de prueba para que los estudiantes solo realicen análisis de resultados.

Todo lo mencionado hasta este punto de la etapa de configuración del analizador lógico y de la generación de estímulos es el modo funcional; sin embargo, ahora se hace un análisis de la etapa de diseño de este proceso.

Para lograr la funcionalidad de esta etapa del laboratorio se requiere de la unión de varios elementos, entre los que se encuentran *scripts* en *html*, *JavaScript* y/o *php*. Con ellos se construye una petición *POST* para enviarla a un *Servlet* de *Java* en el que se procesa la información recibida y se entrega una respuesta.

Se emplea también una petición *POST* desde un formulario que involucra todos los parámetros de configuración del analizador lógico que se le presentan al usuario: el campo para la subida del archivo, los campos de *Tipo de trigger*, *Número de bits*, el tipo de *Disparo*, los valores de *Número de Canal* y *Palabra de Disparo*, la *Frecuencia de Muestreo* y el modo de programación del FPGA (*Prog Fpga)*. Una vez que el usuario ha ingresado los valores deseados en estos campos, éste debe presionar el botón "*ACEPTAR"* con lo que se envía la petición. Se utiliza un *script* en *php* para que en la *POST* se envíe el nombre del usuario actual, que se obtiene mediante una consulta a la base de datos, y también se usa un código en *JavaScript* para generar diversas alertas y poder guiar de forma adecuada al estudiante en la configuración del sistema (ver Figura 6.23).

Cuando se lanza la petición, ésta es atendida por el *Servlet AlTestBench.java*, cuyo diagrama de clases se presenta en la Figura 3.28. La función de este *Servlet* es recuperar los parámetros que están encapsulados en la petición *POST*. Estos parámetros se componen de un archivo y siete variables; pero su extracción desde la petición se hace a través de la clase *MultipartRequest*, que permite también pasar estos valores a variables en *Java*. Con ellos se crea el archivo "*analizador.txt"* que tiene siempre la misma interpretación, dado el orden fijo que posee cada parámetro en el archivo, como se presenta en la Figura 3.29. El tratamiento que se le da al archivo subido por el usuario es diferente.

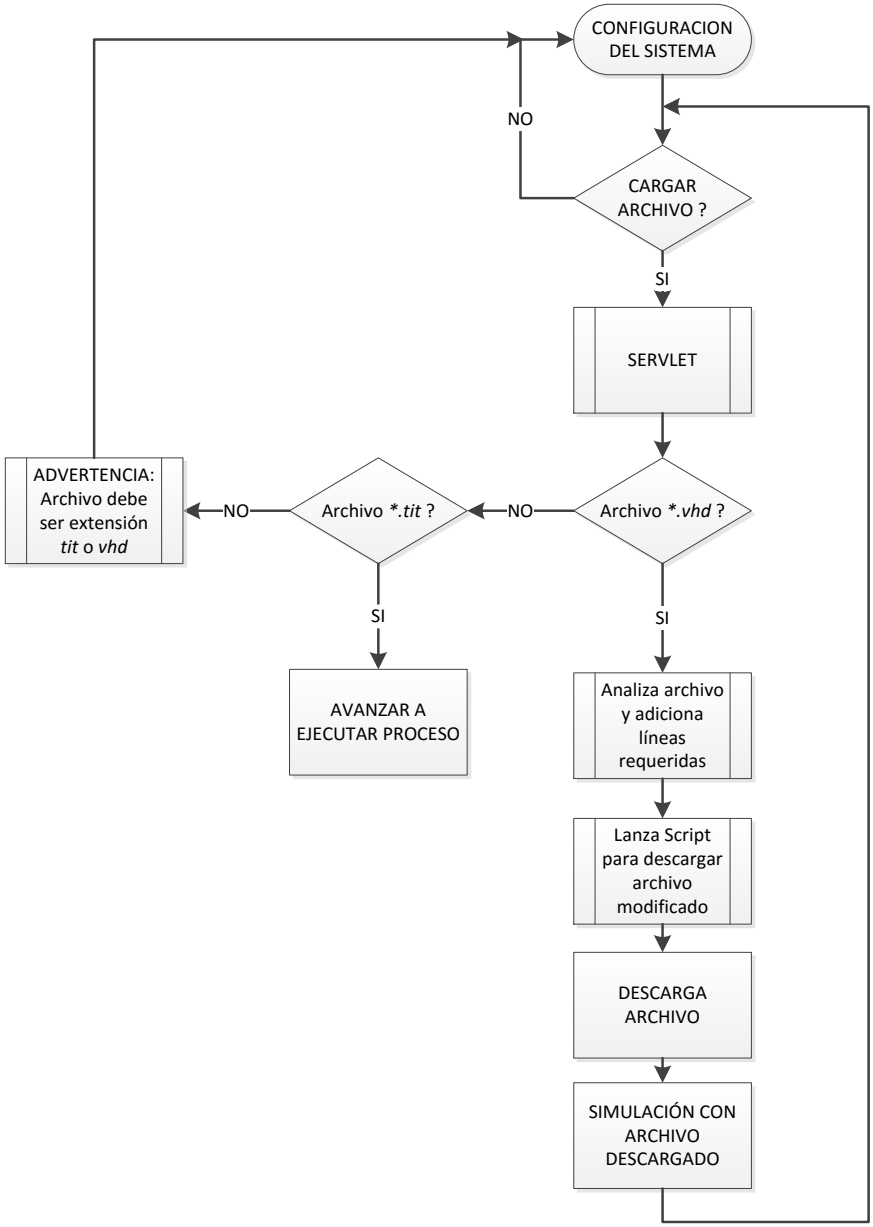


Figura 3.26 Diagrama de flujo del proceso de cargar el archivo para generación de estímulos

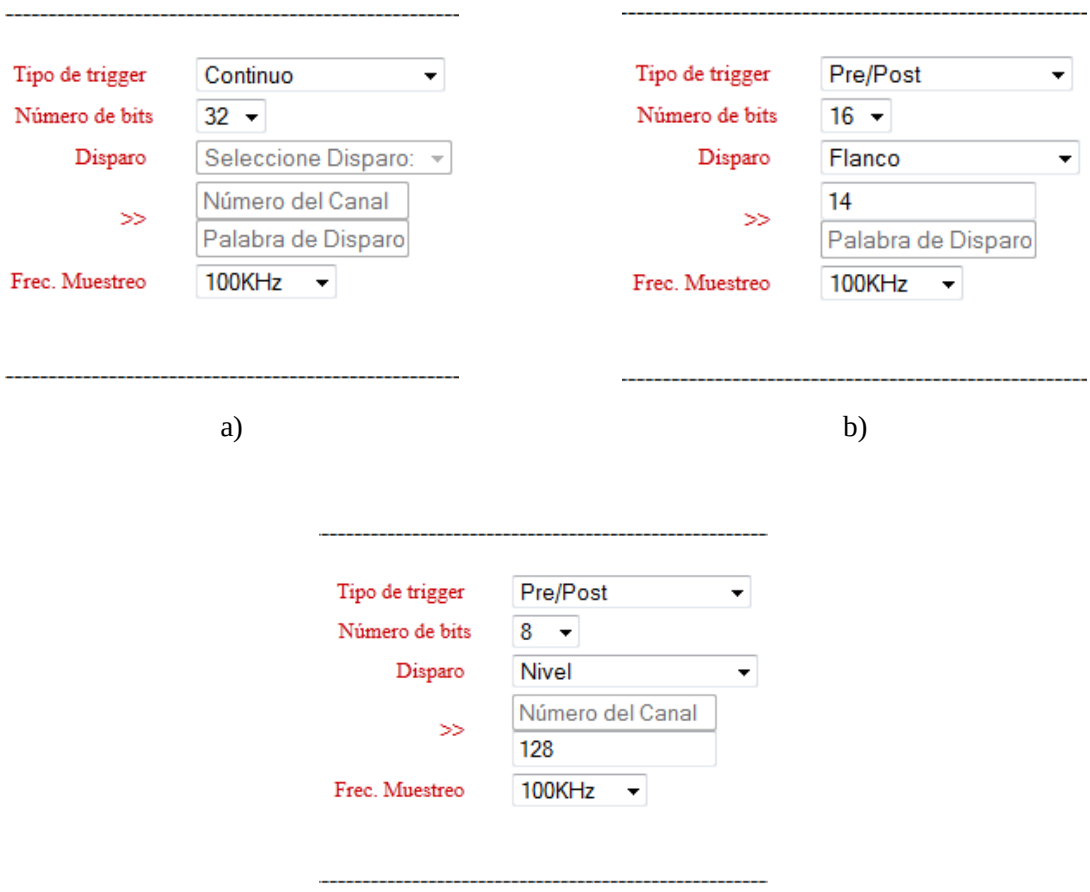


Figura 3.27 Parámetros de configuración del analizador lógico

Inicialmente el *Servlet* verifica la extensión del archivo subido para saber qué acciones tomar. Si se detecta que la extensión es **.vhd*, quiere decir que se deben adicionar las líneas presentadas en la Figura 3.25; para esto, se analiza el archivo para reconocer los nombres de las señales de entrada e ir escribiendo línea a línea el formato establecido. Una vez que el archivo se ha completado, se copia en una carpeta del servidor y posteriormente desde el *Servlet* se lanza un *script* en *php,* con el que se le presenta al usuario la ventana de descarga con el archivo modificado. Con él, debe realizarse nuevamente la simulación del diseño para crear el archivo requerido (**.tit*); sin embargo, si la extensión del archivo subido es **.tit* cuyo formato es el mostrado en la *Figura 6.21*y en la que se observa que existe una columna que da información del tiempo de las transiciones de cada señal; este tiempo siempre se da en *nanosegundos*. Este archivo se procesa de tal forma que, haciendo caso de la columna de tiempo se genere un bit por cada *nanosegundo* como se observa en la *Figura 6.24*. Finalmente, este archivo se guarda en el servidor con el nombre del usuario actual, para lograr diferenciarlo posteriormente, y se le designa como extensión **.gdm.* Esta extensión es creada para ser reconocida solo al interior del proceso del sistema, lo que en el *Servlet* posibilita tener manejo de archivos como escritura, lectura, copiado entre otras, gracias a las clases ***FileAccess.java*** y ***File.java*** que están instanciadas en el mismo.

Finalizado el proceso de este *Servlet*, se entrega una respuesta *html* de redirección hacia el siguiente nivel de la aplicación en el que se establece comunicación con el *hardware* con todos los archivos creados.

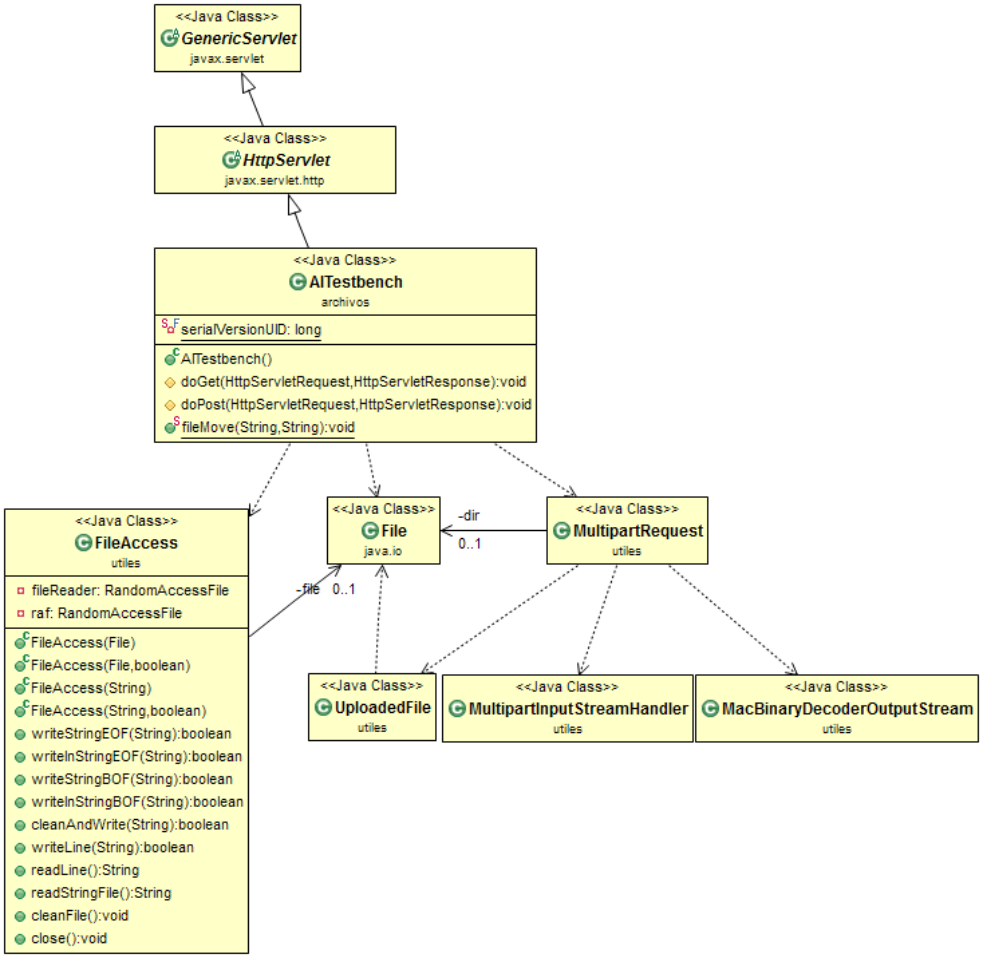


Figura 3.28 Diagrama de clases de la herramienta que permite modificar el archivo subido por el usuario o procesar directamente el archivo de extensión tit.

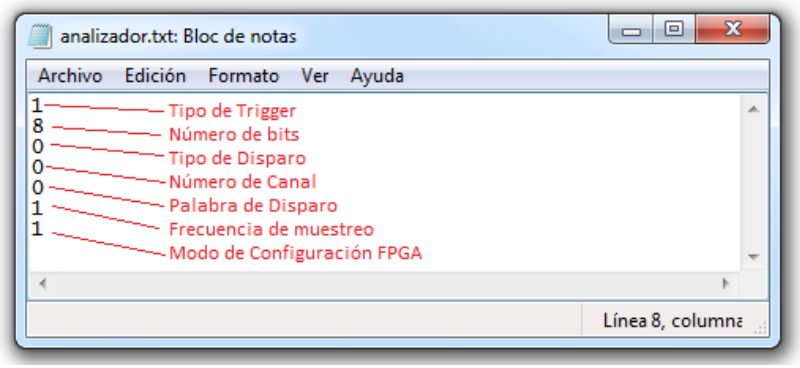


Figura 3.29 Formato del archivo creado a partir de los parámetros de configuración de usuario

3.4.8.2.1 Algunos parámetros adicionales

La especificación de la aplicación flexibiliza la adición y/o sustracción de parámetros y funciones de operación sobre el recurso de la placa de experimentación. En particular, si el analizador lógico soporta además tipo de muestreo transicional y permite que el usuario ajuste el porcentaje de memoria de muestreo para usar post-disparo, los ajustes convergen en un ajuste en el tamaño de la trama que se empaqueta desde la aplicación de *java*.

En la Figura 3.30se muestra el proceso general de selección de parámetros de configuración del analizador lógico desde la interfaz Web. El bloque de "Ajuste de Puntero" establece el cálculo de posiciones discretas de memoria, para ajustar el sistema de direccionamiento a memoria de muestreo del analizador lógico, con base en el porcentaje de memoria que el usuario defina para el post-disparo.

Por otro lado, el orden de los parámetros enviados al microcontrolador los definen el habilitador de disparo, el tipo de disparo y el número de bytes, en el caso de disparo por nivel. Es importante tener en cuenta los valores asignados para cada una de las selecciones, con el fin de establecer la decodificación en el microcontrolador. El diagrama de la Figura 3.31 muestra el proceso de empaquetamiento de los datos de configuración en una trama de máximo 8 campos.

3.4.8.3 Ejecución del Proceso

Para poder ejecutar el proceso de programación del analizador lógico, de la FPGA y el envío de estímulos (ver *Figura 6.25*), es requisito indispensable que el sistema tenga a disposición los archivos necesarios *(*.bit, *.gdm, *.ucf, *.txt, *.vhd, *.tit*) ya que es a partir de éstos que se extraen los datos para enviar al *hardware*.

La arquitectura seleccionada para el laboratorio remoto plantea dos servidores que se comunican entre sí. Uno es el encargado del manejo de usuarios, base de datos, alojamiento del sitio Web y de contener y ejecutar todos los *Servlets* mencionados hasta este punto; sin embargo, este servidor no posee comunicación directa con el *hardware*, por lo que se requiere una forma de hacer llegar estos archivos al servidor de control. Este último es capaz de comunicarse con la tarjeta de entrenamiento, vía USB, en la que un microcontrolador gestiona el acceso a los recursos.

Existe un *Servlet* encargado de ubicar los archivos de extensión **.bit*, **.ucf*, **.gdm* y **.txt* para ser transferidos vía *FTP* al segundo servidor (ver Figura 3.32). La conexión *FTP* entre ambos servidores es soportada por la red, pero se hace efectiva mediante un software especializado en servicios de transferencia de archivos. En esta aplicación se ha seleccionado el *FileZilla Server*, que es un servidor FTP multiplataforma y de código abierto.

Como se observa en el diagrama de clases de la Figura 3.32, el *Servlet* incluye un "<<*JavaComponent*>>" llamado *edtftpj.jar*. Este paquete permite que desde *Java* se pueda acceder a un servidor de *FTP*, por lo que desde este *Servlet* se capturan los archivos a enviar, se establece la conexión con el servidor *FTP* del servidor de *Hardware*, mediante *login* y *password* previamente asignados. Cuando se ha establecido correctamente la conexión, los archivos son enviados al servidor; sin embargo, si se presenta algún error, la aplicación informará del mismo al usuario (ver *Figura 6.26*).

Cuando los archivos se han enviado, el *Servlet* finaliza haciendo un llamado a otro *Servlet* residente en el segundo servidor; esto es, el PC que tiene conexión directa con el *hardware*. Cuando se hace el llamado de este nuevo *Servlet*, se inicia la interacción con la tarjeta de experimentación.

El *Servlet* que interactúa con el hardware es *AtiendePeticion.java* cuyo diagrama de clases se presenta en la Figura 3.33. La principal función de este *Servlet* es garantizar el protocolo de comunicación, usando el

paquete *Protocolo.jar*. No obstante, esto requiere de determinados eventos para su ejecución, además del uso de las clases de *Java FileAccess*, *File*, *PrintWriter* para el manejo de archivos. Otra función que resulta de utilidad para el administrador y que ha sido adicionada en este *Servlet* es la implementación de un archivo que lleva el registro de cada proceso ejecutado por cada usuario (ver Figura 3.34).

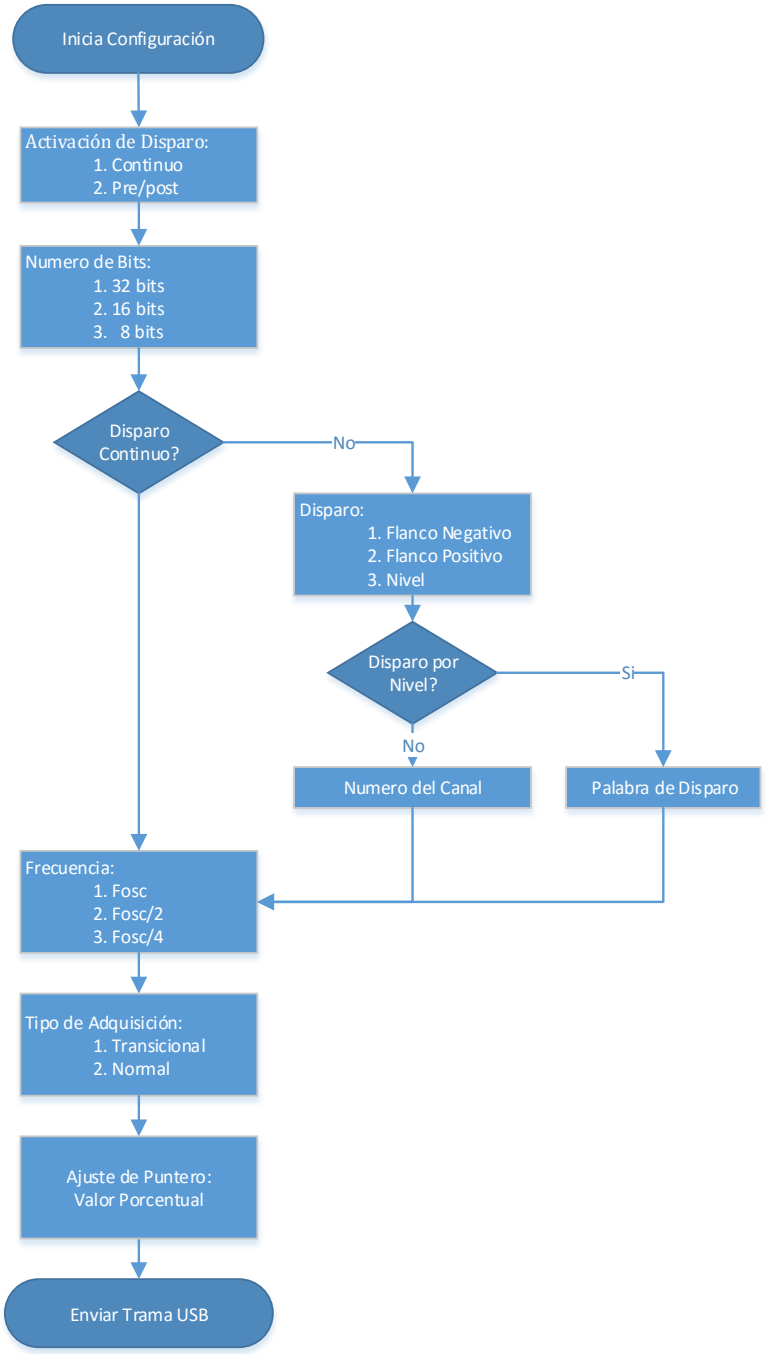


Figura 3.30 Diagrama de flujo para el ingreso de características desde la plataforma Web

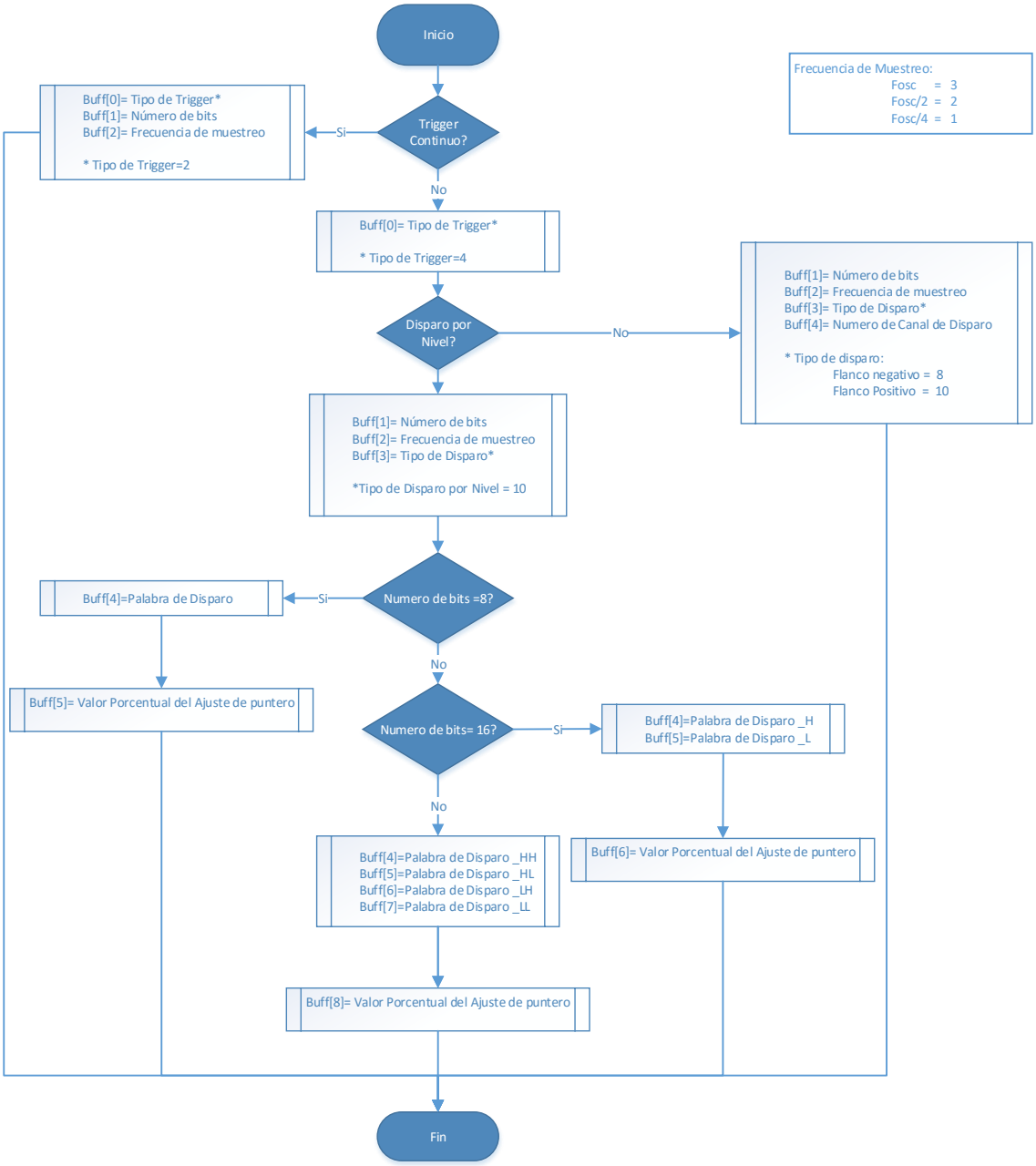


Figura 3.31 Proceso del aplicativo en *java* para el envío de la configuración

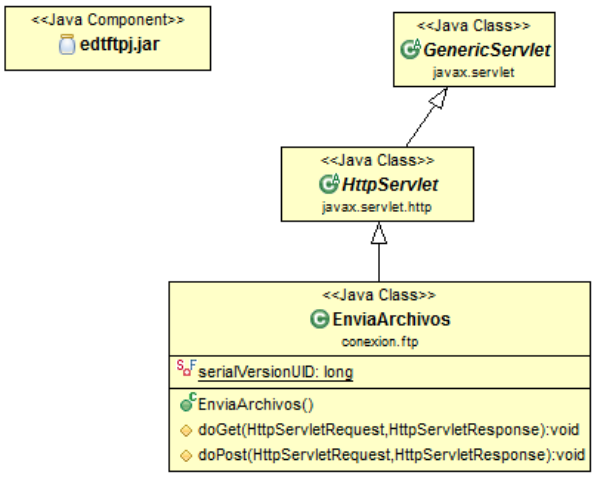


Figura 3.32 Diagrama de clases de la herramienta que permite enviar archivos vía FTP hacia el servidor de *hardware*.

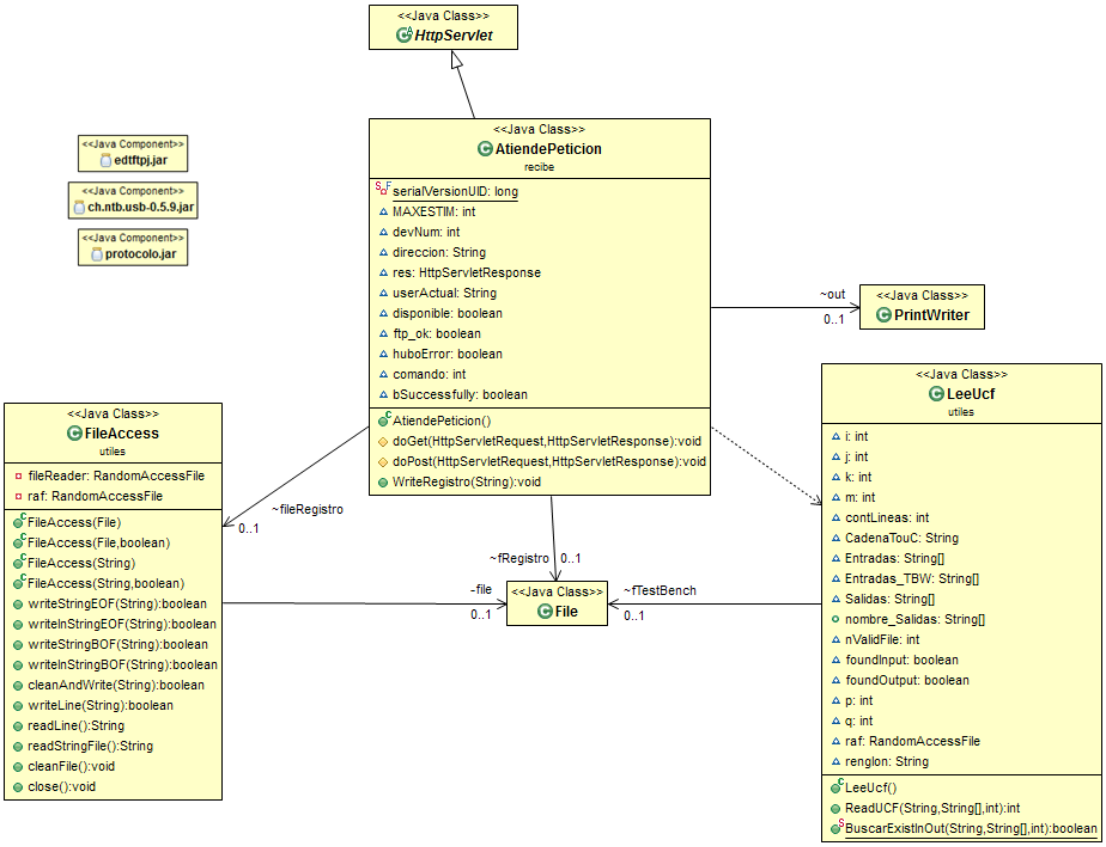


Figura 3.33 Diagrama de clases de la herramienta que permite establecer comunicación con el hardware.

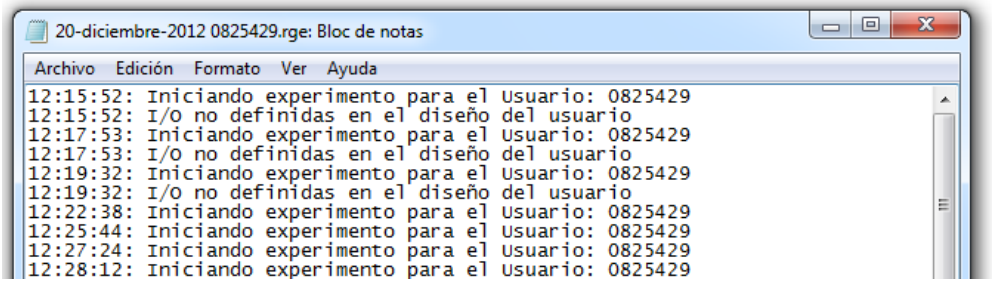


Figura 3.34 Archivo que lleva el registro de las acciones realizadas en el laboratorio

Los archivos transferidos al servidor de *hardware* se abren y, si no se presenta algún error, se comprueba el estado de disponibilidad del *hardware*. Esto consiste en establecer si la tarjeta de experimentación está o no conectada al PC servidor de control; en caso de no encontrarse disponible, se informa al usuario a través del sitio Web, donde existe además un espacio para dejar comentario y que le posibilita a un administrador hacer un seguimiento de posibles fallas del sistema (ver *Figura 6.27*). Si el *hardware* está disponible para su uso, se inicia un proceso automático de cuatro pasos que se describen a continuación.

El primer paso que ejecuta el *Servlet* es programar el FPGA, esto se hace abriendo desde *Java* el *bitstream* de extensión **.bit* y pasándolo a un *buffer* desde donde es leído y enviado por campos de 64 Bytes, ya que este es el máximo tamaño que se dispone en el Microcontrolador para la comunicación USB. En la Figura 3.35 se explica mediante un diagrama de flujo el proceso del envío y recepción de datos. Para esto, primero se debe solicitar la interfaz del dispositivo (*uC*) para verificar que este no se encuentra ocupado en otro proceso de escritura o lectura; el sistema intenta tres veces la solicitud de la interfaz antes de enviar un mensaje de que no es posible llevar a cabo la programación de la FPGA, pero una vez se le concede la interfaz se puede iniciar el proceso de envío del *bitstream*. Si existe un error, el proceso se repite igualmente tres veces antes de avisar que se produjo un error. Si la transferencia finaliza con éxito, entonces se libera la interfaz para que quede disponible para el segundo paso. Cabe mencionar que el *bitstream* se envía al *uC*, pero éste no sabe a "quién" redireccionar los datos recibidos, por lo que se establecieron una serie de comandos con los que se le informa al administrador de tareas qué proceso ejecutar dependiendo del comando recibido (ver Figura 3.36). Cada comando se envía justo antes de enviar el archivo a programar.

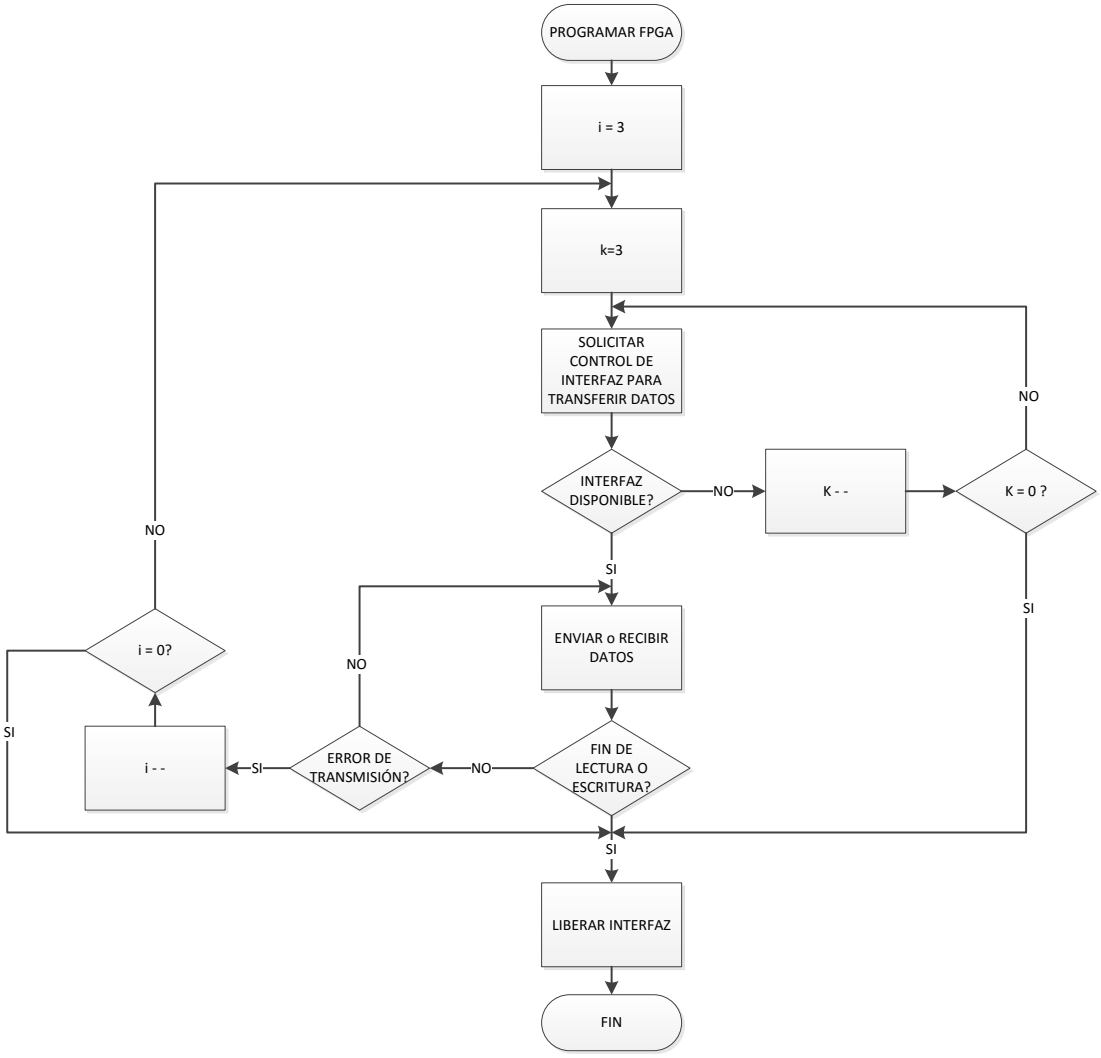


Figura 3.35 Diagrama de flujo del proceso de envío y recepción de datos con el *hardware*

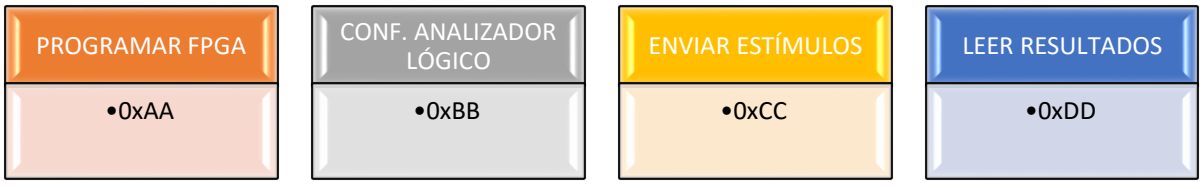


Figura 3.36 Comandos implementados para reconocer cada proceso a ejecutar

Con los comandos establecidos y el archivo de configuración de la FPGA almacenado en un *buffer*, se envía el archivo en tramas de la forma que se presenta en la Figura 3.37.

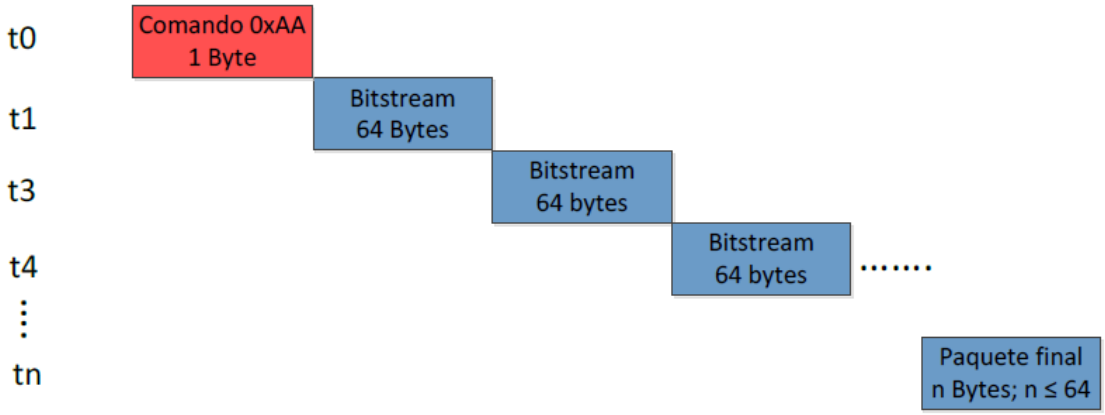


Figura 3.37 Tramas enviadas del archivo de configuración de la FPGA hacia el *hardware*

Antes de ejecutar el segundo paso del *Servlet*, se debe destacar que existen tiempos cortos de espera entre la ejecución de cada paso, para garantizar la total liberación de la interfaz y evitar errores. Este segundo paso corresponde a la configuración del bloque analizador lógico, para lo que el *Servlet* abre el archivo ***analizador.txt***, que contiene los parámetros para esta configuración. Estos parámetros se almacenan en un *buffer* para su posterior envío USB, previa solicitud de la interfaz (ver Figura 3.35).

Las tramas correspondientes a la configuración del analizador lógico, que se envían desde la aplicación hacia el *hardware*, se presentan en la Figura 3.38. Se puede apreciar que, dependiendo del número de bits que el usuario seleccione, las tramas varían su tamaño, dado que la palabra de disparo depende directamente de los bits o canales configurados.

El tercer paso corresponde al envío de estímulos; para este caso, el *Servlet* abre el archivo de extensión **.gdm* correspondiente al usuario actual. Para crear los estímulos, debe considerarse que por cada columna de 8 bits se crea su equivalente hexadecimal y cada uno de los hexadecimales creados se almacena en un *buffer*, que tiene tantas posiciones como estímulos. En la Figura 3.39se observa la organización del archivo y cómo se forma cada estímulo.

Una vez que se han almacenado en el buffer todos los estímulos se procede al igual que en los pasos anteriores con la solicitud de la interfaz para que una vez cedida, se inicie con la transferencia de los estímulos hacia el microcontrolador. Las tramas que describen el envío de estímulos se presentan en la Figura 3.40.

El último paso que este *Servlet* ejecuta es la captura de datos; por lo tanto, primero se envía el código para que el microcontrolador interprete la acción a realizar. Esta acción consiste en leer los datos de una memoria correspondientes a la captura de los resultados producto de los estímulos enviados. El microcontrolador tiene capacidad de enviar paquetes de 64 Bytes y el canal para análisis es de 4 bytes, pero para enviar datos por USB se deben fraccionar estos 4 bytes en datos de 1 byte (ver Figura 3.41); por lo tanto, la aplicación debe reconstruir paquetes de datos al recibirlos para obtener el valor real capturado en el hardware, los valores que se construyen en el *Servlet* están dados en hexadecimales. Estos valores se transforman a binario de 32 bits para escribirlos en un archivo para el resultado.

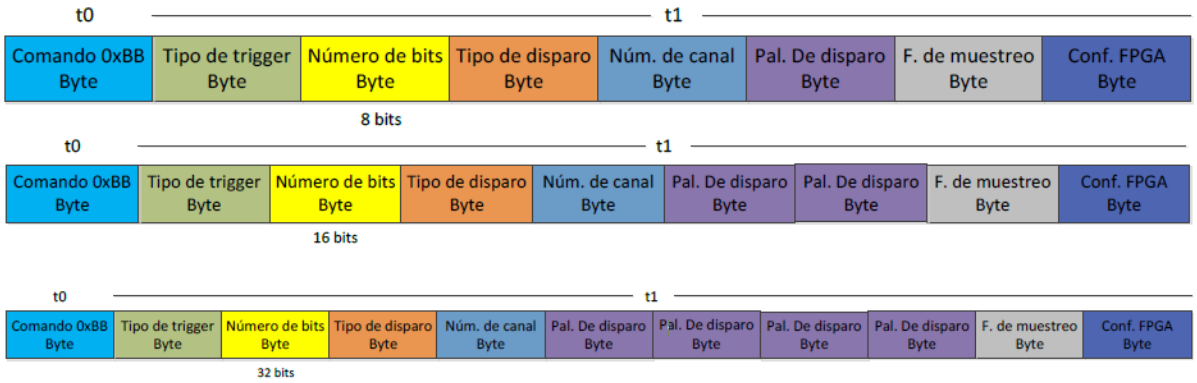


Figura 3.38 Tramas enviadas del archivo de configuración del analizador lógico hacia el hardware

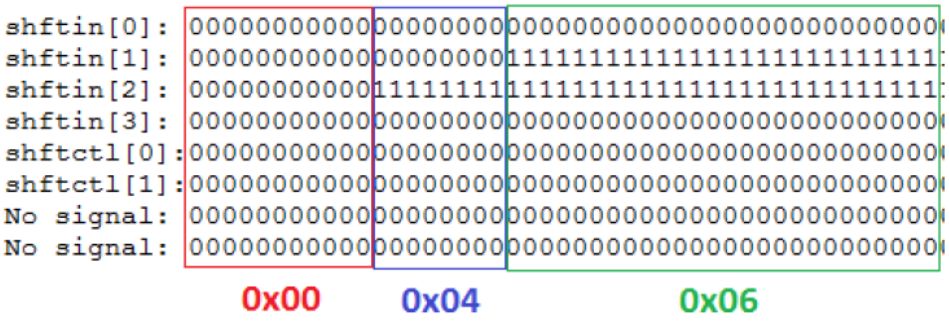


Figura 3.39 Interpretación del archivo *testbench* para la generación de estímulos

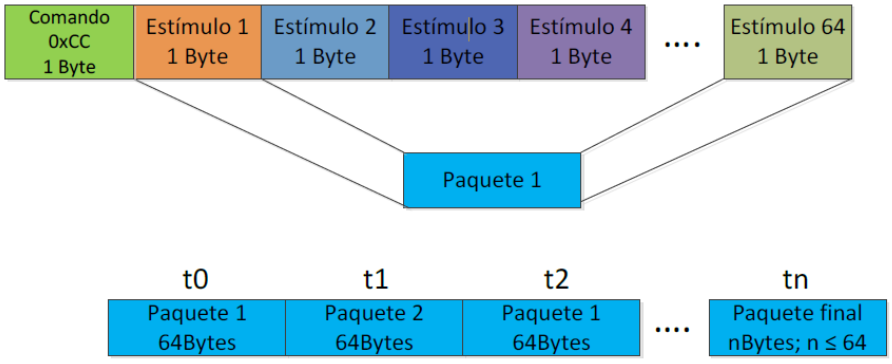


Figura 3.40 Tramas enviadas para el envío de estímulos hacia el *hardware*

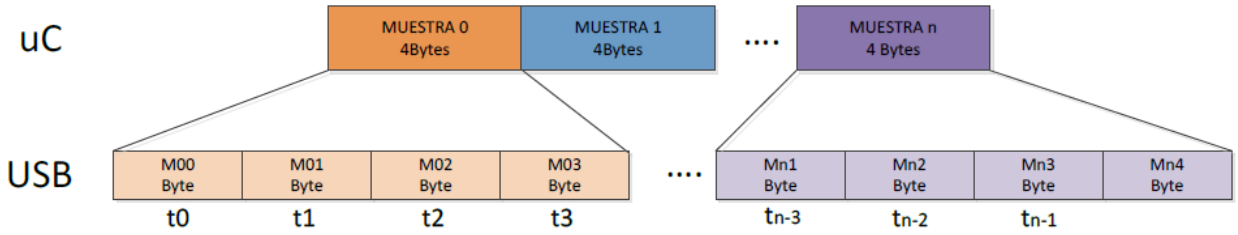


Figura 3.41 Tramas enviadas desde el uC hacia la aplicación

El archivo de resultado se forma a partir del archivo utilizado para la generación de estímulos (de extensión **.gdm*). Una vez se obtienen los resultados, se adicionan al archivo mencionado. En la Figura 3.42se observa el archivo presentado previamente en la Figura 3.39 cuya diferencia con este nuevo archivo radica en la adición de los resultados obtenidos. También se observa que se utilizan los mismos nombres de las salidas proporcionadas por el usuario y las salidas que no se utilicen se nombran como "*No usada*"; este proceso se realiza gracias a la inclusión de la Clase *LeeUcf*.

Una vez el *Servlet* finaliza el cuarto paso, se procede con la transferencia vía *FTP* del archivo resultado que contiene los valores capturados por el *Hardware*; por esta razón, en el diagrama de cclases se usa el paquete ***edtftpj.jar*** para establecer comunicación con el servidor *FTP* del servidor Web. Una vez que el archivo se encuentra en el servidor Web, la última acción del *Servlet* es redireccionar al usuario a una nueva página con la que se le informa de la finalización exitosa del proceso (ver Figura 6.28); sin embargo, durante todo el proceso se verifica cada uno de los pasos para descartar errores e informar al usuario.

entradas

```
shftin[0]: 00000000000000001111111111111111111111111111111111111111111111111
shftin[1]: 00000000011111111111111111111111111111111111111111111111111111111
shftin[2]: 00000000000000000000000000000000000000000000000000000000000000000
shftin[3]: 00000000000000000000000000000000000000000000000000000000000000000
shftctl[0]:00000000000000000000000000000000000000000000000000000000000000000
shftctl[1]:00000000000000000000000000000000000000000000000000000000000000000
No signal: 00000000000000000000000000000000000000000000000000000000000000000
No signal: 00000000000000000000000000000000000000000000000000000000000000000
shftout[0]:00000000000000001111111111111111000000000000000011111111111111111
shftout[1]:00000000000000000000000000000011111111111111111111111111111111111
shftout[2]:00000000000000000000000000000000000000000000000000000000000000000
shftout[3]:00000000000000000000000000000000000000000000000000000000000000000
No usada:  00000000000000000000000000000000000000000000000000000000000000000
No usada:  00000000000000000000000000000000000000000000000000000000000000000
No usada:  00000000000000000000000000000000000000000000000000000000000000000
No usada:  00000000000000000000000000000000000000000000000000000000000000000
No usada:  00000000000000000000000000000000000000000000000000000000000000000
No usada:  00000000000000000000000000000000000000000000000000000000000000000
No usada:  00000000000000000000000000000000000000000000000000000000000000000
No usada:  00000000000000000000000000000000000000000000000000000000000000000
No usada:  00000000000000000000000000000000000000000000000000000000000000000
No usada:  00000000000000000000000000000000000000000000000000000000000000000
No usada:  00000000000000000000000000000000000000000000000000000000000000000
No usada:  00000000000000000000000000000000000000000000000000000000000000000
No usada:  00000000000000000000000000000000000000000000000000000000000000000
No usada:  00000000000000000000000000000000000000000000000000000000000000000
No usada:  00000000000000000000000000000000000000000000000000000000000000000
No usada:  00000000000000000000000000000000000000000000000000000000000000000
No usada:  00000000000000000000000000000000000000000000000000000000000000000
No usada:  00000000000000000000000000000000000000000000000000000000000000000
```

salidas

Figura 3.42 Construcción del archivo de resultados

3.4.8.4 Herramienta de Visualización

Una vez el usuario ha llegado a la página de la Figura 6.28, el proceso se traslada al servidor principal o servidor Web para realizar el último proceso, que corresponde a la visualización de resultados. En este último paso, al igual que en las etapas de configuración se ha involucrado un *script* en *html* para crear un formulario que haga el llamado al *Servlet* ***ArmaApplet.java*** inmediatamente después de que el usuario presione el botón "*VISUALIZAR RESULTADOS*".

Este proceso final, involucra un *Applet* de *Java* que se construye a partir de una plantilla diseñada con *html* en la que se presentan campos editables para los nombres de los canales y los valores correspondientes, tanto a las entradas como las salidas y otros campos requeridos para el funcionamiento de la aplicación. El encargado de llenar los campos de esta plantilla es el *Servlet ArmaApplet.java* cuyo diagrama de clases se presenta en la Figura 3.43. Este *Servlet* abre el archivo de resultados que se le envío vía *FTP* desde el servidor de *hardware*, extrae los nombres de las entradas y de las salidas, así como los valores correspondientes a éstas. A través de las Clases *File*, *Plantilla* y *Bloque* se abre el archivo plantilla, se completan los campos que se encuentran en las etiquetas *"<!—var:nombre_variable>"* con los nombres de los 32 canales, los valores de cada canal, el archivo *graficador.Class* que usa el *Applet* y los demás parámetros.

Una vez la plantilla ha sido completada con los valores de resultado de la experiencia del usuario, el *Servlet* guarda la plantilla con extensión **.htm* en una carpeta del servidor. Al cliente se le envía la página que contiene la plantilla completada y la visualización de la herramienta para graficar las señales de estímulos y los resultados obtenidos en el hardware (ver Figura 6.29). Por otra parte, la construcción del *Applet* es realizado en código *Java* y el diagrama de clases de esta herramienta se presenta en la Figura 6.1.

El *Applet* requiere un cuidadoso proceso de diseño dado la cantidad de señales y opciones que posibilita la herramienta. Esto implica una gran cantidad de código que resulta más cómodo visualizar a través del diagrama de clases. El proceso que lleva a cabo el *Applet* de forma general es obtener los datos que se ingresaron en la plantilla *htm*, se procesan los valores de cada canal mediante la clase *ArchivoBinario*, para cada canal se adiciona un contenedor para los nombres de las señales, se generan también los contenedores requeridos para un canal, para el conjunto de todos los canales y el *frame* que contiene todo el *Applet*. Adicionalmente, la herramienta ofrece cuatro funciones para mejorar la visualización de los resultados o para su análisis, estas son hacer *zoom* bien sea de acercamiento o alejamiento, también existen dos tipos de cursores; el primero permite ver en cada *slot* de tiempo el estado de cada señal y el segundo es un par de cursores diferenciales que sirven para medir el tiempo existente entre un instante de una señal determinada y otro momento bien sea adelante o atrás del otro cursor.

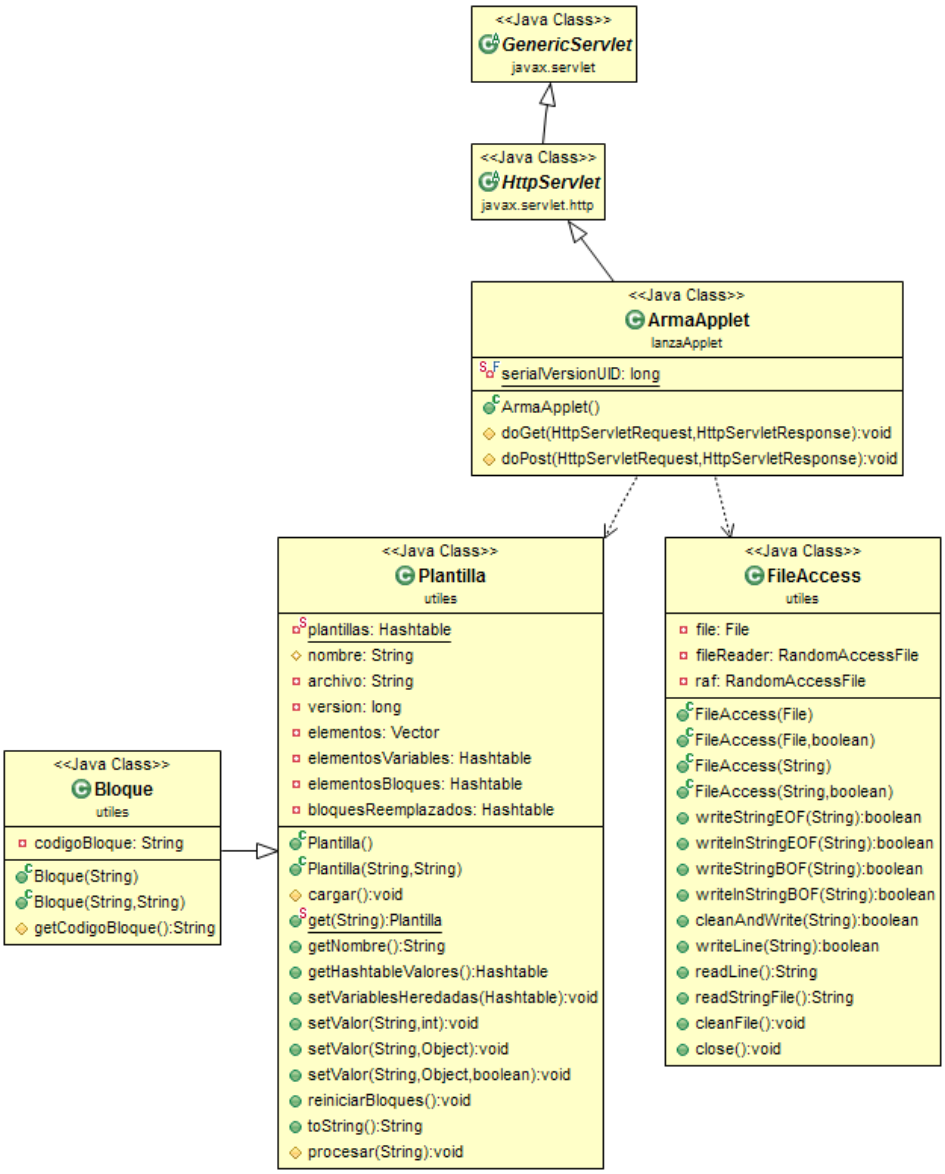


Figura 3.43 Diagrama de clases de la herramienta que permite editar la plantilla para construir el *applet*

4. DESARROLLO DE LA COMPONENTE HARDWARE DEL SISTEMA INFORMÁTICO

Las metodologías basadas en *e-learning* para el diseño de circuitos electrónicos deben integrar aspectos experienciales que lo encaminen hacia un verdadero aprendizaje significativo. Sin embargo, vincular la experiencia en un modelo presencial en este contexto implica el uso de recursos de instrumentación, que generalmente tienen un coste elevado, para implementar laboratorios de experimentación que soporten la programación, prueba y medición, como componentes esenciales en la enseñanza de circuitos digitales. Los analizadores lógicos también apoyan estos procedimientos, ya que se identifican como instrumentos para la validación y depuración de sistemas digitales. Además, se utilizan tarjetas de desarrollo comerciales basadas en arquitecturas digitales reprogramables para el entrenamiento adecuado, ya que facilitan la implementación y depuración de diseños digitales utilizando una herramienta de programación basada en PC. Por lo general, esta herramienta se utiliza para programar los procesadores embebidos, analizadores lógicos y otros núcleos específicos sobre una matriz de puertas programable (FPGA).

Para un aprendizaje activo semipresencial, las garantías de acceso a estos recursos son bajas y generalmente condicionadas al porcentaje de tiempo de asistencia obligatoria a los laboratorios físicos donde residen los elementos para la realización de prácticas. En el capítulo 3 se presentan los detalles del diseño de una aplicación basada en Web, que permite la interacción entre usuarios con el objetivo de programar y probar funciones lógicas digitales, de acuerdo con las prácticas de laboratorio programadas, sobre una plataforma *hardware* configurable de acceso remoto. La integración de redes sociales y otros servicios Web 2.0 en esta herramienta, garantiza un recurso que favorece la implementación de didácticas de aprendizaje activo semipresencial que impliquen también el trabajo en equipo.

En este capítulo se describe el proceso de diseño de la placa de experimentación que constituye el recurso de acceso remoto de la arquitectura propuesta en la sección 2.4.2. El hardware propuesto incorpora un analizador lógico y un FPGA como módulos independientes para asegurar que el principal recurso reprogramable por el usuario está totalmente disponible. Este capítulo presenta los detalles del desarrollo de la plataforma *hardware* del laboratorio remoto, describiendo los principales aspectos tecnológicos de los módulos incorporados para el diseño, además de la documentación pertinente para el desarrollo del diseño PCB basado en normas y recomendaciones de fabricantes. Así mismo, se describe la jerarquía del diseño de la tarjeta, el desarrollo del diseño esquemático, del PCB y los requerimientos técnicos para cada uno de los módulos. Teniendo en cuenta que la tarjeta de entrenamiento representa un recurso disponible para múltiples usuarios, de la revisión tecnológica y la disposición de los elementos en el PCB, se presenta también una propuesta para el acceso simultáneo de más de un usuario al uso de la tarjeta. Por último, se presentan las pruebas y los resultados obtenidos al implementar un planificador basado en RTOS, como alternativa de gestión de los recursos disponibles en la arquitectura *hardware*.

4.1 Aproximación Tecnológica

El sistema está formado por cuatro bloques fundamentales que se comunican entre sí para programar un dispositivo lógico configurable, de tipo FPGA, desde un proyecto de usuario remoto y depurarlo a través

de un analizador lógico incrustado en la misma plataforma hardware. El usuario debe contar con un equipo de cómputo en línea, que le permita interactuar con una aplicación Web para enviar los archivos de configuración y hacer uso de los recursos de depuración.

De forma global, se puede considerar que el sistema está formado por cuatro dispositivos que se comunican entre sí para lograr un único objetivo final: que el archivo de configuración de la FPGA generado por el usuario se cargue correctamente en la misma, configurar el modo de funcionamiento del analizador lógico y los datos de la prueba se envíen al servidor para que sea visualizado por el usuario. Los cuatro elementos implicados son los siguientes:

- **Servidor:** En él estará instalado el software encargado de enviar el diseño digital del usuario por el puerto USB hacia la FPGA.
  El servidor de recursos establece los mecanismos y el soporte de servicios para el envío de parámetros de configuración y operación hacia la tarjeta de entrenamiento, la cual se conecta a uno de sus puertos USB. Así mismo, recibe reportes de esta plataforma hardware para presentarlos al usuario que los requiere, de acuerdo con su solicitud. Un servidor Web apoya el trabajo del servidor de recursos y administra las bases de datos para los gestores de contenido y de aprendizaje, así como los permisos de acceso al portal y a la plataforma *hardware*.
- **Microcontrolador:** Se desempeña como el administrador de los recursos disponibles en la tarjeta de entrenamiento del laboratorio, de acuerdo con su secuencia de funcionamiento y los parámetros definidos por cada usuario.
  - **Controlador USB:** Gestiona la comunicación entre el Servidor y FPGA.
  - **Planificador de Tareas:** Realizar la habilitación adecuada de cada uno de los subsistemas en función de las órdenes enviadas desde el Servidor y de la secuencia correcta de funcionamiento de ellos.
- **FPGA(***Field Programmable Gate Array***):** Es el dispositivo de *hardware* reconfigurable donde se carga el diseño realizado por el usuario autorizado.
- **CPLD(***Complex Programmable Logic Device***):** Es el dispositivo de *hardware* configurable donde residen los circuitos del analizador lógico, que se constituye como el instrumento base para la depuración de los proyectos de usuario.

Antes de presentar el desarrollo del proyecto, es necesario describir en detalle las características fundamentales de los bloques centrales que constituyen la esencia del recurso FPGA donde se carga el diseño de los usuarios, el microcontrolador que soportará la comunicación USB entre el servidor y un RTOS que gestiona la habilitación de los recursos y por último, el Analizador lógico que reside en un CPLD. Se expondrán además los rasgos principales del sistema operativo en tiempo real que se utiliza y las recomendaciones pertinentes para el diseño del prototipo realizado.

4.1.1 FPGA

Es un dispositivo lógico configurable donde se pueden programar circuitos digitales sin sustitución o cambios en el *hardware*. Está compuesto por bloques de lógica configurables (CLB – *Configurable Logic Blocks*), bloques de entrada salida (IOB – *Input Output Blocks*), Memoria RAM, DSP, reloj y toda la lógica de interconexión de los elementos funcionales del FPGA (ver Figura 4.1).

Cada CLB tiene un conjunto de celdas lógicas (*Logic Cells - LC*). Un LC contiene un generador de funciones implementado, a su vez, con un *Look-Up Table* (LUT). Además de los cuatro LCs, cada CLB contiene lógica que combina los generadores de funciones para permitir una mayor complejidad.

Los IOBs proporcionan interfaz entre los pines externos y la lógica interna. Cada IOB controla un pin, que puede ser configurado como entrada, salida o en modo bidireccional. Además, cada uno tiene tres registros que comparten la señal de reloj (CLK), pero con distintas señales de capacitación de reloj (CE) [30].

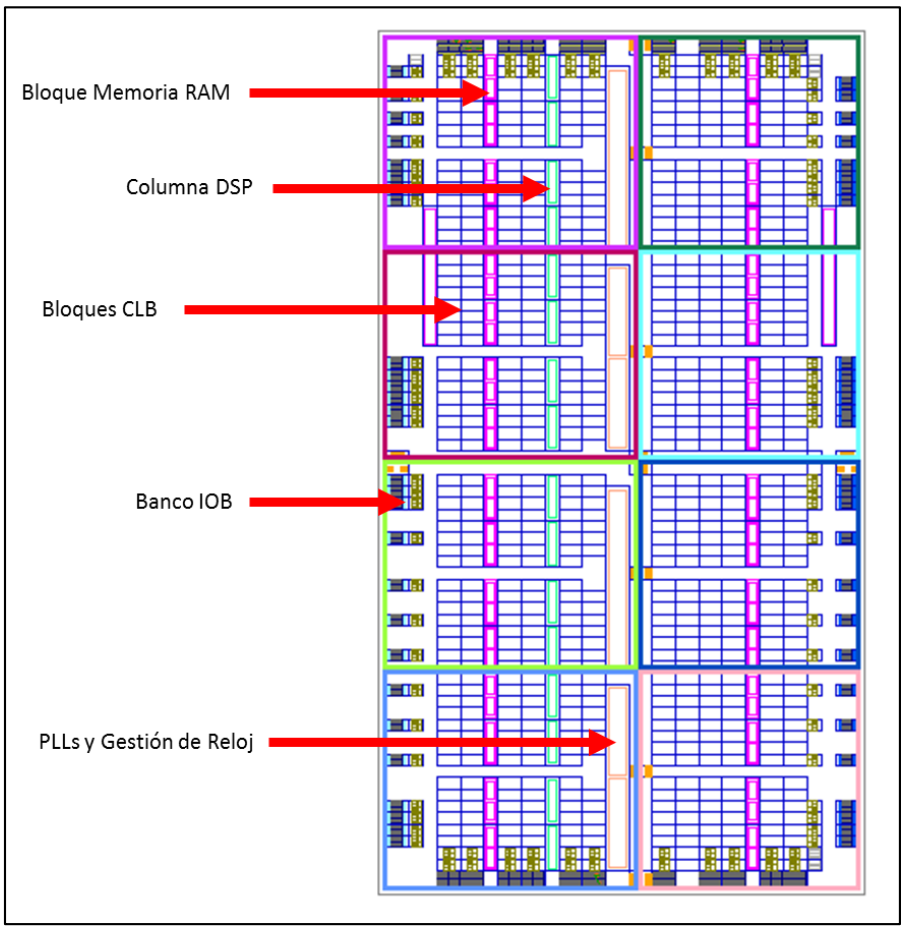


Figura 4.1 Componentes de la Estructura Interna en un FPGA de *Xilinx*

4.1.1.1 FPGA de la Familia Spartan-6

La familia *Spartan®-6* ofrece capacidades líderes de integración de sistemas de bajo costo para aplicaciones de alto volumen. Esta familia proporciona densidades expandidas que van desde 3.840 a 147.443 celdas lógicas, con la mitad del consumo de energía de las familias anteriores, más rápida, y con una conectividad más completa. La familia *Spartan- 6* ofrece un registro doble de 6 entradas a la lógica en una LUT. Además, incluyen bloques RAM de 18 Kb (2 x 9 Kb), secciones DSP48A1 de segunda generación, entre otras características que proporcionan una alternativa programable de bajo costo para productos ASIC personalizados con facilidad de uso. Los FPGA de esta familia ofrecen la mejor solución para los diseños de grandes volúmenes lógicos, diseños DSP orientados al consumidor y las aplicaciones integradas de presupuesto limitado. Así mismo, son una base de silicio programable para plataformas de diseño dirigidos que entregan *software* integrado y componentes de *hardware* que permiten a los diseñadores centrarse en la innovación, desde el inicio de su ciclo de desarrollo.

4.1.1.1.1 CLBs, Slices y LUTs

Cada bloque de lógica configurable (CLB) en una Spartan-6 consta de dos "*Slices*", dispuestas lado a lado como parte de dos columnas verticales. Hay tres tipos de secciones CLB (Ver Figura 4.2) en la arquitectura

Spartan-6: *SLICEM*, *SLICEL* y *Slicex*. Cada uno contiene cuatro LUTs, ocho *flip-flops*, y lógica variada. Las LUTs son para fines generales de soporte lógico combinatorio y secuencial [94].

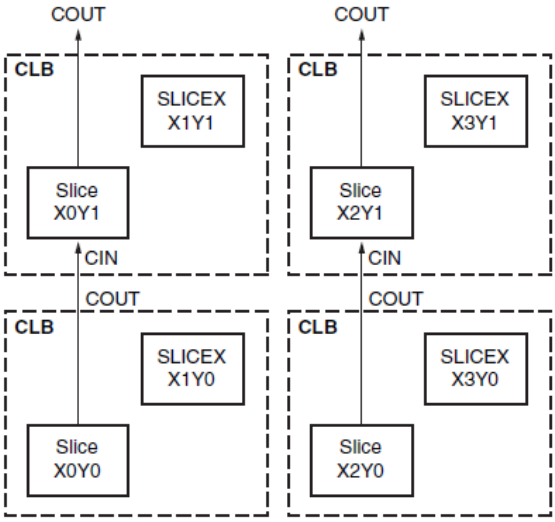


Figura 4.2 Concepto de Fila y Columna - Relación entre CLB y *Slices* [95]

4.1.1.1.2 *Slicem*

Una cuarta parte (25%) de las secciones en Spartan-6 FPGA son SLICEMs. Cada uno de los cuatro LUTs SLICEM se puede configurar ya sea como una LUT de 6 entradas con una salida, o como LUTs *dual* de 5-entrada con direcciones de 5 bits idénticos y dos salidas independientes. Estas LUTs también se pueden utilizar como RAM distribuida de 64 bits o dos veces 32 bits por LUT, como un único registro de desplazamiento de 32 bits (SRL32) o como dos registros de desplazamiento de 16 bits (SRL16s) con longitud *direccionable*. Cada salida LUT se puede registrar en un flip-flop dentro del CLB. Para las operaciones aritméticas, una cadena de transporte de alta velocidad propaga señales hacia arriba en una columna de *slices*.

4.1.1.1.3 *Slicel*

Una cuarta parte (25%) de las secciones en un FPGA Spartan-6 son *SLICELs*, que contienen todas las características de la SLICEM excepto la función de registro de memoria/*shift*.

4.1.1.1.4 *Slicex*

La mitad (50%) de las secciones en Spartan-6 FPGA son SLICEXs. Los SLICEXs tienen la misma estructura que SLICELs, excepto algunos detalles relativos a la aritmética y ancho de los multiplexores.

4.1.1.1.5 *Control de Reloj*

Cada Spartan-6 FPGA tiene hasta seis CMT (*Clock Management*), cada uno formado de dos *MCDs* y un *PLL*, que pueden ser utilizados individualmente o en cascada.

4.1.1.1.6 *DCM(Digital Clock Manager)*

El DCM proporciona cuatro fases de la frecuencia de entrada (CLKIN): desplazado 0°, 90°, 180°, y 270° (CLK0, CLK90, CLK180, y CLK270). También proporciona una frecuencia duplicada (CLK2X) y su complemento (CLK2X180). La salida CLKDV proporciona una frecuencia de reloj fraccional que puede alinear o eliminar a CLK0.

4.1.1.1.7 *PLL(Phase-Locked Loop)*

El PLL puede servir como un sintetizador de frecuencia para una gama más amplia y como un filtro de fluctuación de fase para los relojes entrantes en conjunción con los MCDs. El corazón del PLL es un oscilador controlado por voltaje (VCO) con un rango de frecuencia de 400MHz a 1080MHz, que abarca por lo tanto más de una octava. Tres conjuntos de divisores de frecuencia programables (D, M y O) adaptar el VCO a la aplicación requerida.

4.1.1.1.8 *Distribuidor de Reloj*

Cada FPGA *Spartan-6* proporciona abundantes líneas de reloj para hacer frente a los diferentes requisitos de marcado de alta *fanout*, retardo de propagación corto, y muy baja inclinación.

4.1.1.1.9 *Líneas Globales de Reloj*

En cada *Spartan-6*, están disponibles 16 líneas globales de reloj impulsadas por buffers especializados para llevar este tipo de señal, minimizando el *clock skew.* Los relojes globales son a menudo extraídos de los CMT, que pueden eliminar por completo el retardo básico de distribución de reloj.

4.1.1.1.10 *Bloques de RAM*

Cada Spartan-6 FPGA tiene entre 12 y 268 RAM en bloque de dos puertos, cada uno que almacenan 18 Kb. Cada bloque de memoria RAM tiene dos puertos totalmente independientes que comparten sólo los datos almacenados.

4.1.1.1.11 *Secciones DSP48A1*

Las aplicaciones DSP utilizan muchos multiplicadores y acumuladores binarios y se implementan en secciones DSP dedicadas. Todas las Spartan-6 cuentan con este tipo de bloques de baja potencia, que combina una alta velocidad con pequeño tamaño, a la vez que conserva la flexibilidad de diseño del sistema.

4.1.1.1.12 *Puertos de Entrada/Salida*

Todos los pines de E/S se organizan en bancos, con cuatro bancos en los dispositivos más pequeños y seis bancos de los dispositivos más grandes. Cada banco tiene varios pines de alimentación de tensión de salida VCCO común, que también alimenta ciertos buffers de entrada. Algunas áreas de entrada de una sola terminal requieren un voltaje de referencia aplicado externamente (VREF). Hay varios pines de doble propósito de VREF-E/S en cada banco.

4.1.1.1.13 *Configuración*

Los FPGA *Spartan-6* se configuran mediante la carga de un flujo de bits de datos en la memoria de configuración específica de la aplicación interna. Los datos de configuración para la FPGA pueden cargarse desde un dispositivo de memoria no volátil externo o pueden ser configurados por una fuente inteligente externa, tal como un microprocesador, DSP, microcontrolador o PC. En cualquier caso, hay dos caminos de datos de configuración general. El primero es el camino de datos en serie que se utiliza para reducir al mínimo los requisitos de pines del dispositivo. La segunda ruta de datos es el camino de datos de 8 o 16 bits utilizados para un mayor rendimiento o acceso a las interfaces estándar de la industria, ideales para fuentes de datos externos como procesadores o memoria flash paralelo de 8 o 16 bits.

Al igual que los procesadores, los FPGA de *Xilinx*® pueden ser reprogramadas un número ilimitado de veces. Dado que los datos de configuración de estos dispositivos se almacenan en los *latches* CMOS de configuración (LCC), debe ser reconfigurado después de que se apaga. El flujo de bits (*bitstream*) se carga a través de los pines de configuración de propósito específico. Estas FPGA soportan distintos modos de configuración, definidos a través de las interfaces adaptadas para tal fin (ver numeral 4.1.1.1.13.1).

El modo de configuración específica se selecciona ajustando la palabra apropiada en los pines de entrada del modo de configuración M [M1:M0] (“01” o “00”). Los bits de modo M1 y M0 deberían fijarse a un

nivel de tensión de CC constante, ya sea a través de *pull-up* o resistencias desplegables (2,4 kΩ), o vinculados directamente a tierra o VCCO_2 (Voltaje de Alimentación de Banco 2 del FPGA).

- **Encendido**. La puesta en marcha es cuando se alimenta por primera vez el FPGA. La Máquina de estados interna se restablece y el dispositivo comienza a despertar. En este punto, los pines PROGRAM_B e INIT_B del FPGA tienen nivel bajo.
- **Inicialización del dispositivo**. El dispositivo se alimenta adecuadamente, pero la memoria de configuración interna necesita ser reiniciada. Esta parte de la configuración de flujo se establece PROGRAM_B en nivel alto y un tiempo más tarde la FPGA establece INIT_B en este mismo nivel. El dispositivo puede permanecer indefinidamente en este estado si el usuario tiene el pin PROGRAM_B en bajo, de esta forma INIT_B no cambiará de nivel.
- **Cargar Configuración**. El comienzo de la fase de carga de configuración es alertada por la señal INIT en nivel alto y después se leen los pines de modo (M1: M0). En esta fase el dispositivo recibe datos de configuración. Todos los eventos de configuración se producen en el flanco de subida de CCLK.
- **Inicio**. Después de que el dispositivo recibe todos los datos de configuración, se procede a la secuencia de arranque. Esta secuencia va hasta que se activa la señal de DONE aunque el puesto a alto no representa el final de la configuración, son necesarios unos ciclos de CCLK adicionales, el número de estos ciclos después de la entrega del *bitstream* de configuración depende de las opciones del *BitGen* seleccionados, que afectan la puesta en marchan de la FGPA (por ejemplo, *DCM_lock* o *DCI_match*). Sin embargo, la mejor práctica es cargar todos los datos del archivo de configuración y continuar aplicando ciclos CCLK (mientras que los bits de datos sean todos 1) hasta que DONE se ponga en alto y finalmente, se aplican ocho ciclos CCLK para asegurar la terminación de secuencia de arranque del FPGA.
  La mayoría de los datos de configuración se pueden leer de nuevo sin afectar la operación del sistema.

4.1.1.1.13.1 Modos de Programación

Los FPGAs de familia *Spartan-6* pueden ser configuradas en cuatro modos principalmente: *Master Serial*, *Slave Serial*, *Slave Parallel* y *Boundary-Scan*, que se muestran en la Tabla 4.1.

En el modo *Master Serial*, el FPGA controla el proceso de configuración, generando una señal de reloj (por el pin CCLK) como salida, que es recibida por una memoria PROM externa y que contiene los datos de configuración. La PROM envía estos datos bit a bit al pin DIN del FPGA en cada flanco de subida del reloj.

En el modo *Slave Serial*, el FPGA recibe pasivamente la señal de CCLK de un dispositivo externo que puede ser un microprocesador o un segundo FPGA en modo Maestro.

Tabla 4.1 Modos genéricos para la configuración

Modo de Configuración	*M[0:1]*	*Ancho de BUS*	*Dirección de CCLK*
Master Serial/SPI	01	1,2,4	Salida
Master SelectMAP/BPI	00	8,16	Salida
JTAG (Boundary-Scan)	xx	1	Input (TCK)
Slave SelectMAP	10	8,16	Input
Slave Serial	11	1	Input

Para el diseño de la placa propuesta en este trabajo se han seleccionado los modos de configuración *Serial Slave* y *SelecMAP*. La generación de las señales de configuración del FPGA (PROGRAM, CCLK, DIN, D[7..0],M0) está a cargo de un microcontrolador. Sin embargo, para la validación y puesta a punto es conveniente la disponibilidad de un puerto JTAG de configuración.

Configuración Slave-Serial

La configuración en este modo se consigue estableciendo en nivel alto los pines [M0:M1] y enviando una señal reloj hacia el FPGA *Spartan-6*, además de los bits del *bitstream* que se cargan por el pin DIN en cada flanco positivo de CCLK. La Tabla 4.2 describe los pines utilizados durante la configuración *slave-serial.*

Tabla 4.2 Pines para configuración *Slave-serial*

Nombre de la señal	Dirección	Descripción
CCLK	Entrada	Reloj de configuración.
PROGRAM	Entrada	Reinicio asíncrono de la lógica de configuración.
INIT	Entrada/Salida	Entrada para retrasar la configuración. Indica que el dispositivo está listo para recibir datos de configuración, también indica algún error de configuración.
DONE	Entrada/Salida	Entrada de retardo para el arranque del dispositivo. Indica cuando la configuración está en la secuencia de arranque.
M[2:0]	Entrada	Selección modos de configuración.
DIN	Entrada	Entrada serial de dato de configuración.
DOUT	Salida	Salida de dato para *serial daisy chains.*

La comunicación serial debe de ser sincronizada para que la configuración se realice exitosamente; por lo tanto, deben cumplirse tiempos específicos de cada una de las señales. En la Figura 4.3, se muestra el diagrama de tiempos de la configuración *slave-serial*.

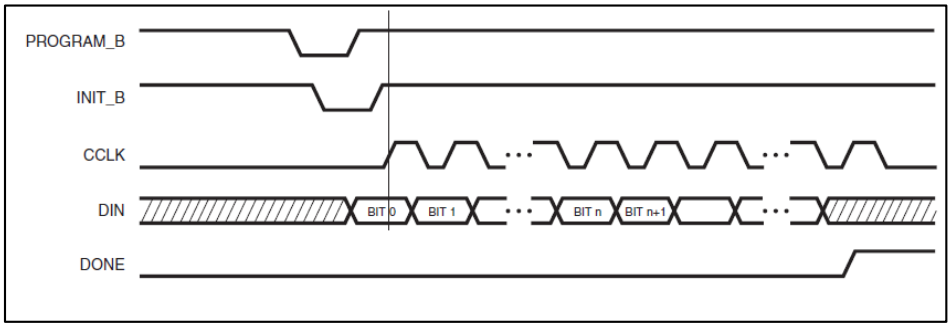


Figura 4.3 Diagrama temporal de configuración *Slave-serial* [96]

Configuración SelectMAP

La configuración *SelectMAP* tiene muchas similitudes con la configuración *Slave-serial*, pero con un poco más de complejidad. Esta configuración se hace de forma más rápida ya que se cargan 8 bits por cada ciclo de reloj; además, por este método se puede leer o reescribir la memoria de configuración de la FPGA, después de que se ha puesto en alto el pin DONE, lo que permitiría la ***Reconfiguración Parcial***.

Dos señales de control adicionales se suministran para *SelectMAP*: CS y WRITE. Estas señales deben ubicarse en nivel bajo para transferir un *byte* de configuración al FPGA. En la Tabla 4.3 se muestran los pines utilizados para esta configuración.

Tabla 4.3 Descripción de señales para la configuración

Nombre de la Señal	Dirección	Descripción
CCLK	Entrada	Reloj de configuración.
PROGRAM	Entrada	Reinicio asíncrono de la lógica de configuración.
INIT	Entrada/Salida	Entrada para retrasar la configuración. Indica que el dispositivo está listo para recibir datos de configuración, también indica algún error de configuración.
DONE	Entrada/Salida	Entrada de retardo para el arranque del dispositivo. Indica cuando la configuración está en la secuencia de arranque.
M[2:0]	Entrada	Selección modos de configuración.
D[0:7]	Entrada	Entrada del Byte de datos para configuración.
CS	Entrada	Entrada *chip-select* activo bajo.
WRITE	Entrada	Entrada activo bajo para escritura.

La sincronización con el FPGA en el momento de su configuración debe cumplir las características presentadas en el diagrama de tiempos de la Figura 4.4.

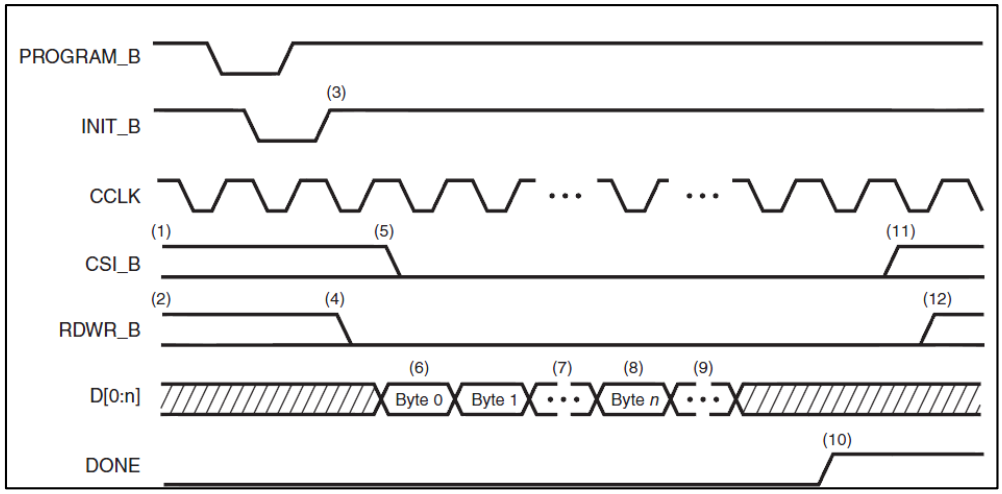


Figura 4.4 Diagrama temporal para configuración *Select-MAP* [96]

Teniendo en cuenta que las señales de CSI_B y RDWR_B son activas en nivel lógico bajo y con base en el diagrama de tiempos ilustrado, debe considerarse la siguiente secuencia durante el proceso de configuración del FPGA:

- La señal CSI_B puede ser puesta en bajo por tener un solo dispositivo conectado en el bus de *SelectMAP*, además la señal de RDWR_B también puede ser puesto en bajo mientras no se realice una acción de *readback.*

- La activación de RDWR_B debe de ser antes de CSI_B, de lo contrario, esto puede ser causa de aborto en la configuración *SelectMAP*.
- CSI_B es activado para habilitar la configuración *SelectMAP*.
- El primer byte del *bitstream* es cargado con el flanco ascendente de CCLK.
- Después de que la orden de arranque se carga, el dispositivo entra en modo de "Arranque".
- La secuencia de arranque tiene una duración mínima de ocho ciclos de CCLK
- DONE se activa durante la secuencia de arranque; además, se puede necesitar de algunos ciclos de CCLK adicionales para completar la secuencia.
- Después de finalizar la configuración, CSI_B puede deshabilitarse.
- Una vez deshabilitado la señal CSI_B se puede deshabilitar RDWR_B sin causar error en la configuración.

4.1.1.1.13.2 Los Paquetes de Configuración

Todos los comandos de flujo de bits, en un *Spartan-6*, se ejecutan mediante la lectura o escritura sobre el registro de lo que ha configurado. Los datos de configuración se organizan como palabras de 16 bits; sin embargo, algunos de ellos pueden ocupar varias palabras. Hay tres comandos principales que los datos de configuración pueden contener: **NOP**, **READ** y **WRITE**, como muestra la Tabla 4.4.

Tabla 4.4 Comandos principales en el *bitstream*

OPERACION	*CODIGO*
NOP	00
READ	01
WRITE	10

Un comando de configuración se ejecuta cuando se lee o se escribe en el registro adecuado.

4.1.1.1.13.3 Composición del *Bitstream*

La configuración puede comenzar después de que el dispositivo está encendido y la inicialización ha terminado. Después de la inicialización, el procesador de paquetes ignora todos los datos que aparecen en la interfaz de configuración hasta que recibe la palabra de sincronización. Después de esto, el procesador de paquetes espera una cabecera válida para comenzar el proceso de programación. Un *Bitstream* tiene la estructura que se muestra en la Tabla 4.5 para la configuración regular.

Tabla 4.5 Estructura básica del *Bitstream*

SECCIÓN	*DESCRIPCIÓN*	*EJEMPLO*
DUMMYWORD	Dieciséis palabras ficticias para ciclo de cambio de dirección de BPI.	0xFFFF
SYNC WORD	Dos palabras (32 bits) patrón para la sincronización.	0xAA99 0x5566
HEADER	Configuración del registro de configuración.	
CFG BODY	Dirección inicial Comando R / W FDRI / FDRO Contenido de la memoria de configuración Palabras de AUTO CRC	
HEADER2	Configuración del registro de configuración (para la cadena y las características disponibles después de la configuración margarita).	CTL
DESYNC WORD	Una palabra (16 bits) patrón que significa el final del flujo de bits.	0x000D

4.1.1.1.13.4 Formatos de archivos de datos de configuración

La herramienta CAD de *Xilinx* puede generar archivos de datos de configuración en un número de diferentes formatos, como se describe en la Tabla 4.6. El *BitGen* convierte el archivo *post-PAR NCD* en un archivo de configuración o un flujo de bits. *PROMGen*, el generador de archivos PROM, convierte uno o más archivos de flujo de bits en un archivo de PROM. Estos archivos PROM se pueden generar en diferentes formatos y no tienen que ser utilizado con una PROM, se pueden almacenar en cualquier lugar y se entregan por cualquier medio.

Tabla 4.6 Formatos de *Bitstream*

Extensión de archivo	Descripción
BIT	Archivo binario que contiene información de encabezado que no se debe descargar en la FPGA.
RBT	Archivo ASCII que contiene un encabezado de texto y ASCII 1s y 0s.
BIN	Archivo binario que no contiene la información del encabezado.
MCS EXO	Formatos ASCII PROM contienen la dirección, así como información de control.
HEX	Formato ASCII PROM que contiene únicamente los datos.

Generalmente, los archivos *.bin* o *.bit* son la fuente de datos más útiles para la programación de la memoria no volátil y más fácil de manejar cuando el medio de programación es un microcontrolador.

4.1.2 Microcontrolador

Para el desarrollo del proyecto se ha seleccionado un microcontrolador de Microchip de la familia PIC24E que incorpora en su arquitectura un módulo USB, de esta forma el planificador de tareas que se ejecute en esta arquitectura tendrá bajo su disposición las interfaces necesarias para realizar tanto la comunicación con el servidor, como los puertos para el control y configuración de los recursos de la plataforma hardware de entrenamiento.

4.1.2.1 Protocolo USB

La placa diseñada se comunicará con el servidor a través del puerto USB, dadas las tasas de transferencia de datos soportadas y la comodidad en la instalación de la tarjeta de entrenamiento.

El protocolo USB (*Universal Serial Bus*) es un protocolo serie diseñado originariamente para PCs y Macintosh, pero su popularidad hizo que se comenzara a usar también en consolas de video, cámaras fotográficas, reproductores de música o dispositivos de almacenamiento.

A nivel eléctrico, el cable USB transfiere la señal y la alimentación sobre 4 hilos. La alimentación se suministra por el primero (VBUS) con una tensión nominal de 5 V. El cuarto establece la masa (GND). La señal se transmite a través de un par trenzado con impedancia característica de 90 Ω. El reloj se incluye en el flujo de datos (la codificación es de tipo NRZI, existiendo un dispositivo que genera un bit de relleno que garantiza que la frecuencia de reloj permanezca constante) y cada paquete va precedido por un campo de sincronismo.

En cualquier caso, existen dos formas de suministrar la energía. El periférico puede obtener la energía del *host* a través del bus (*bus powered*) o puede estar autoalimentado (*self powered*). En el primer caso, es el

ordenador el que gestiona el consumo, teniendo capacidad de poner en reposo (*suspend*) o en marcha al periférico USB. En reposo, este reduce su consumo (si puede), quedándose la parte USB funcional (ideal para equipos portátiles).

Un sistema USB tiene un diseño asimétrico, que consiste en un solo servidor y múltiples dispositivos conectados en una estructura de árbol utilizando concentradores especiales. Se pueden conectar hasta 127 dispositivos a un solo servidor, pero la suma debe incluir a los concentradores también, así que el total de dispositivos realmente usables es algo menor. La ramificación máxima es de 5 niveles.

El USB es un bus basado en encuesta (*polling*), no hay interrupciones. Es un bus punto a punto: dado que el lugar de partida es el host (*PC* o *hub*), el destino es un periférico u otro *hub*. Los periféricos comparten la banda de paso del USB. El protocolo se basa en el llamado paso de testigo (*token*). El ordenador proporciona el testigo al periférico seleccionado y seguidamente, éste le devuelve el testigo en su respuesta. Este bus permite la conexión y la desconexión en cualquier momento sin necesidad de apagar el equipo [30].

El protocolo USB soporta tres tasas de transferencia: *Low Speed* (1.5 Mbit/s), *Full Speed* (12Mbit/s) y *Hi-Speed* (480Mbit/s). Los dispositivos con tasa de transferencia de 480 Mbit/s suelen denominarse como dispositivos USB 2.0, aunque no siempre se da el caso de que un dispositivo etiquetado como USB 2.0 alcance esa velocidad de transferencia (es el caso de controlador USB utilizado, que sólo llega a los 12Mbits/s, pero se denomina “USB 2.0”). Para cada tasa de transferencia, existe un límite en el tamaño de información útil que se puede enviar en cada paquete, siendo de 64 bytes cuando se selecciona el modo *Full Speed*.

El protocolo USB es flexible, soporta una interfaz serie bidireccional isócrono y permite una cómoda instalación dinámica al conectar el dispositivo al PC.

4.1.2.2 Descripción del Módulo USB de PIC24EPGU8XX

El módulo USB contiene los componentes analógicos y digitales para proporcionar un USB 2.0 de alta velocidad con un mínimo de componentes externos.

El módulo incluye soporte de *host* y dispositivo USB *full-speed* para:

- soporte de acogida de baja velocidad
- soporte *On-The-Go* USB
- resistencias de señalización integrados
- comparadores analógicos integrados para monitoreo VBUS
- transceptor USB Integrado
- controlador DMA integrado para acceder a la RAM del sistema
- soporte para los cuatro tipos de transferencia:
    - control
    - interrumpir
    - datos por lotes
    - isócrono
- colas de transferencia de hasta cuatro puntos finales sin servicio
- controlador del suministro de carga del USB (5V)

El módulo USB consiste en el generador de reloj, los comparadores de tensión USB, el transceptor, el Motor de la interfaz en serie (SIE), resistencias de *pull-up* y *pull-down*, y la interfaz de registro. La Figura 4.5 ilustra el diagrama de bloques del módulo USB OTG[97].

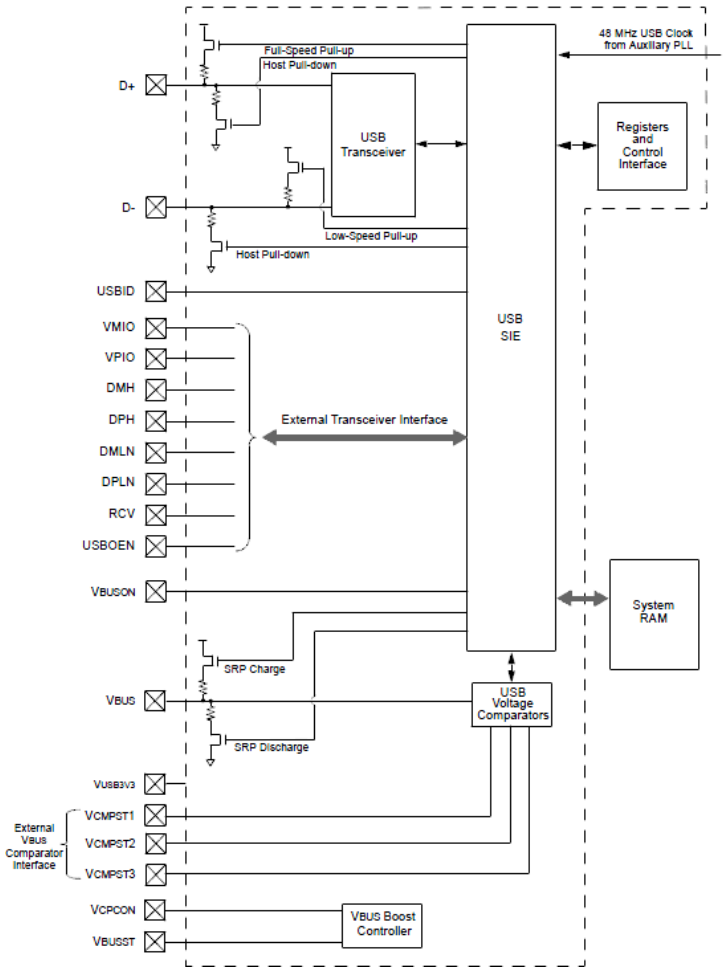


Figura 4.5 Diagrama de Bloques del Módulo USB-OTG [97]

El generador auxiliar de reloj proporciona al dispositivo un reloj de 48 MHz necesario para la comunicación USB. Los comparadores de voltaje monitorean en el pin VBUS para determinar el estado del bus. El transceptor proporciona la traducción en analógico entre el bus USB y la lógica digital. El SIE es una máquina de estado que transfiere datos hacia y desde las memorias intermedias de punto final y genera el protocolo para la transferencia de datos. El *pull-up* integrado y resistencias desplegables eliminan la necesidad de componentes de señalización externos. La interfaz de registro permite a la CPU configurar y comunicarse con el módulo.

4.1.3 Sistemas Embebidos

El término sistema embebido se refiere a la utilización de la electrónica y el software dentro de un producto, con un propósito específico y considerando las interfaces adecuadas para la obtención y presentación de información. A diferencia de un ordenador de propósito general, tal como un ordenador portátil o sistema de escritorio, el sistema embebido es especializado en el desempeño particular de la aplicación para la que

ha sido diseñado, razón por la cual, sus recursos han sido seleccionados cuidadosamente de acuerdo con restricciones de temporización, consumo de potencia, tamaño, entre otras.

En [98] el autor apunta que conforme la complejidad del software en un sistema embebido crece, resulta necesario la inclusión de componentes que resuelvan aspectos usuales:

- Coordinación de múltiples tareas.
- Cumplimiento de las restricciones temporales impuestas por la aplicación.
- Abstracción de detalles de hardware y/o de interfaces de bajo nivel.
- Implementación de funcionalidad compartida entre múltiples aplicaciones.

Un sistema embebido sencillo, con una funcionalidad simple, puede ser controlado por un programa de propósito especial o conjunto de programas sin ningún otro *software*; no obstante, en los casos más complejos se incluye un sistema operativo. Aunque es posible, utilizar un sistema operativo de propósito general, tal como *Linux*, para un sistema embebido, las limitaciones de espacio de memoria, el consumo de energía, y requisitos de tiempo real normalmente dictan el uso de un sistema operativo de propósito especial diseñado para el entorno de este tipo de sistemas.

Las siguientes son algunas de las características y requisitos de diseño de los sistemas operativos integrados [98]:

Operación en tiempo real: En muchos sistemas embebidos la exactitud de un cálculo depende parcialmente del momento en que se entrega - A menudo restricciones de tiempo real son dictadas por señales externa para controlar los requisitos para el funcionamiento del sistema.

Operación reactiva: software integrado puede ejecutar en respuesta a eventos externos, si estos acontecimientos no se producen periódicamente o en intervalos predecibles, el software incluido puede tener que tomar en cuenta las condiciones del peor caso y establecer prioridades para la ejecución de las rutinas

Capacidad de configuración: Debido a la gran variedad de sistemas embebidos, hay una gran variación en los requisitos, tanto cualitativos como cuantitativos, para la funcionalidad de sistema operativo. Por lo tanto, un sistema operativo incrustado debe prestarse a la configuración flexible de modo que sólo se proporciona la funcionalidad necesaria para una aplicación específica.

Flexibilidad de E/S del dispositivo: no hay prácticamente ningún dispositivo que necesita ser apoyado por todas las versiones del sistema operativo. Tiene sentido para manejar dispositivos relativamente lentos, como discos e interfaces de red, el uso de tareas especiales en lugar de la integración de sus unidades en el núcleo del sistema operativo.

El uso directo de las interrupciones: sistemas operativos de propósito general no suelen permitir, a cualquier proceso de usuario, utilizar interrupciones directamente.

4.1.3.1 FreeRTOS

FreeRTOS™ es un líder en el mercado del sistema operativo en tiempo real (RTOS) de *Real Time Engineers Ltd*. que soporta 34 arquitecturas y recibe 103.000 descargas al año. Se desarrolla profesionalmente, estricto control de calidad, robusto, soportado, y libre para integrar en productos comerciales, sin ningún requisito para exponer su código fuente propietario.

FreeRTOS está diseñado para ser lo suficientemente pequeño para funcionar en un microcontrolador, aunque su uso no se limita a aplicaciones de microcontrolador.

Los microcontroladores son utilizados en aplicaciones que normalmente tienen un trabajo muy específico y dedicado a hacer.

Las restricciones de tamaño y naturaleza de una aplicación dedicada, rara vez justifican el uso de una aplicación RTOS plena. Por lo tanto, FreeRTOS proporciona la funcionalidad de programación del núcleo en tiempo real, comunicación entre tareas, tiempos y sincronización primitivos. Con mayor precisión se describe como un *kernel* en tiempo real.

FreeRTOS proporciona las siguientes funciones estándar:

- Elección de la política de planificación RTOS
    - Preventivo: Siempre corre la más alta tarea disponible. Tareas de igual prioridad dividen el tiempo de CPU en igual proporción.
    - Cooperativa: Los cambios de contexto sólo se producen si un bloque de tarea lo solicita.
- Las colas de mensajes.
- Semáforos [Vía macros].
- Capacidad de visualización de seguimiento (requiere más memoria RAM).
- Mayoría de código fuente común a herramientas de desarrollo.

4.1.3.1.1 Filosofía de Diseño

FreeRTOS está diseñado para ser simple, portátil y conciso. Casi todo el código está escrito en lenguaje C, con sólo unas pocas funciones de ensamblador donde es completamente inevitable. El *kernel* RTOS utiliza varias listas de prioridades, lo cual proporciona máxima flexibilidad en el diseño de la aplicación. A diferencia de los núcleos de mapa de bits, cualquier número de tareas puede compartir la misma prioridad.

4.1.3.1.2 Modos Funcionamiento del Planificador

El *kernel* de FREERTOS permite elegir entre la multitarea preventiva y cooperativa. La elección adecuada depende de los requisitos de la aplicación. Para tomar la decisión correcta, es necesario comprender exactamente las diferencias entre multitarea preventiva y cooperativa y los resultados del funcionamiento en estos modos de operación.

4.1.3.1.2.1 Modo preventivo

La principal ventaja de un planificador preventivo es la respuesta en tiempo real en el nivel de tarea. El tiempo de respuesta tarea, es decir, el tiempo necesario para activar una tarea en espera de una interrupción depende en gran medida sólo en la latencia de interrupción.

4.1.3.1.2.2 Modo Cooperativo

Con la programación cooperativa se encuentran sustancialmente menos problemas de cambios de contexto que en un programa preventivo, ya que las tareas no se pueden interrumpir arbitrariamente por otras tareas, pero sólo en las posiciones permitidas por el programador (es decir, en las llamadas al *kernel*).

Para los programas que no requieren un tiempo de respuesta del orden de micro o milisegundos a nivel de tareas, se recomienda la programación cooperativa.

Teniendo en cuenta que en la transmisión USB se debe de realizar de forma ágil para no llegar a perder datos como sucedía en la programación de la FPGA. El mejor método para esta aplicación es el planificador preventivo.

4.1.4 CPLD (*Complex Programmable Logic Device*)

Las funciones de un Analizador lógico, cuyos detalles de diseño se discuten en el capítulo 6, se cargan en un dispositivo lógico programable complejo (CPLD) que es una combinación de una matriz de arreglos AND/OR totalmente programable y un banco de *macroceldas*. El arreglo AND/OR es reprogramable y puede realizar una multitud de funciones lógicas. Las *macroceldas* son bloques funcionales que realizan la lógica combinatoria o secuencial, y también tienen la flexibilidad añadida para *true* o complemento, junto con diversos caminos de realimentación.

4.1.5 Diseño de PCB (*Printed Circuit Board*)

Las placas de circuitos impresos (PCB – *Printed Circuit Board*) son sistemas eléctricos, con propiedades tan complicadas como los componentes y dispositivos discretos montados sobre ellos. Para el diseño del PCB se debe de tener control total sobre muchos aspectos físicos y metodológicos; sin embargo, la tecnología actual pone restricciones y límites a las geometrías y propiedades eléctricas resultantes.

La siguiente información proporciona como una guía de las libertades, las limitaciones y las técnicas para el diseño de PCB.

4.1.5.1 Estructuras de PCB

La tecnología PCB no ha cambiado significativamente en las últimas décadas, corresponde a un material con sustratos aislantes (FR4 por lo general, en compuestos epoxi/vidrio) con revestimiento de cobre gravado para formar caminos conductivos. Las capas de cobre grabado se pegan juntas en capas con sustrato aislante adicional entre ellos. Los agujeros son perforados a través de las capas. Los agujeros metalizadas forman conexiones selectivamente entre el cobre grabado de diferentes capas.

Si bien hay avance en la tecnología PCB, como propiedades de los materiales, uso en el número de capas apiladas, geometrías y técnicas de perforación, las estructuras básicas de los PCB no han cambiado. Las estructuras formadas a través de la tecnología de PCB se abstraen a un conjunto de estructuras físicas/eléctrica: trazos, planos, vías, y las almohadillas (*pads*).

4.1.5.1.1 Trazo

El trazo es una franja física de metal (generalmente cobre) que realiza la conexión eléctrica entre dos o más puntos a coordenadas XY de un PCB. El trazo conlleva la síntesis de señales entre puntos.

4.1.5.1.2 Plano

Un plano es una superficie ininterrumpida de metal que cubre la capa de PCB completo. Un plano local (*Planelet*) es un área ininterrumpida de metal que cubre sólo una porción de una capa de PCB. Típicamente, un número de *planelets* existe en una capa de PCB. Planos y *planelets* distribuyen el poder a una serie de puntos en una PCB. Son muy importantes en la transmisión de la señal a lo largo de trazos, ya que son el medio de transmisión de corriente de retorno.

4.1.5.1.3 Vías

Una vía es una pieza metálica que conecta eléctricamente dos o más puntos en el eje Z de un PCB. Las vías llevan señal y la alimentación entre capas de un PCB. En la tecnología actual PTH (*Plated Through Hole*), una vía se forma por el enchapado de la superficie interior de un agujero taladrado a través de la PCB.

4.1.5.1.4 Pads y Antipads

Debido a que las vías PTH son conductores sobre toda su longitud, se necesita un método para hacer conexiones eléctricas selectivamente entre los trazos, planos y *planelets* de las diversas capas de una PCB. Esta es la función de los *pads* y *antipads*.

Los *pads* son pequeñas áreas de cobre en las formas prescritas. *Antipads* son pequeñas áreas, en las formas prescritas, que se elimina el cobre. Los *pads* se utilizan en vías y cobre expuesto de la capa exterior para el montaje de componentes superficiales. Los *antipads* se utilizan principalmente con vías.

4.1.5.2 Estándar IPC (*Institute for Circuit Boards*)

Las normas y publicaciones del IPC están diseñadas para facilitar la interacción entre fabricantes y compradores, facilitando el intercambio y la mejora de los productos además pretende ayudar al comprador a seleccionar y obtener el menor plazo posible el producto adecuado para su necesidad particular, la aplicación de estas normas y demás publicaciones son de uso voluntario por todos aquellos que no sean miembros de la IPC, de modo tal que la norma puede ser aplicada a nivel nacional o internacional.

Esta norma establece los requisitos generales para el diseño y ensamble de tarjetas electrónicas. Todos los aspectos y detalles de los requisitos de diseño se tratan en la medida en que se aplican a un amplio espectro de diseños que utilizan materiales orgánicos o materiales orgánicos en combinación con materiales inorgánicos (metal, vidrio, cerámica, etc.) para proporcionar una estructura para el montaje y la interconexión de los componentes electrónicos, electromecánicos y mecánicos. Una vez que el montaje de componentes y elementos de interconexión se ha seleccionado, es posible suministrar un documento en el cual se especifique la tecnología elegida.

4.1.5.2.1 Propósito

Establecer los principios de diseño y recomendaciones que se utilizan en la construcción de estructuras de interconexión. El montaje de los componentes se puede realizar por medio de hoyos pasantes, de superficie o *fine pitch*. Los materiales pueden ser de cualquier combinación, capaces de realizar la función física, térmica, ambiental y electrónica dentro del diseño.

4.1.5.2.2 Jerarquía de los documentos

Los principios genéricos de diseño físico se complementan por varios documentos sectoriales que proporcionan información y mayor atención a los aspectos específicos del impreso. Ejemplo de ellos son:

- IPC-2222 Diseño de impresos orgánicos de estructura rígida.
- IPC-2223 Diseño de impresos orgánicos de estructura Flexible.
- IPC-2224 Diseño de impreso, formato de tarjeta de PC,
- IPC-2225 Diseño para módulos orgánicos *multi-chip* (MCM).
- IPC-2226 Interconexión de alta densidad (IDH) estructura diseño de la tarjeta.
- IPC-2227 Diseño de placa orgánica con cableado discreto.

Definición de términos -La definición de todos los términos utilizados en esta sección se especifican en la norma IPC-T-50.

4.1.5.2.3 Clasificación de productos

Los circuitos impresos y los circuitos ensamblados están sujetos a clasificaciones dependiendo del uso final, esta clasificación hace relación a la complejidad del diseño y de la precisión requerida en su producción.

Tipo de tarjeta - Los tipos de tarjetas varían según la tecnología por la cual se clasifican para su diseño.

Clases de rendimiento - Reflejan los aumentos de sofisticación, desempeño y resultados de las pruebas de inspección. Cada tarjeta debe contar con un documento en el cual se especifiquen los parámetros de operación. Las clases establecidas son:

- *Clase 1*: productos electrónicos generales, incluye algunos productos de consumo, computadores y periféricos.
- *Clase 2*: productos electrónicos de servicio dedicado, comunicaciones, equipos de negocios, instrumentos y material militar de alto rendimiento, en general son equipos de vida útil prolongada pero en los que no es crítico que se presenten fallas en la prestación del servicio.
- *Clase 3*: productos electrónicos de alta confiabilidad, productos comerciales y militares, donde el rendimiento debe ser continuo o el rendimiento es crítico. Los tiempos de parada son intolerables, tal como los elementos de soporte vital o sistemas de armamento críticos.

Nivel de producibilidad - Proporciona tres niveles de complejidad en características de diseño, tolerancias, medidas, montaje, pruebas de cumplimiento o verificación del proceso de fabricación, materiales o procesamiento de fabricación. Estos niveles son:

- *Nivel A:* complejidad de diseño general - Preferida
- *Nivel B:* complejidad de diseño moderada - Standard
- *Nivel C:* complejidad de diseño alta – Reducida

Estos niveles no se deben interpretar como requisitos de diseño, aunque se reconoce que la precisión, el rendimiento, la densidad patrón-conductor, equipo, montaje y ensayo pueden determinar el nivel de producibilidad del diseño. El requisito específico para alguna de las funciones que se deben controlar en el producto se especifica en el plano principal de la tarjeta o en el diagrama.

4.1.5.2.4 Requisitos generales

Las características de diseño y la selección de los materiales para una tarjeta implican equilibrar el rendimiento eléctrico, mecánico y térmico así como la fiabilidad, la fabricación y el costo de la junta. La lista de verificación de compensación identifica el efecto probable de cambio en cada una de las características físicas o de materiales

4.1.5.2.5 Diseño de la distribución

El éxito o el fracaso de un diseño, dependen de muchas consideraciones relacionadas entre sí. Desde el punto de vista de uso final del producto, el impacto en el diseño, de los parámetros característicos deben considerarse los siguientes:

- Condiciones ambientales, temperatura ambiente, el calor generado por los componentes, la ventilación, los golpes y las vibraciones.
- Si es más fácil de mantener y reparar, se debe considerar la densidad del circuito, materiales de revestimiento y componentes del ensamble.
- Interfaz de instalación que pueden afectar el tamaño y la ubicación de los orificios de montaje, ubicación de los conectores, la distribución de los elementos, soportes y otros accesorios.
- Asignaciones de procesos tales como la compensación del factor del ancho de las pistas, distancias, las tierras, etc.
- Las limitaciones de fabricación, características, espesor mínimo del recubrimiento, la forma y el tamaño del impreso, etc.
- Recubrimiento y marcas.
- Tecnologías de ensamble, montaje en superficie, atreves de orificio y mixtas.
- Producibilidad del ensamblaje en cuanto a limitaciones del equipo de fabricación, flexibilidad, requisitos electrónicos y de desempeño.

4.1.5.2.6 Requerimientos finales del producto

Los requisitos para el producto final se conocerán antes de diseñar. Los requisitos de mantenimiento y facilidad de mantenimiento son factores importantes. Con frecuencia, estos factores afectan la distribución y la instalación de conductores.

4.1.5.2.7 Diagrama esquema-lógica

Se debe proporcionar por parte del diseñador en el cual se designen las funciones de entrada y salida. Debe definir, las áreas fundamentales del circuito, blindajes, puesta a tierra y distribución de energía.

4.1.5.2.8 Listado de partes

Se debe identificar un listado de piezas, quedan excluidos los materiales utilizados en el proceso de fabricación, pero se puede incluir información de referencia, es decir, especificaciones pertinentes para la fabricación. Todas las partes mecánicas que aparecen en el diagrama esquemático tendrán asignado un número de orden que coincidirá con el número del artículo asignado en la lista de piezas.

A los componentes se les asignarán referencias de identificación. Es aconsejable agrupar elementos similares, resistencias, capacitores, IC, etc., en algún tipo de orden ascendente o numérico.

La lista de piezas puede ser escrita a mano, de forma manual en un formato estándar, o generadas por ordenador.

4.1.5.2.9 Consideración de pruebas de exigencia

Normalmente antes del diseño se debe realizar una reunión en la cual se debe examinar la capacidad de fabricación, montaje y realización de pruebas. Cualquier cambio en el diseño tiene impacto en el programa de pruebas, o la herramienta de prueba, es por esto que se debe informar a las personas adecuadas para la determinación en cuanto a la mejor solución.

4.1.5.2.10 Características y requerimiento generales del circuito

4.1.5.2.10.1 Características del conductor

Los conductores en la placa de circuito impreso pueden tomar una variedad de formas. Pueden tener forma de trazos conductores individuales. Las características críticas que pueden afectar el circuito son la distribución de inductancia, capacitancia, etc.

4.1.5.2.10.2 Espesor y ancho del conductor

El ancho y grosor de los conductores en la placa terminada impreso se determinará sobre la base de las características de la señal, capacidad de conducción de corriente requerida y el aumento máximo de temperatura permitido. El diseñador debe reconocer que el procesamiento puede variar el espesor de cobre en las capas de circuito. (Ver Tabla 4.7 y Tabla 4.8)

Tabla 4.7 Espesor interno de la capa

CAMINO DE COBRE, OZ [μM]	**ACABADO FINAL MIN. (μM)[μIN]**
1/8 OZ. [5.10]	3.1 [122]
1/4 OZ.[8.50]	6.2 [244]
3/8 OZ. [12.00]	9.3 [366]
1/2 OZ.[17.10]	11.4 [449]
1 OZ. [34.30]	24.9 [980]
2 OZ. [68.60]	55.7 [2.193]
3 OZ. [102.90]	86.6 [3.409]
4 OZ. [137]	117.5 [4.626]

MÁS DE 4 OZ. [137.20]	4 μm [157 μin] Por debajo del espesor mínimo indicado ver IPC-MF-150

Tabla 4.8 Espesor externo de la capa

CAMINO DE COBRE, OZ [μM]	**ESPESOR MÍNIMO EN SUPERFICIE DEL CONDUCTOR (μM)[μIN]**	
	Clase 1 & 1	**Clase 3**
1/8 OZ. [5.10]	23.1 [909]	28.1 [1.106]
1/4 OZ.[8.50]	26.2 [1.031]	31.2 [1.228]
3/8 OZ. [12.00]	29.3 [1.154]	34.3 [1.350]
1/2 OZ.[17.10]	33.4 [1.315]	38.4 [1.512]
1 OZ. [34.30]	47.9 [1.886]	52.9 [2.083]
2 OZ. [68.60]	78.7 [3.098]	83.7 [3.295
3 OZ. [102.90]	109.6 [4.315]	114.6 [4.512]
4 OZ. [137]	139.5 [5.492]	114.5 [5.689]

4.1.5.2.10.3 Espacio Eléctrico

Espacios libres son aplicables para todos los niveles de complejidad de diseño (A, B, C) y clases de rendimiento (1, 2, 3). Las marcas conductoras pueden tocar un conductor en un lado, pero un espacio mínimo entre el carácter marcado y conductores adyacentes se mantiene (Ver Tabla 4.9).

Tabla 4.9 Distancias mínimas entre conductores

Tensión entre conductores (V_{pico} DC o AC)	**Separación Mínima**						
	Board Descubierta				**Montaje**		
	B1	**B2**	**B3**	**B4**	**A5**	**A6**	**A7**
0-15	0.05 mm [0.00197 in]	0.1 mm [0.0039 in]	0.1 mm [0.0039 in]	0.05 mm [0.00197 in]	0.13 mm [0.00512 in]	0.13 mm [0.00512 in]	0.13 mm [0.00512 in]
16-30	0.05 mm [0.00197 in]	0.1 mm [0.0039 in]	0.1 mm [0.0039 in]	0.05 mm [0.00197 in]	0.13 mm [0.00512 in]	0.25 mm [0.00984 in]	0.13 mm [0.00512 in]
51-100	0.1 mm [0.0039 in]	0.6 mm [0.024 in]	0.13 mm [0.00512 in]	0.13 mm [0.00512 in]	0.13 mm [0.00512 in]	0.4 mm [0.016 in]	0.13 mm [0.00512 in]

- B1: conductor interno
- B2: conductor externo, sin recubrimiento, hasta 3050 msnm
- B3: conductor externo, sin recubrimiento, más de 3050msnm
- B4: conductor externo, con polímero permanente sobre el revestimiento (cualquier elevación)
- A5: conductor externo, con revestimiento protector sobre el ensamble revestimiento (cualquier elevación)
- A6: Terminación de componente externo, sin recubrimiento, hasta 3050 msnm
- A7: Terminación de componente externo, con revestimiento protector revestimiento (cualquier elevación)

4.1.5.2.10.4 Conductor de enrutamiento

La longitud de un conductor entre dos tierras debe mantenerse a una distancia mínima. Sin embargo, los conductores que son líneas rectas y se ejecutan en X, Y, o direcciones de 45 grados es preferible ayudarse con documentación automatizado para los diseños de mecanizados o automatizados. Todos los conductores que cambian de dirección, donde el ángulo incluido es inferior a 90 grados, deben tener sus esquinas internas y externas redondeadas.

En ciertas aplicaciones de alta velocidad, las reglas específicas de enrutamiento se pueden aplicar. Un ejemplo típico es el enrutamiento de serie entre la fuente de señal, carga y terminales.

4.1.5.2.10.5 Espacio entre conductores

Separación mínima entre conductores, entre los materiales conductores y los conductores se definirá en el dibujo principal. Los espacios entre conductores se deben maximizar y optimizar siempre que sea posible (ver Figura 4.6).

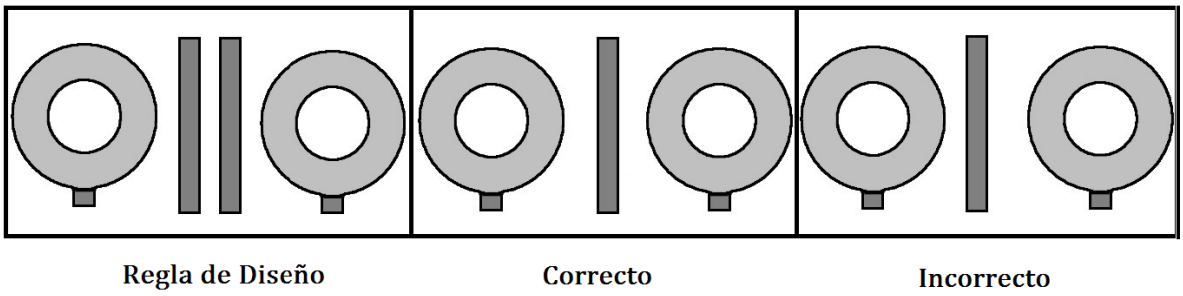


Figura 4.6 Recomendación para espacio entre conductores

4.1.5.2.10.6 Características de Tierra

Asignaciones de Manufactura - El diseño de todas las tierras tendrá en cuenta las normas de fabricación, especialmente los referentes al conductor en ancho y el espacio.

Tierras para montaje superficial - Cuando se requiere montaje superficial, los requisitos de la sección 4.1.5.2.10.1 se consideran en el diseño de la placa de circuito impreso. La selección de la forma y la posición de la geometría de la tierra, en relación con la parte, pueden impactar significativamente la unión de soldadura. La posibilidad de pérdida de calor se reduce en un estrangulamiento del conductor cerca de la zona de soldadura. El diseñador debe entender las capacidades y limitaciones de la fabricación y montaje (ver IPC-SM-782).

4.1.5.3 Reglas de *Layout* Según Fabricantes

La guía [99] proporciona información sobre el diseño de PCB para dispositivos *Spartan* ® *-6*, enfocándose en estrategias para la toma de decisiones de diseño en el PCB y el nivel de interfaz.

4.1.5.3.1 Sistema de distribución de energía

Esta sección tomada de [99], documenta el sistema de distribución de potencia (PDS) para FPGAs *Spartan-6* ®, incluyendo la selección y colocación de condensador de desacoplamiento y geometrías de PCB. Se proporciona un método de desacoplamiento sencillo para cada dispositivo en la familia *Spartan-6*.

4.1.5.3.1.1 Condensadores de desacoplamiento

La Tabla 4.10 muestra los capacitores a usar en una red de desacople simple para dos de los dispositivos Spartan-6 utilizados en este proyecto.

Tabla 4.10 Cantidad de Capacitores por dispositivo en PCB

Encap.	Ref	VCCINT (μF)			VCCAUX (μF)			VCCO Bank 0 (μF)			VCCO Bank 1 (μF)			VCCO Bank 2 (μF)			VCCO Bank 3 (μF)		
		100	4.7	0.47	100	4.7	0.47	100	4.7	0.47	100	4.7	0.47	100	4.7	0.47	100	4.7	0.47
TQG144	**LX9**	0	3	1	0	7	1	0	1	2	0	1	2	0	1	2	0	1	2
CSG324	**LX45T**	1	1	2	1	1	2	1	1	1	1	1	2	1	1	2	1	1	2

La red de desacoplamiento debe ser diseñada para cumplir o superar el rendimiento de las redes que se presentan en [99]. La impedancia de la red alternativa debe ser menor o igual a la recomendada, en frecuencias de 100 kHz a 500 MHz.

Tabla 4.11 Especificaciones del Capacitor de PCB

VALOR IDEAL	RANGO DEL VALOR	ENCAPSULADO	TENSIÓN NOMINAL
100 μF	C > 100 μF	1210	6.3V
4.7 μF	C > 4.7 μF	0805	6.3V
0.47 μF	C > 0.47 μF	0204 o 0402	6.3V

Las características de los condensadores *bulk* del PCB de alta frecuencia se especifican en la Tabla 4.11, seguidos por directrices sobre las sustituciones aceptables.

- **Condensadores *Bulk***. El propósito de los condensadores *bulk* es cubrir la gama de baja frecuencia entre el lugar en el que el regulador de tensión deja de funcionar (~ 100 KHz) y donde los condensadores de alta frecuencia comienzan a trabajar (~ 2 MHz). Tal como se especifica en la Tabla 4.10, todos los suministros de FPGA requieren condensadores *bulk*.
- **Condensadores de alta frecuencia**. Hay dos valores de condensador de alta frecuencia en la Tabla 4.11: el condensador 4,7 μF en un paquete de 0805 y el condensador de 0,47 μF en un paquete de 0402 o 0204.

4.1.5.3.1.2 Colocación de Capacitores y técnicas de montaje

Restricciones de colocación y montaje que se presentan para cada tipo de condensador de la Tabla 4.11.

- **Condensadores *Bulk***. Pueden ser grandes y difíciles de colocar muy cerca de la FPGA. Afortunadamente, esto no es problema debido a que la baja frecuencia que regula hace poco sensible su localización. Los condensadores *Bulk* se pueden colocar casi en cualquier lugar en el PCB, pero la mejor ubicación es tan cerca como sea posible al FPGA. El montaje del condensador debe seguir las prácticas normales de diseño de la PCB, que tiende hacia formas cortas y anchas que se conectan a planos de energía con múltiples vías.
- **Capacitor Cerámico 0805**. El condensador 0805 de 4,7μF cubre el rango de frecuencias medias. La colocación tiene algún impacto en su rendimiento. El condensador se debe colocar lo más cerca posible a la FPGA. Cualquier colocación dentro de dos pulgadas de borde exterior del dispositivo es aceptable.
- **Capacitor Cerámico 0402**. El condensador 0402 de 0,47μF cubre la gama media alta frecuencia. La colocación y el montaje son fundamentales para estos condensadores. El condensador debe montarse lo más cerca del FPGA como sea posible (logra la inductancia parásita menor posible).

4.1.5.3.2 Reglas recomendadas para el diseño PCB

En [99] describen las normas de diseño recomendadas para los encapsulados disponibles de FPGA *Spartan-6*.

- **Paquetes QFP.** En la Figura 4.7 se ilustra este tipo de montaje y la Tabla 4.12 muestra sus dimensiones.

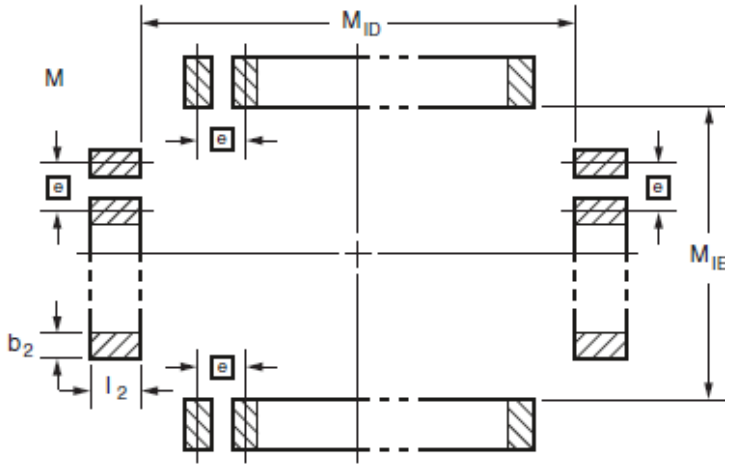


Figura 4.7 *Pads* de paquetes QFP [99]

Tabla 4.12 Dimensiones para Paquetes *Quad Flat Pack*

DIMENSIÓN	TQG144 (mm)
M_{ID}	19.80
M_{IE}	19.80
E	0.50
B_2	0.3–0.4
I_2	1.60

- **Paquetes CSP**. *Xilinx* proporciona el diámetro para *pads* de conexión en el lado del componente. Esta información es necesaria antes del inicio del diseño de la placa por lo que los *pads* de conexión pueden ser diseñados para adaptarse a la geometría en conexión del lado del componente. Los valores típicos de estos *pads* de conexión se describen en la Figura 4.8 y se resumen en la Tabla 4.13.

4.1.5.3.3 Recomendaciones de *Linear Technology*

Los dispositivos de *Linear Technology*, LT3501 y LTC3546, son doble reductores convertidores de corriente PWM DC/DC con dos interruptores internos, seleccionados para la implementación de los circuitos de alimentación.

Las características LT3501 son:

- Rango de entrada: 3.1V a 25V
- Dos Reguladores de conmutación con capacidad de 3A de salida
- Alimentación independiente para cada regulador
- Operación Ajustable/sincronizable a frecuencia fija de 250 kHz a 1,5 MHz

- Conmutación anti-fase
- Las salidas pueden poner en paralelo
- Independiente, Secuencial, radiométrico o *Tracking* absoluta entre salidas
- Mejorada protección del cortocircuito
- Baja caída: 95% Ciclo de trabajo máximo
- Desconexión por baja corriente: <10µA

Las características del LTC3546:

- Rango de entrada: 2.5 V a 5.5 V
- Rango de Voltaje de Salida: 0.6 V a 5 V
- Operación Ajustable/sincronizable a frecuencia fija de 0.75MHz a 4MHz
- Alta eficiencia: mayor a 96%
- Conmutación anti fase
- No requiere Diodos *Schottky*
- Desconexión por baja corriente: menor a 10µA
- Baja caída: 100% de Ciclo de trabajo máximo
- Protección de cortocircuito

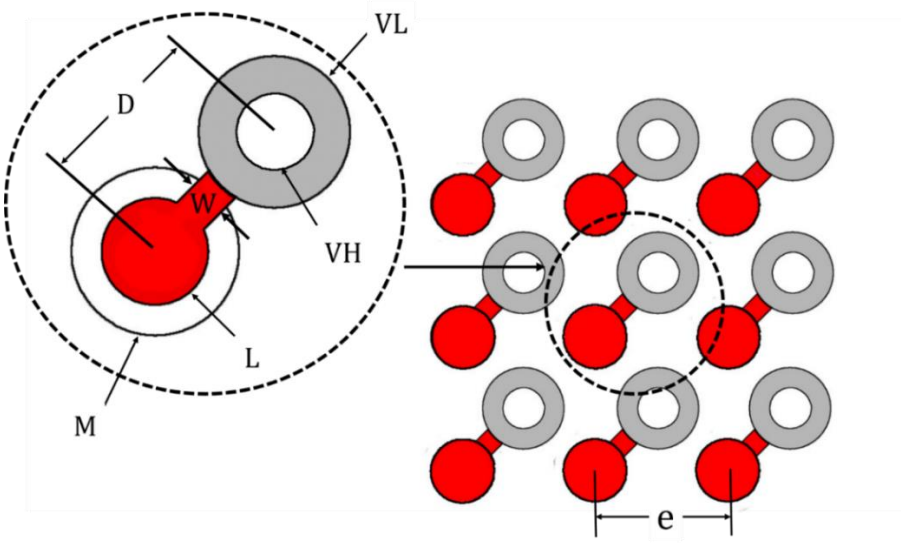


Figura 4.8 *Pads* de paquetes CSP o BGA

Tabla 4.13 Reglas de Diseño PCB Recomendadas para los paquetes de CSP (mm)

REGLA DE DISEÑO	CSG324
SMD	0.40
L	0.37
M	0.47
E	0.80
W	0.13
D	0.56
VL	0.51
VH	0.25

4.1.5.3.3.1 Diseño de PCB para los circuitos de alimentación

Para un correcto funcionamiento y una mínima inducción electromagnética (EMI) en el circuito con el TL3501, se debe tener cuidado durante el diseño del PCB. La Figura 4.9 muestra los caminos de altos *di/dt* en el circuito regulador *buck* [100].

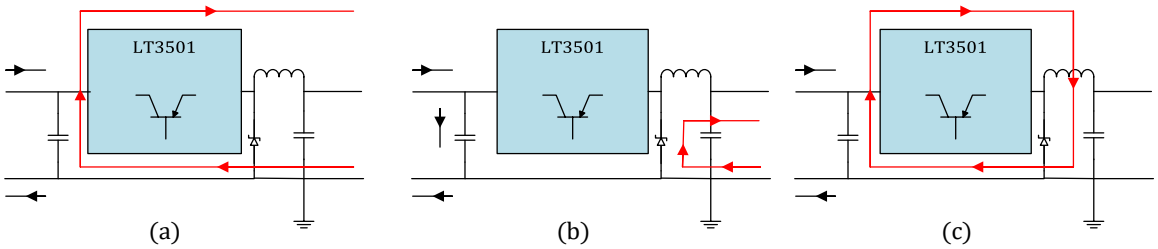


Figura 4.9 La sustracción de corriente cuando el interruptor está encendido (a) cuando el interruptor está apagado (b) muestra el camino de la corriente de conmutación de alta frecuencia (c). mantiene el bucle

Grandes corrientes conmutadas fluyen en el interruptor de encendido, el diodo de la captura y el condensador de entrada. El bucle formado por estos componentes debe ser tan pequeña como sea posible. Estos componentes, junto con el condensador y el inductor de salida, deben ser colocados en el mismo lado de la placa de circuito y sus conexiones se deben realizar en esa capa. Coloque un plano de tierra intacta locales por debajo de estos componentes, y atar este plano de tierra a tierra del sistema en un solo lugar, a ser posible en el terminal de tierra del condensador de salida C2. Además, los trazos SW y BST deben ser lo más cortos posible. El metal de la parte superior de la placa de demostración DC964A en la Figura 4.10 ilustra la colocación de componentes adecuada y rutas a trazar.

Para el diseño del PCB con el dispositivo LTC3546 debe utilizarse la siguiente lista de características para garantizar su correcto funcionamiento. Los elementos se ilustran gráficamente en el diagrama de distribución de la Figura 4.11

- Los pines SW1, SW2a, SW2b y SW1D deben de están conectados a través de un trazo amplio de cobre.
- El Capacitor CIN debe conectarse con el chip por el mismo lado de la placa de circuito impreso y lo más cerca posible. Los trazos SW irán directamente bajo el condensador. CIN proporciona la corriente alterna para los MOSFET de potencia internas y sus conductores.
- El capacitor COUT y su respectiva inductancia deben de estar estrechamente conectados
- El divisor de resistencia, R1 y R2, debe conectarse entre los terminales (+) de COUT1 y tierra cercano a GNDA. Al utilizar el PLL interno, la conexión a tierra de la red de compensación RC debe terminar en el pin GNDA.
- Los componentes sensibles deben mantenerse lejos de los pines de SW. El condensador de entrada CIN, los condensadores de compensación CFF1, CFF2, CITH1 y CITH2 y todas las resistencias R1, R2, R3, R4, RITH1 y RITH2 se deben colocar lejos de los trazos de SW e inductores L1 y L2.

Se deben rellenar todas las áreas no utilizadas de todas las capas con cobre. El rellenar con cobre reducirá el aumento de la temperatura de los componentes de potencia.

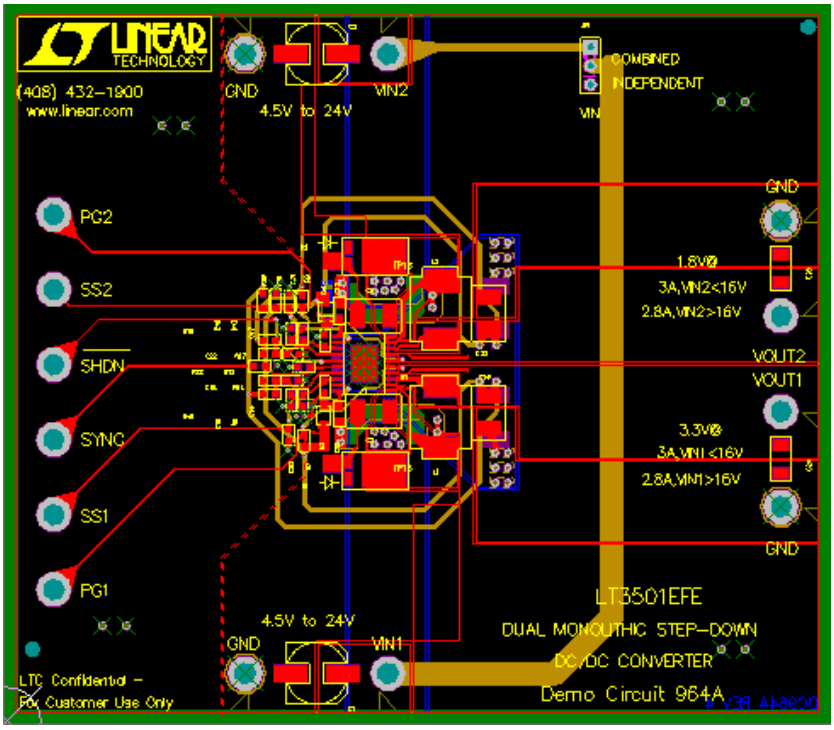


Figura 4.10 Esquema PCB de la placa de demostración DC964A (Vista en *Altium Designer 10*)

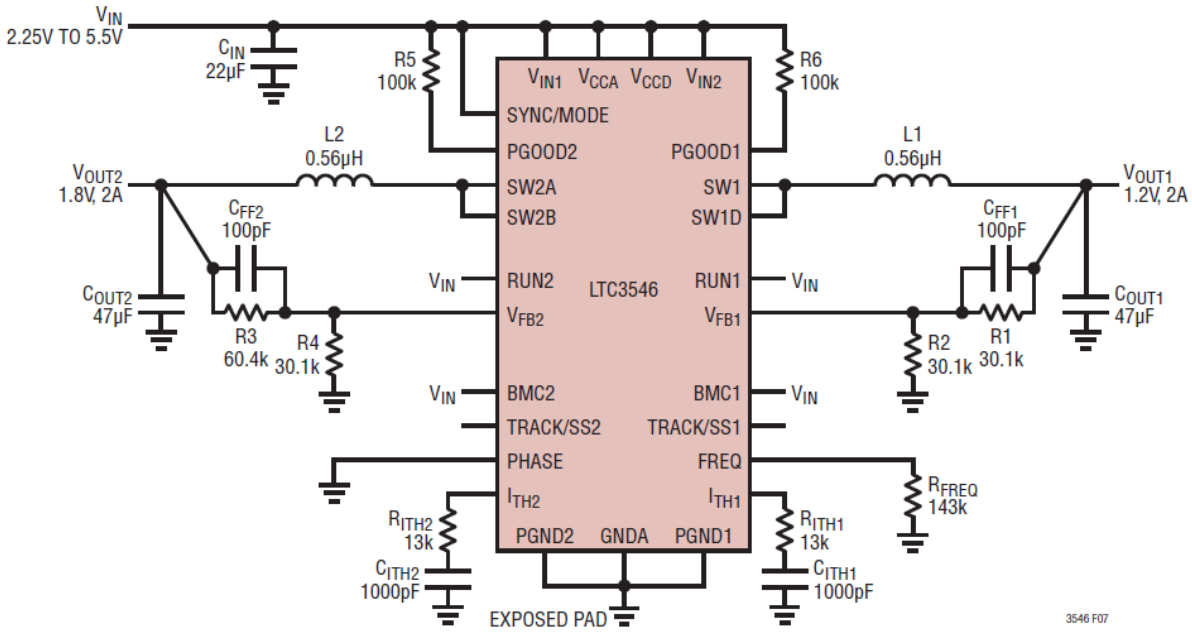


Figura 4.11 Esquema típico para Regulador 2A/2A [101]

En la Figura 4.12 se aprecia la colocación de componentes adecuada y rutas a trazar. Este es un ejemplo de la tarjeta DC1086A desarrollada por *Linear Technology*.

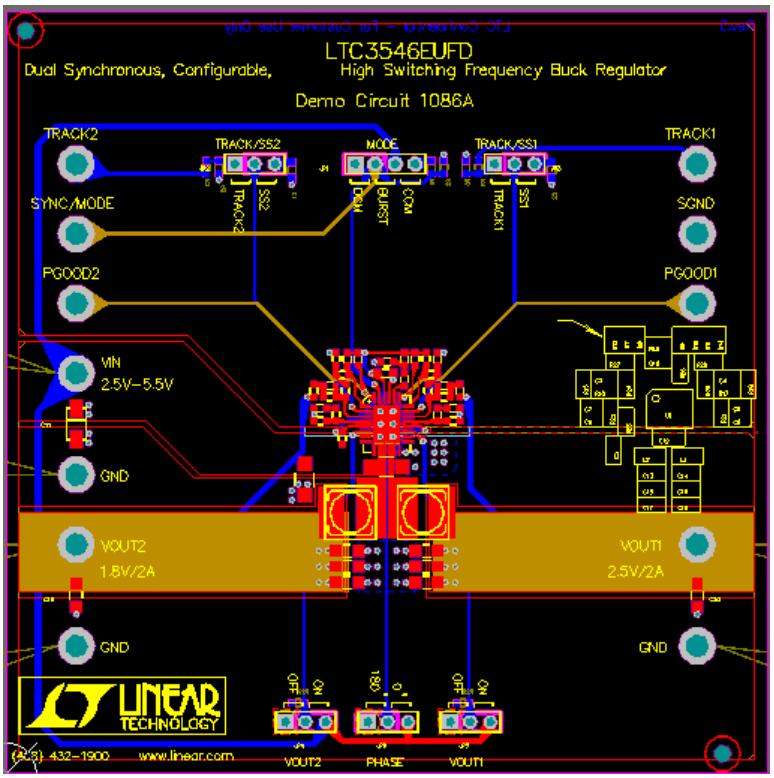


Figura 4.12 Diseño PCB de la placa de demostración DC1086A (Vista en *Altium Designer 10*)

4.2 Plataforma *Hardware* para Interacción Remota

Para el diseño del prototipo de la plataforma *hardware* se consideran los siguientes aspectos, que se encuentran relacionados entre sí:

- Modelo del Sistema
- Requerimiento del sistema
- Características y disposición de dispositivos
- Técnicas de fabricación
- Técnicas de montaje de los componentes

Además, se tienen en cuenta las recomendaciones sugeridas en las normas IPC-2222, relacionadas con el diseño de impresos orgánicos de estructura rígida, y las sugerencias de los fabricantes de los dispositivos previamente descritos para los principales dispositivos.

4.2.1 Modelo General del Sistema

La plataforma *hardware* diseñada plantea la unificación de recursos de implementación y validación en sistemas digitales. Los recursos son compartidos por medio de un servidor de recursos que hace parte de una aplicación de red propuesta por Quiñones (2013) para que usuarios remotos puedan hacer uso de la plataforma.

Como se describió en el capítulo 3, la arquitectura implementada para el laboratorio plantea el uso de tres servidores (Ver Figura 2.15). El primero es el servidor Web, encargado de controlar la interfaz de usuario

y una base de datos; el segundo servidor, es de control de *hardware* y tiene como función interactuar con la tarjeta de experimentación, a través de una comunicación USB y finalmente, una cámara IP actúa como servidor de video, para permitir al usuario un acercamiento visual con la tarjeta de experimentación.

Para el diseño de la plataforma *hardware*, también llamada tarjeta o placa de experimentación en este trabajo, se siguió una metodología *top-down*, concibiendo inicialmente un diagrama de bloques del sistema, como lo sugiere [102]. En esta instancia se designan las funciones de entrada y salida, con base en las áreas fundamentales del circuito, para posteriormente identificar los dispositivos, las especificaciones de fabricación y, con ello, definir las reglas finales de diseño del PCB.

4.2.1.1 Bloques funcionales en la plataforma hardware

La plataforma hardware consta de los siguientes subsistemas: analizador lógico y su memoria de muestreo, FPGA, interfaz de configuración, administrador del sistema basado en un microcontrolador y el módulo de Comunicación (Ver Figura 4.13).

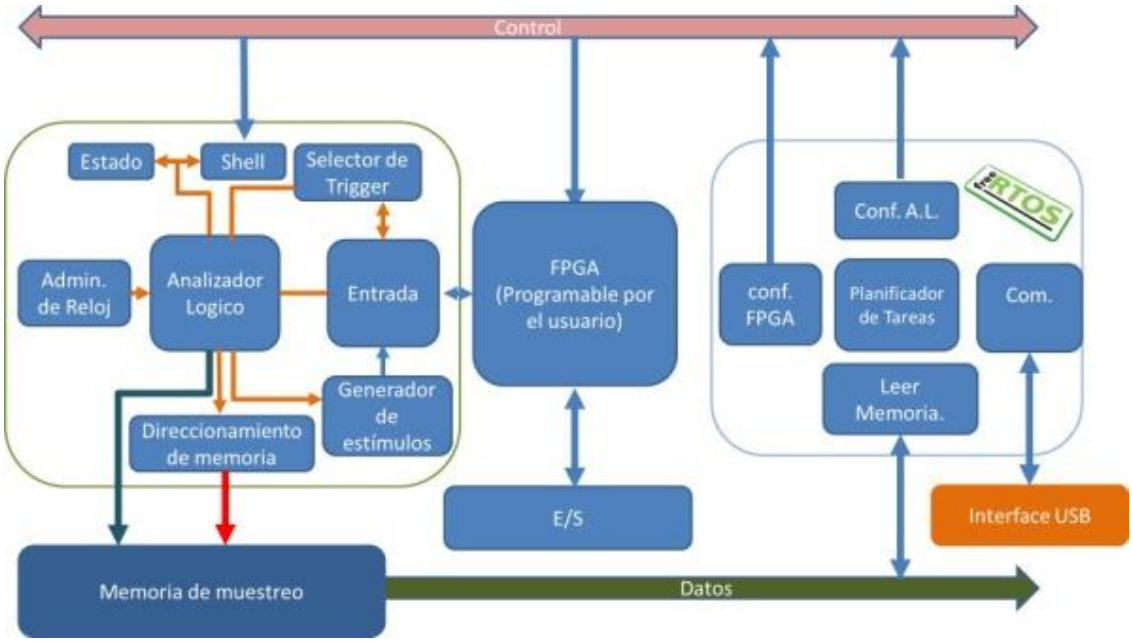


Figura 4.13 Bloques funcionales de la plataforma *hardware*

4.2.1.1.1 Analizador lógico

El Analizador Lógico es un módulo de 32 canales, de un bit cada uno, para probar las señales que están definidas por el usuario. La arquitectura de este instrumento propuesto se discute en el capítulo 5 [103] y se implementa sobre *hardware* configurable.

4.2.1.1.2 Memoria de muestro

La memoria de muestreo para analizador lógico es un módulo SRAM que almacena hasta 4M palabras de 32 bits. Su control y el tratamiento son gestionados por el mismo módulo para la lectura y escritura de los procesos. Este bloque de memoria sólo se utiliza para el almacenamiento de datos muestreados.

4.2.1.1.3 FPGA e Interfaces

Un Spartan-6 FPGA Xilinx se utiliza en la plataforma de hardware para la programación de las funciones lógicas definidas por el usuario. Este dispositivo (XC6SLX45T) tiene 43.661 elementos lógicos y 190 pines de E/S, es económica y útil para el uso de los estudiantes.

La FPGA Spartan-6 soporta varios modos de configuración paralelo/serie, como se describe en el capítulo anterior. Sin embargo, en este trabajo se utilizan dos modos poco usuales*: slave-Serie* y *SelectMAP*. De estos dos modos de programación, el más rápido y robusto es *SelectMAP*, pero por limitaciones de E/S en el microcontrolador se utiliza la versión de 8 bits para programar la FPGA con un proyecto de usuario.

La tarjeta de entrenamiento tiene también interfaces de E/S básicas como *leds*, interruptores DIP, botones pulsadores y LCD. Estos dispositivos de E/S se asignan a la misma orilla de la FPGA. Sin embargo, se requiere una interconexión directa de 32 pines entre la FPGA y los canales de entrada del módulo del analizador lógico. Los usuarios deben conocer este mapa conexiones antes de la configuración de la FPGA, con el fin de seleccionar los pines de análisis correctos. Asimismo, la tarjeta de entrenamiento tiene conectores de expansión para la conexión de módulos externos en las prácticas locales o cuando el docente tiene que planificar las prácticas específicas de interfaces con hardware externo.

4.2.1.1.4 Gestor del sistema basado en microcontrolador

La unidad de microcontrolador es un MCU de *Microchip* de 16 bits. Tiene una conexión USB con el servidor de recursos para acceder a la unidad de gestión de la tarjeta de entrenamiento. El servidor envía tramas con comandos y datos que contienen de las peticiones de usuario a la unidad de microcontrolador. Éste interpreta la información, e identifica entre estas tramas al *bitstream*, que permite la configuración de la FPGA, además de los modos de operación del analizador lógico. El proceso de carga de datos también es controlado por el microcontrolador, que lee la memoria de muestras mientras el módulo del analizador lógico direcciona la SRAM. Los datos de muestreo leídos se envían al servidor de recursos a través del módulo USB incorporado en el microcontrolador.

Un sistema en tiempo real para sistemas embebidos reside en la unidad de microcontrolador para la programación de todas las tareas exigidas. Se utiliza como núcleo el código de *FreeRTOS* con el fin de gestionar los recursos de la plataforma hardware. El código se compila en C30, mediante el entorno integrado MPLAB IDE, que permite configurar, depurar y programar este RTOS sobre el microcontrolador seleccionado, haciendo uso de *Pickit* 3 *debugger*.

La tarjeta de entrenamiento define cinco tareas para ser gestionados por el planificador del *FreeRTOS*: decodificación de instrucciones, programación de FPGA, configuración de analizador Lógico, lectura de SRAM y como tarea opcional, según lo requiera el usuario, guardar el *bitstream* en la memoria *flash*. La decodificación de instrucciones tiene la prioridad más alta, ya que se constituye como la interfaz principal del sistema, al interactuar con el servidor de recursos para el control de la plataforma de *hardware*. Todas las solicitudes de usuario pueden ser enviadas desde el servidor web hacia la tarjeta de entrenamiento, a través del intercambio de comandos por la interfaz mencionada.

Tareas de menor prioridad se pueden crear temporalmente en el momento que sea necesaria su ejecución, la tarea de decodificación de instrucciones crea o activa la tarea necesaria según el proceso que se requiera, disminuyendo las solicitudes al *kernel* y mejorando la eficiencia en la asignación del recurso de procesamiento. Este proceso de creación de la tarea se ilustra en la Figura 4.14.

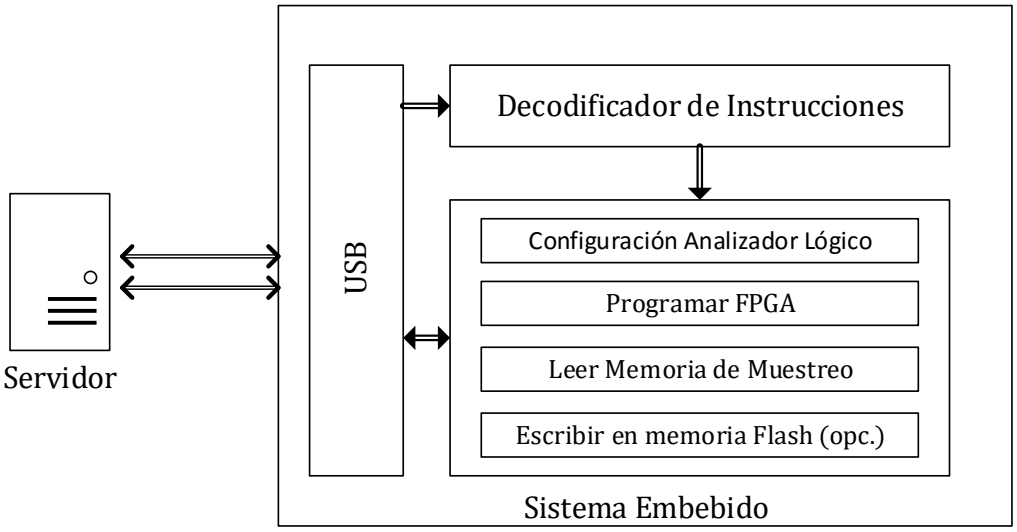


Figura 4.14 Diagrama de Tareas dentro del Sistema Embebido

Teniendo en cuenta que el desarrollo de una práctica de laboratorio sigue la secuencia de la plataforma software presentada en el capítulo 3, las acciones a ejecutarse en la tarjeta de entrenamiento obedecen a la siguiente secuencia:

- Programación FPGA
- Configuración Analizador Lógico
- Cargar Estímulos
- Enviar Resultados

De esta forma, el acceso a los recursos de la tarjeta de entrenamiento por un usuario no demanda paralelismo en las tareas seleccionadas; lo contrario sucede cuando se soporta el acceso simultáneo a múltiples usuarios, como se describe en la sección 4.3.

4.2.2 Diagrama Esquemático Propuesto

El diseño del sistema debe plantearse inicialmente de forma modular, de acuerdo con la estructura que se muestra en la Figura 4.13; sin embargo, deben considerarse también algunos módulos complementarios como alimentación y circuitos de desacoplo para la FPGA, dispuestos por el fabricante [99]–[101]. La herramienta *Altium Designer 10* facilita este diseño modular y jerárquico del proyecto, soportando la creación de diferentes hojas de esquemático, con los elementos necesarios para cumplir las funciones de cada módulo, así como buses y puertos para la interconexión entre ellos.

Para el diseño esquemático de cada módulo, se deben tener en cuenta los requerimientos técnicos de cada dispositivo que lo compone. Esta información se extrae de sus hojas de datos y permite establecer los formatos, componentes y conexiones dentro del esquemático. La construcción de esquemáticos se apoya en la inserción de librerías que contienen los componentes y sus parámetros físicos, de simulación y/o de montaje, como el tipo de encapsulado y la huella o *footprint*, necesaria para la creación del archivo PCB. Sin embargo, en ocasiones no se encuentran disponibles las librerías para algunos elementos que deben hacer parte del proyecto, por lo que resulta necesario diseñar el esquemático del componente, su *footprint* y adicionar los modelos de simulación IBIS (*Input/output Buffer Information Specification*) necesarios para realizar el

análisis de integridad de señal. Cabe resaltar que estos modelos generalmente son proporcionados por el fabricante.

A continuación se describen los diferentes bloques esquemáticos:

4.2.2.1 Fuente de alimentación

El diseño para el circuito de alimentación debe de cumplir con las necesidades de potencia de los dispositivos que conforman el sistema. La tensión requerida por cada elemento para su funcionamiento se presenta en la Tabla 4.14.

Tabla 4.14 Niveles de tensión requeridos para los principales dispositivos en la tarjeta de entrenamiento

		Dispositivo	**Voltaje de Alimentación (V)**	**Max**
FPGA	**VCCINT**	1.14	1.2	1.26
	VCCAUX	2.375	2.5	2.625
	VCCO	1.1	_	3.45
CPLD		2.95V	_	3.6V
Memoria Flash		2.95V	_	3.6V
Memoria RAM		2.95V	_	3.6V
Microcontrolador		3.0	_	3.6
LCD		3.0	3.3	4.7

En consecuencia, se han definido tres (3) circuitos esquemáticos para la fuente de alimentación, de manera que proporcionen los niveles adecuados para el suministro de potencia de la tarjeta, según se muestra en la Tabla 4.15.

Tabla 4.15 Corriente necesaria para cada nivel de tensión y su uso dentro de la tarjeta de entrenamiento

TENSION (V)	**CORRIENTE MAXIMA(A)**	**USO**
1.2	**2**	Inicialización FPGA
1.8	**2**	Programación CPLD
2.5	**2**	Suministro auxiliar de FPGA
3.3	**6**	Operación de la tarjeta
3.3	**2**	Operación del Analizador lógico y SRAM

Para la verificación de la estabilidad de los niveles de tensión requeridos, resultan necesarios los procesos de simulación y montaje de prototipos. En el desarrollo de este proyecto, las fuentes de alimentación se implementaron con fuentes conmutadas compuestas por los circuitos integrados LT3501 y LTC3546 de *Linear Technology*. La simulación de estos módulos fue apoyada en el software *LTSpice IV* 4.17,

proporcionado por el fabricante. En las siguientes figuras se muestra el diseño esquemático de cada una de las fuentes de alimentación para la placa de experimentación.

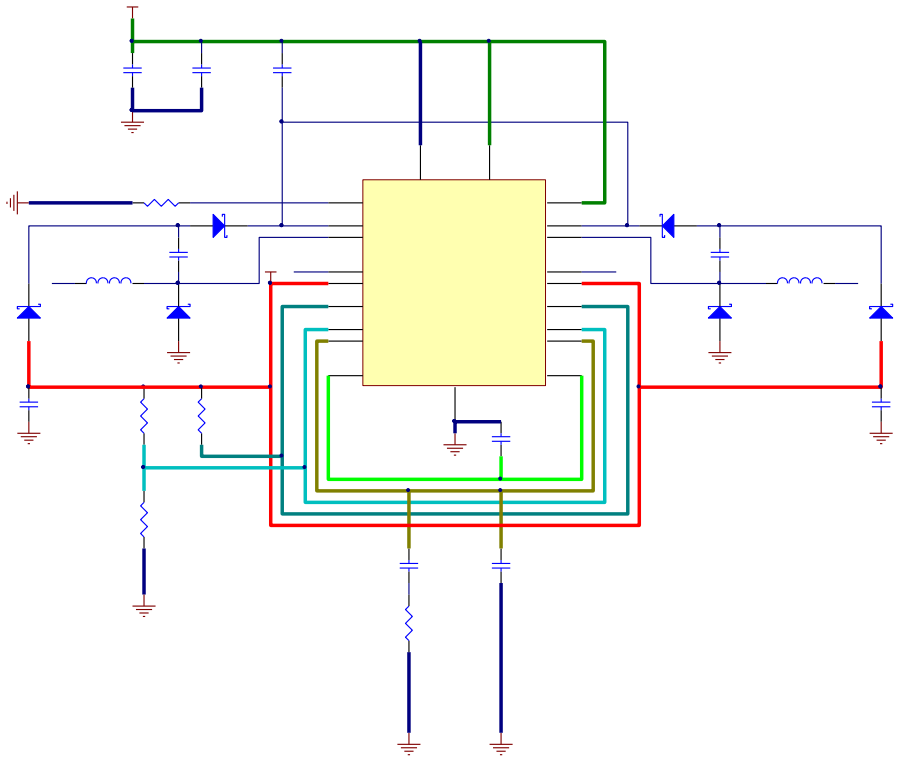


Figura 4.15 Fuente de Alimentación de 3.3V con el dispositivo *TL3501*

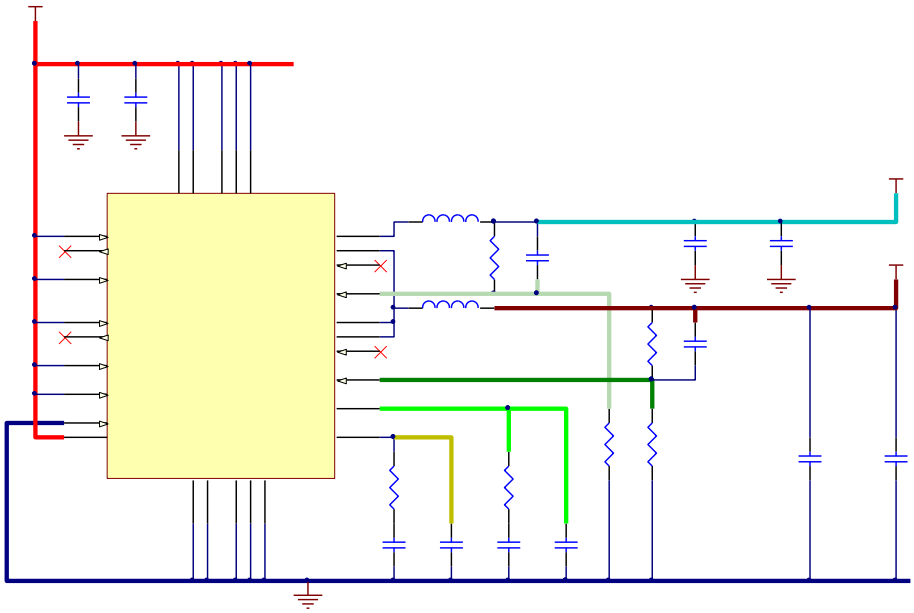


Figura 4.16 Fuente de Alimentación de 1.2V y 2.5Vcon el dispositivo *TL3546*

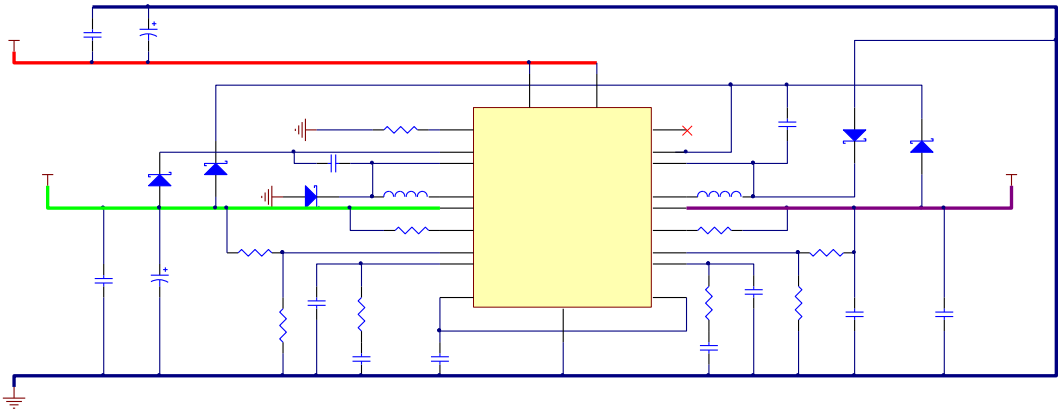


Figura 4.17 Fuente de Alimentación de 1.8V y 3.3Vcon el dispositivo *TL3501*

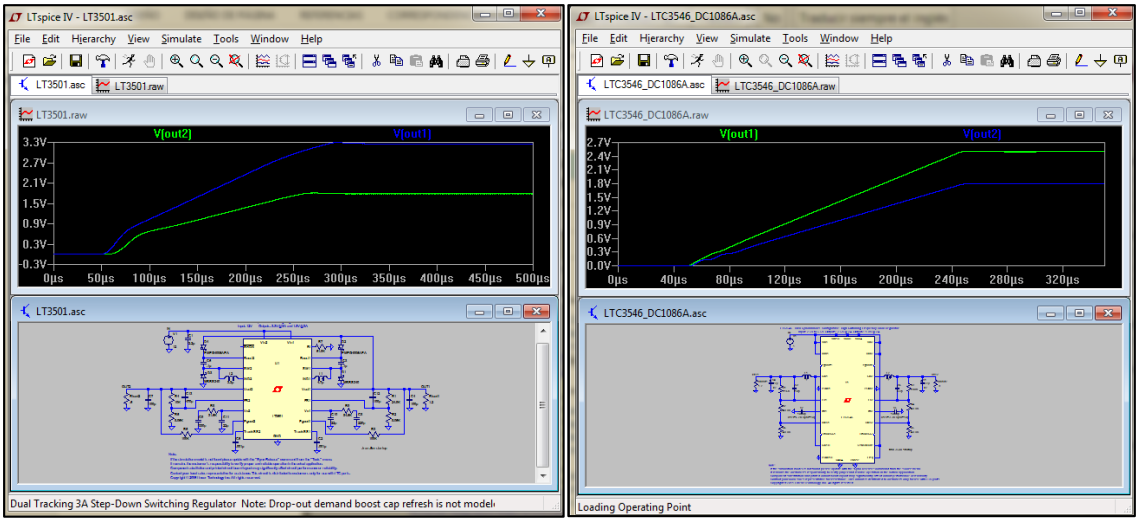


Figura 4.18 Simulación de circuitos de la Fuente en LTspice IV

4.2.2.1.1 Simulación funcional de las fuentes de alimentación

Los circuitos integrados de regulación utilizados en el diseño de las fuentes de alimentación tienen soporte de simulación en el programa *LTspice IV*, que suministra *Linear Technology*. El fabricante proporciona también diagramas esquemáticos que corresponden a configuraciones genéricas para reguladores conmutados de características muy específicas de tensión y potencia, como se muestra en la Figura 4.18. Estos circuitos son tomados como base para el diseño de las fuentes de regulación de la tarjeta de entrenamiento, con ligeros cambios en los valores R, L y C.

En la simulación realizada se pueden medir los tiempos de estabilización del voltaje, de salida, además de observar los efectos de cargabilidad y tolerancia a fallos.

4.2.2.2 Esquemático FPGA

El FPGA es el elemento central del módulo “Recurso Programable”. El diagrama esquemático que involucre este dispositivo debe contemplar circuitos de alimentación, desacoplo, comunicación, reloj, programación y expansión tanto para E/S como para conexión con el analizador lógico (Ver Figura 4.19). Los dispositivos de alimentación y programación corresponden a pines dedicados y/o específicos para realizar estas conexiones. Los dispositivos de E/S pueden ir conectados a pines de propósito general y la ubicación de estas conexiones son más flexibles, al igual que la ubicación física de cada uno de ellos en el PCB.

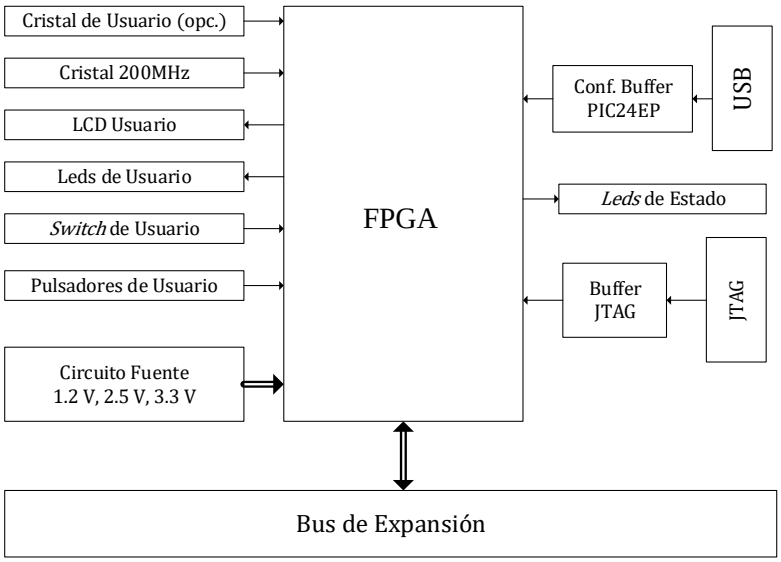


Figura 4.19 Periféricos de la FPGA

Tabla 4.16 Cantidades de condensadores requeridos por dispositivo

Banco	**Capacitores (uF)**		
	100	4.7	0.47
VCCINT	1	1	2
VCCAUX	1	1	2
VCC0 Bank 0	1	1	1
VCC0 Bank 1	1	1	2
VCC0 Bank 2	1	1	2
VCC0 Bank 3	1	1	2

Una de las principales recomendaciones del fabricante es la selección de capacitores de desacoplo para cada línea de alimentación de la FPGA, que para el dispositivo que se ha seleccionado en este proyecto, corresponde a los mostrados en la Tabla 4.16.

4.2.2.3 Diagrama Esquemático de Microcontrolador

El microcontrolador gobierna el módulo de control. Como se muestra en la Figura 4.20, el dispositivo se conecta directamente a la FPGA para programarla, con el analizador lógico para configurar el modo de operación y con el banco de memorias RAM para acceder a los datos de resultado del análisis. Además, cuenta con el soporte de una memoria Flash e interfaz de comunicación USB para interactuar con el servidor de recursos. El diseño esquemático contempla todas estas conexiones además de circuitos de desacoplo.

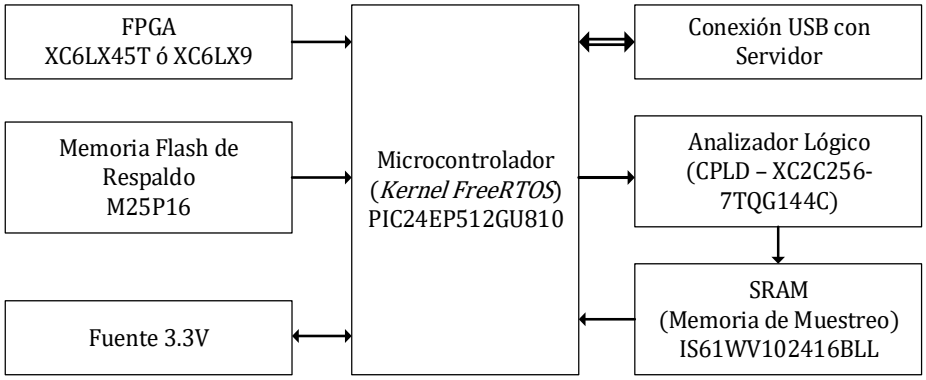


Figura 4.20 Diagrama de conexiones para uC

La distribución de pines del microcontrolador considera el tamaño de los buses de conexión con cada dispositivo, como se muestra en la Tabla 4.17.

Tabla 4.17 pines de E/S requeridos para la conexión del microcontrolador con los dispositivos a controlar

CONEXION	***TAMAÑO (bits)***
Analizador Lógico	8
FPGA	13
Memoria Flash	6
Memoria RAM	32
Puerto USB	2

El microcontrolador debe contar con el soporte para la programación IC (*in-circuit*) de las aplicaciones que ha de realizar, razón por la cual se utiliza el programador *pickit 3* de *microchip*. Las líneas de programación se muestran en la Tabla 4.18.

Tabla 4.18 Pines de programación del uC

Pines PIC24E	***Dirección***
MCLR	Input
PEGD	Output
PEGC	Input
VDD	Alimentación
GND	Referencia

El microcontrolador seleccionado, el PIC24EP512GU810, pertenece a una nueva familia de Microchip posterior a las familias PIC24F. Por esta razón, las librerías de *Altium designer 10* no lo incluyen y hace necesario crear una nueva librería de esquemático, así como adicionarle un modelo de simulación IBIS y

un modelo de *footprint* para la posterior integración a un proyecto PCB. En la Figura 4.21 se muestra el diagrama esquemático para el microcontrolador.

La designación de los pines para la conexión con la memoria RAM se hace a través de los 2 puertos de 16 bits disponibles. Las interfaces físicas para la programación *SelecMAP* y la configuración del analizador lógico se mapea en un puerto de 8 pines. Finalmente, se establecen conexiones para las señales de sincronización con la memoria *flash* de respaldo y con la programación de la FPGA.

4.2.2.4 Identificación de tareas y configuración de planificador.

El sistema encargado del control de los diferentes recursos presentes en la tarjeta, requiere la ejecución de tareas específicas, conforme se muestra en la Figura 4.22.

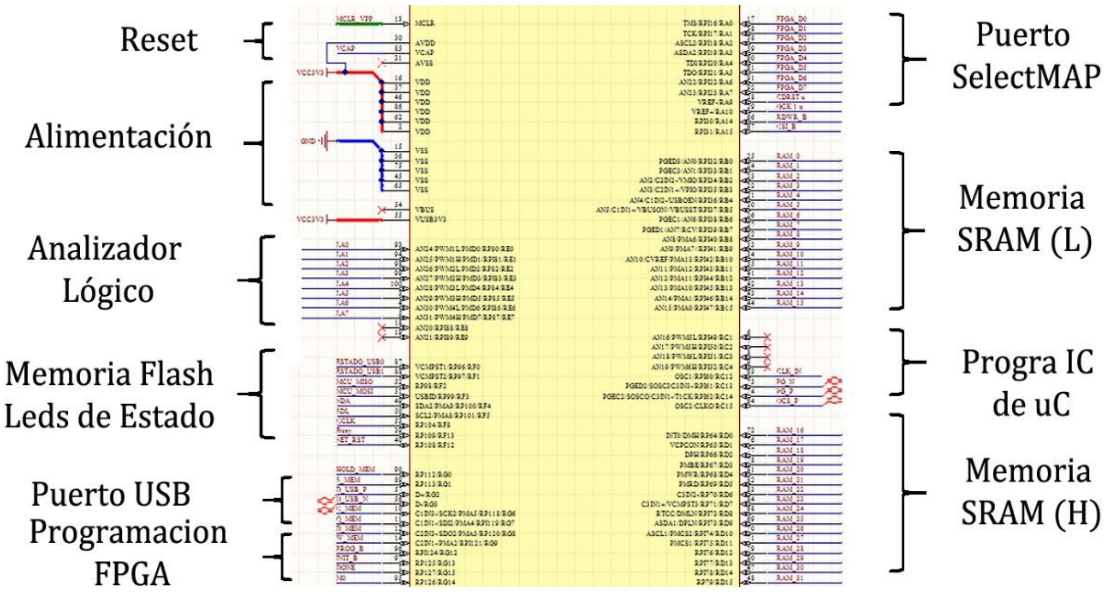


Figura 4.21 Diagrama esquemática para uC

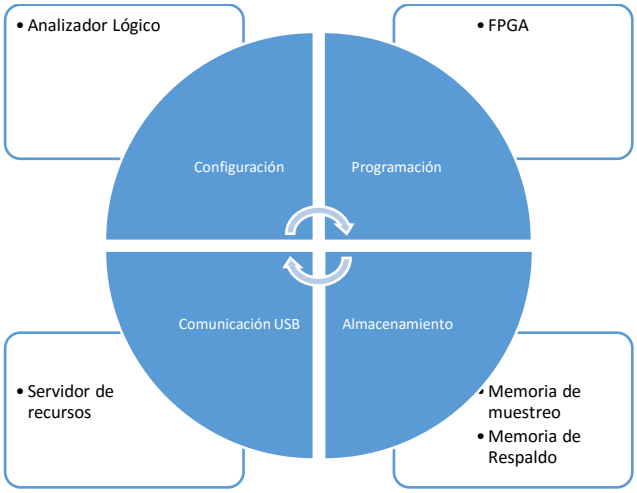


Figura 4.22 Módulos del diseño general

Durante el proceso de una práctica de laboratorio remoto, el servidor sigue una secuencia de envío y solicitud de tramas en el siguiente orden:

1. Envío archivo *bitstream* para la configuración de la FPGA.
2. Envío de tramas para la configuración del Analizador Lógico.
3. Envío de la tramas correspondiente a los estímulos
4. Solicitud de datos de la prueba (captura de datos).

Por esta razón, se establecen una serie de comandos para identificar el proceso que se debe de ejecutar, por lo que resulta importante realizar la identificación de este encabezado para ejecutar la tarea adecuada, dependiendo del comando recibido (Ver Figura 4.23), justo antes de la trama de información.

De esta forma se conforman las siguientes tareas a ejecutar:

- Decodificación de instrucciones
- Leer/escribir *bitstream* en memoria *Flash* (opcional)
- Programación de FPGA
- Configuración y operación del analizador lógico
- Transferencia de muestras desde la memoria

En la tarea que se denota como opcional, el usuario debe seleccionar que el archivo *bitstream* se guarde en la memoria *Flash* de soporte con la que cuenta el microcontrolador, como almacenamiento de respaldo.

Figura 4.23 Comandos de identificación de proceso

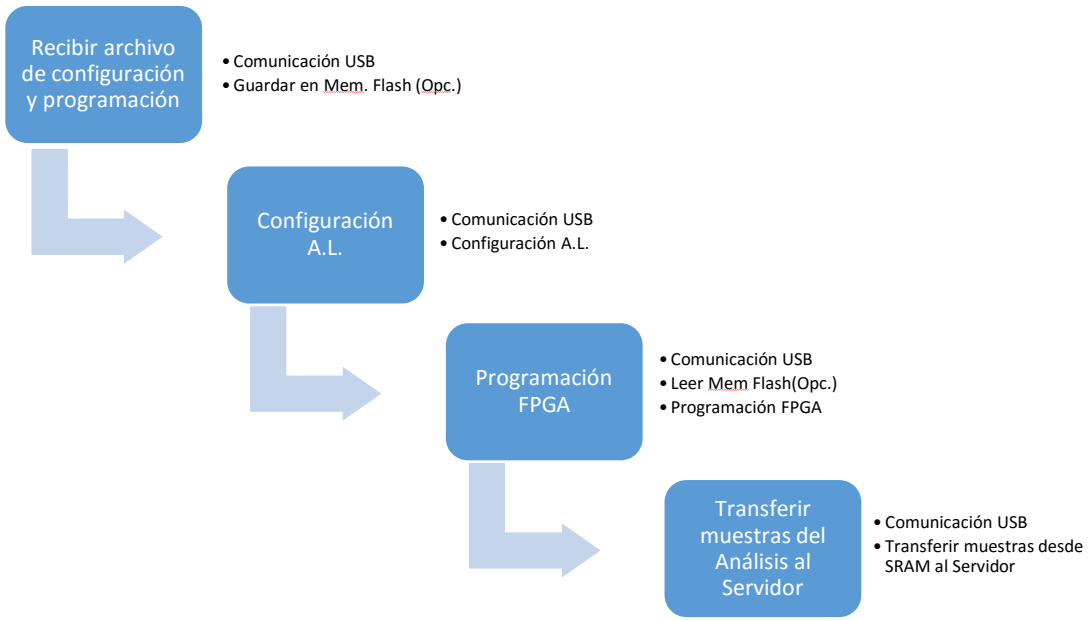


Figura 4.24 Secuencia de ejecución de las tareas

Teniendo en cuenta que algunas de las tareas que se ejecutan son utilizadas una sola vez dentro del proceso de la práctica, se puede eliminar para disminuir la carga tanto en memoria RAM como en procesamiento

del microcontrolador. Se definen entonces las tareas que se necesitan en cada uno de los pasos realizados para la práctica de laboratorio (Ver Figura 4.24).

El planificador de tareas implementado con *FreeRTOS* controla el tiempo de ejecución de estas tareas, según dos modos de operación: modo apropiativo y modo cooperativo. El mejor método para este proyecto lo definirán las pruebas que se realicen del funcionamiento con estos dos modos de operación.

4.2.2.5 Interfaz de Comunicación USB

El uC PIC24EP512GU810 soporta la comunicación USB y su fabricante *Microchip* entrega un *framework* [104] con el soporte para el desarrollo de diferentes aplicaciones USB. Para lograr la comunicación con el servidor se debe incluir la librería *libusb*, utilizada también en el diseño de la plataforma software expuesta en el capítulo 3. Esta librería de C facilita el acceso a las aplicaciones de los dispositivos USB en varios sistemas operativos y es un proyecto de código libre. El soporte de la librería *libusb*, suministrado por Microchip en el *framework*, está configurado para la tarjeta de desarrollo PIC24E USB *Starter Kit*; por lo tanto, hay que hacer las modificaciones necesarias que permitan acoplar las funciones a la tarjeta de entrenamiento. Entre los ajustes a aplicar se encuentran la configuración en los registros del microcontrolador para cambio de reloj del USB, desactivar los comparadores analógicos integrados para monitoreo VBUS, ya que el diseño en la tarjeta de entrenamiento no contempla esta conexión y el modulo USB permanecería inactivo. Además, se debe de acoplar la librería en el proyecto del *FreeRTOS* para que las tareas en el planificador tomen al módulo USB como uno de sus recursos.

La forma ubicar un dispositivo en el bus USB del servidor es a través de sus parámetros *vendorID* (0x04D8) *y ProductID* (0x0204), que están especificados en la descripción del dispositivo y deben coincidir con los valores configurados en la aplicación *java* para lograr una comunicación USB exitosa entre el microcontrolador y el Servidor.

4.2.2.6 Programación del FPGA

El primer paso desde el servidor es el envío del archivo para programar el FPGA, para lo cual se hace uso de la aplicación java desarrollada en [105], donde abre el archivo *bitstream* de extensión *.bit y lo copia a un buffer que posteriormente es leído y envía paquetes de 64 Bytes (tamaño máximo dispuesto para la comunicación USB en el microcontrolador). Estos paquetes se leen del buffer USB, creando un puntero de memoria para la lectura de los bytes de cada una de las 64 posiciones e identificar los campos importantes del *bitstream*, localizando los paquetes válidos para comenzar el proceso de configuración. El microcontrolador ignora todos los paquetes hasta encontrar la palabra de sincronización; la primera palabra que encuentra para la programación en *selectMAP* es 0x5566. Después de esto, es necesario provocar el proceso de inicialización del dispositivo estableciendo PROGRAM_B para reiniciar la memoria interna de la FPGA. A continuación de la inicialización, el microcontrolador envía los paquetes de configuración, incluida la palabra de sincronización, por el puerto *SelectMAP* de 8 bits. Por cada byte enviado, este procesador señaliza un ciclo de reloj, como se muestra en la Figura 4.4. Los datos se envían hasta encontrar el comando de la desincronización (0x000D), a partir del cual el FPGA entra en la secuencia de arranque, que requiere un mínimo de ocho (8) ciclos de reloj adicionales. Sin embargo, para asegurar el inicio de la FPGA se envían veinte (20) ciclos de reloj, al cabo de los cuales se lee el pin DONE, que se establece siempre que la programación es exitosa. El proceso descrito se muestra gráficamente en la Figura 4.25.

Cuando el pin DONE no se establece en nivel alto después de terminado el proceso de programación del FPGA, significa que hubo un problema en la programación.

4.2.2.7 Configuración y operación del analizador lógico

Las tramas que el microcontrolador recibe desde el servidor se envían al analizador lógico con el protocolo asíncrono *4-phase Handshake*, con 8 bits de datos y dos señales de control REQ y ACK (CLK0 y CLK1 en el esquemático del microcontrolador). La trama de comunicación establecida en 3.4.8, se envía desde el servidor con la información correspondiente a la configuración del instrumento, como se presenta en la Figura 4.26. Nótese la longitud de la trama para cada uno de los modos de funcionamiento del analizador lógico (ver sección 3.4.8.2.1).

La tarea encargada de configurar al analizador lógico debe procesar la trama de comunicación teniendo en cuenta campos como el tipo de muestreo, activación de disparo, tipo de disparo y canal de disparo. Cada uno de estos parámetros tiene asociado un comando que sería enviado desde el microcontrolador hacia el analizador lógico, para hacer efectiva su configuración (Ver comandos en capítulo 5). En el caso de tener seleccionado el disparo por nivel, para 16 y 32 bits, se debe considerar una palabra de disparo que se adiciona a la trama. El diagrama de la Figura 4.27 muestra el proceso realizado por la tarea de configuración en el microcontrolador.

Después de realizar la configuración del analizador lógico, el servidor inicia la transferencia de los estímulos hacia el microcontrolador. Las tramas de estímulos se reciben como se presenta en la Figura 4.28.

En el microcontrolador se crea una constante N con la cantidad máxima de estímulos, lo que permite el control en el envío de esta trama hacia el analizador lógico; el valor de *N* lo define la cantidad de memoria asignada para estímulos. Los *bytes* de estímulos son enviados mientras el servidor continúa enviando estos paquetes hasta el final del proceso (Ver Figura 4.29). Una vez terminado este proceso, el microcontrolador queda a la espera de la solicitud de inicio de adquisición.

El inicio de adquisición (comando STRTAQ) depende de la orden del usuario en la aplicación Web (Ver sección 3.4), que llevará al analizador lógico al estado *running*. La adquisición culmina cuando la memoria está llena en el estado *Triggered* o con la solicitud de pausar (comando PSACQ), en el caso de adquisición continua. Una vez el analizador se encuentre en el estado *Acq Complete*, el sistema queda a la espera de la solicitud de descarga de muestras.

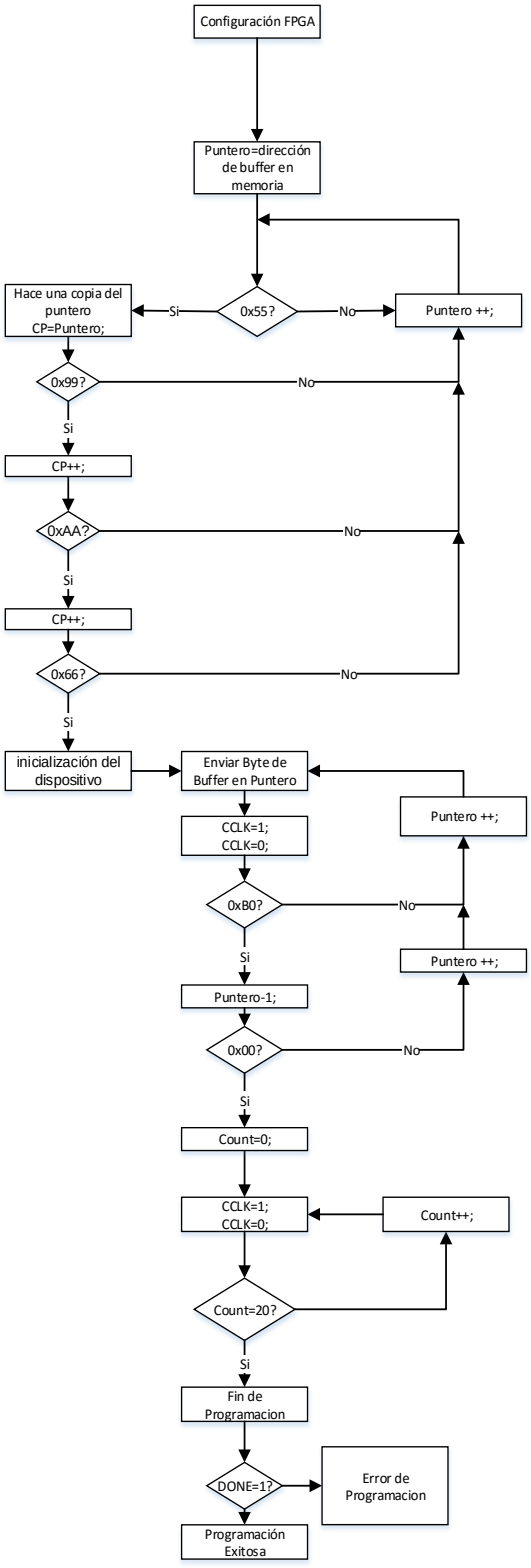


Figura 4.25 Diagrama de proceso para la configuración de FPGA

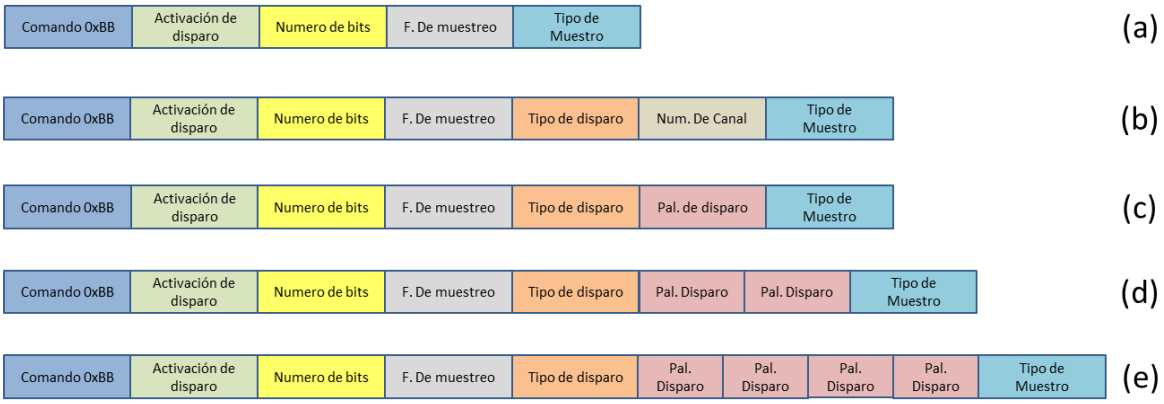


Figura 4.26 Trama de comunicación , (a) *Trigger* Continuo (b) Disparo por flanco (Pos/Neg) (c) Disparo por Nivel, 8 bits (d) Disparo por Nivel, 16 bits (e) Disparo por Nivel, 32 bits (ver sección 3.4.8.2.1)

4.2.2.8 Transferencia de muestras desde memoria

El servidor solicita el inicio de la transferencia de los datos de muestreo desde la memoria hacia el terminal (estado *Uploading Data* del analizador lógico*)*, y el microcontrolador lo transmite mediante el comando RDSMEM. El proceso de transferencia consiste en leer los datos muestreados. El microcontrolador tiene capacidad de enviar paquetes de 64 *bytes* y el canal para análisis es de 4 *bytes*; pero para enviar datos por USB se deben segmentar estos 4 *bytes* en datos de 1 *byte*, como se muestra en la Figura 4.30.

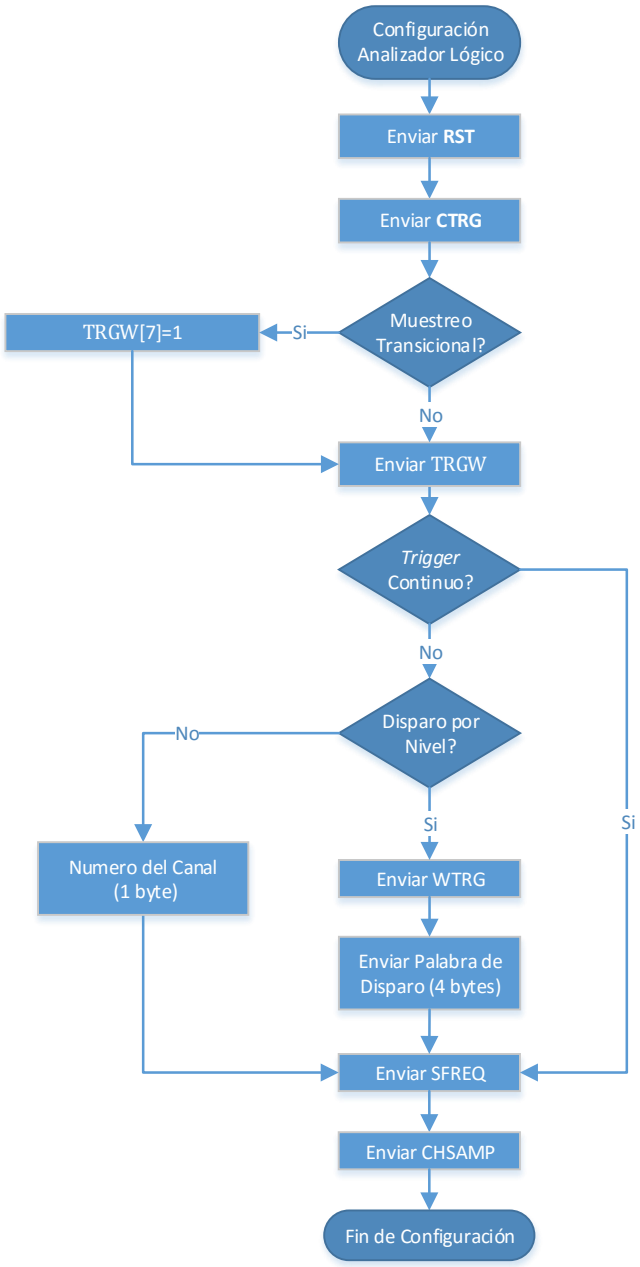


Figura 4.27 Diagrama de flujo para el envío de parámetros de configuración al analizador lógico

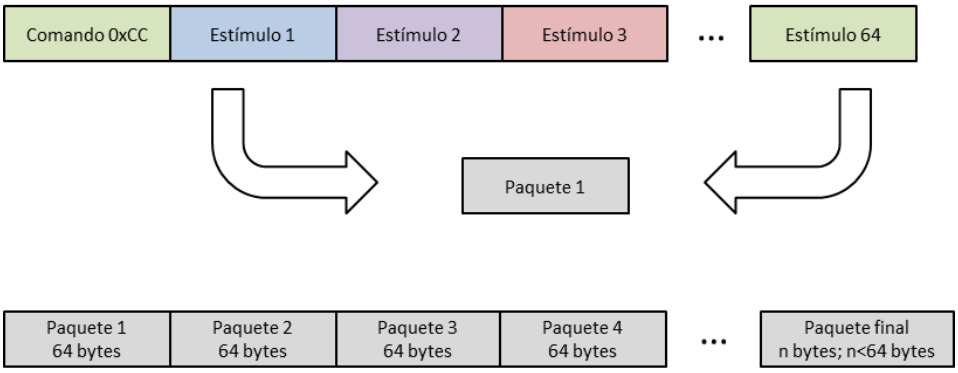


Figura 4.28 Trama de estímulos

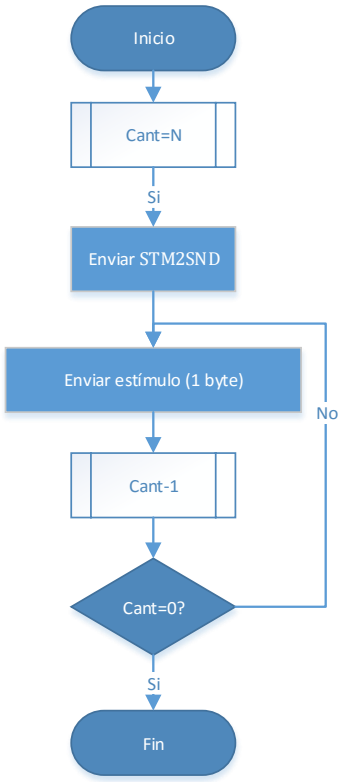


Figura 4.29 Diagrama de flujo para el envío de estímulos al analizador lógico

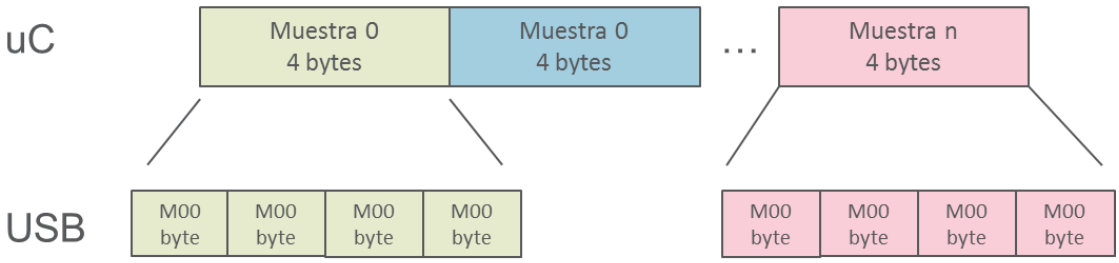


Figura 4.30 Trama USB para enviar los estímulos hacia el servidor

4.2.3 Diseño del PCB

De acuerdo con la metodología de diseño abordada, se especifican dos circuitos independientes que permitan realizar pruebas modulares de funcionamiento y faciliten la realización de cambios, considerando aún trabajos futuros sobre la misma plataforma hardware. En el diagrama de bloques de la tarjeta, presentado en la Figura 4.13, se identifican las funciones principales de este sistema en dos módulos fundamentales: un bloque principal de configuración (microcontrolador, FPGA, dispositivos de E/S) y uno secundario, correspondiente al circuito del instrumento analizador lógico; cada uno de ellos con sus respectivas fuentes de alimentación y los conectores que permiten el intercambio de información. Los módulos especificados son mostrados en la Figura 4.31.

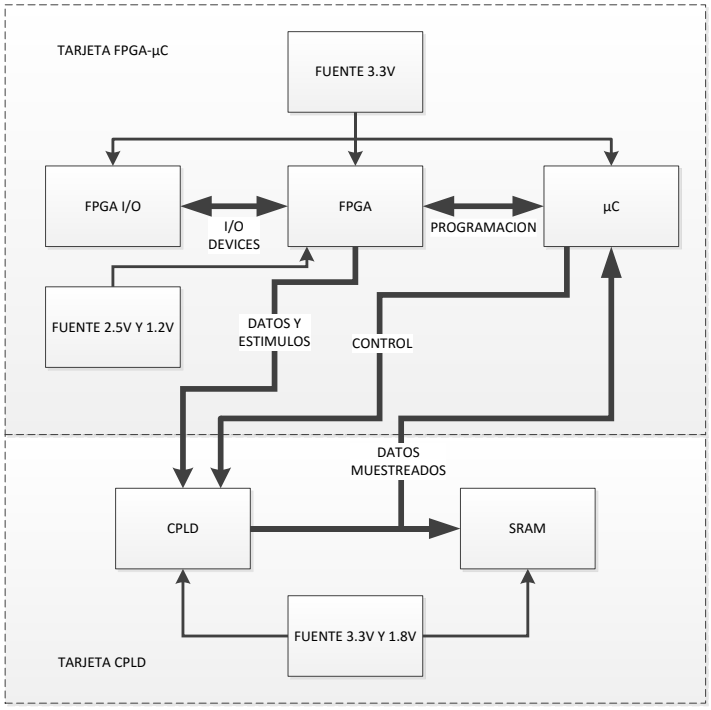
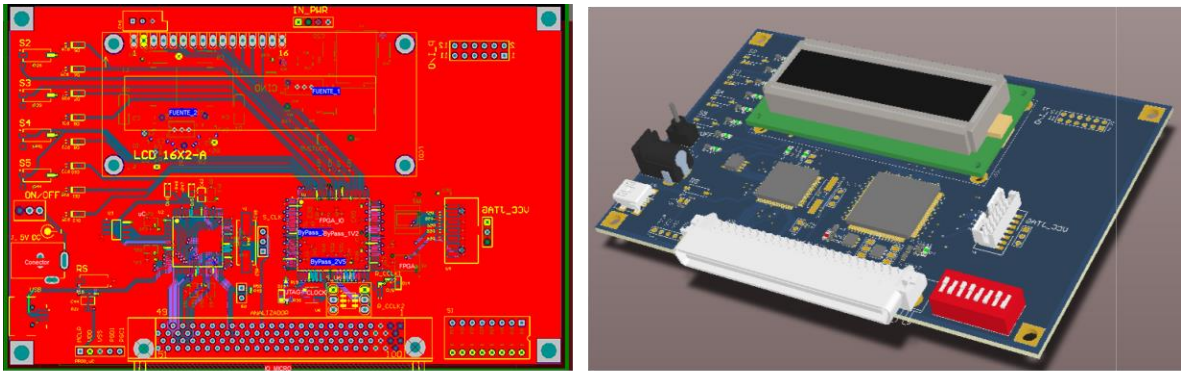


Figura 4.31 Diagrama de bloques de la tarjeta de desarrollo [106]

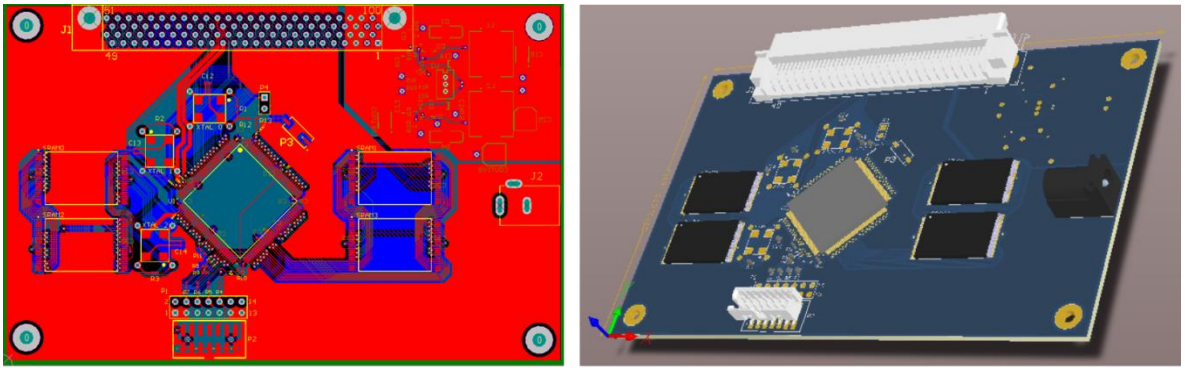
Usando el diseño jerárquico de la herramienta EDA *Altium Designer*, se adicionan a los proyectos los archivos de esquemático, previamente estructurados, para conformar el diseño de cada tarjeta de circuito impreso. En el diseño final, ha sido necesario proponer una versión alternativa con un FPGA de menores recursos, facilitando el montaje e implicando unos parámetros de menor resolución en PCB. Esto ocurre por las limitaciones técnicas de la empresa de fabricación de la placa. El FPGA seleccionado inicialmente (XC6LX45T) tiene un encapsulado CSP o BGA, que implica procesos rigurosos de montaje y que sugiere un espaciado mínimo entre trazas, vías y/o *pads*, de 3mil en sus rutas de escape (*fanout*) al interior del *stack-up* del PCB. Estos requerimientos de resolución y montaje no son comunes en las empresas de fabricación del exterior, y mucho menos en empresas nacionales, razón por la cual el prototipo diseñado, en su versión alternativa, considera una FPGA de encapsulado TQFP, cuyo montaje es más flexible en cuanto a los parámetros de espaciamiento mínimo de trazos, haciendo técnicamente más factible el proyecto PCB.

Para el diseño de la tarjeta correspondiente al analizador lógico se utilizan los diagramas esquemáticos propuestos, correspondientes a las memorias SRAM, el CPLD y los conectores respectivos para la programación del mismo, además de la interconexión con la tarjeta principal.



(a) (b)

Figura 4.32 Vista de (a) *Layout* y (b) Modelo 3D de la tarjeta con FPGA y uC desarrollada con *Altium Designer*



(a) (b)

Figura 4.33 Vista de (a) *Layout* y (b) Modelo 3D de la tarjeta con CPLD y SRAM desarrollada con *Altium Designer*

En la Figura 4.32 y Figura 4.33 se muestran las vistas del *layout* y modelos 3D de las tarjetas diseñadas con *Altium Designer*.

Así, los módulos de alimentación, FPGA y microcontrolador se embebieron en una única tarjeta de 6 capas (*Signal/Power/Signal/Signal/GND/Signal*) diseñada con soporte adicional para dispositivos de E/S, de uso frecuente en un laboratorio de sistemas digitales.

4.2.3.1 Placas de Prueba

Una vez diseñados los diagramas esquemáticos modulares de la tarjeta y teniendo en cuenta que es necesario realizar pruebas específicas del funcionamiento de cada una de las secciones funcionales de la misma, se crean proyectos PCB para cada uno de los módulos. Cuando éstos son validados, se integran al diseño global de un prototipo funcional más compacto.

4.2.3.1.1 Placa de prueba para fuentes de alimentación

Para determinar la estabilidad de las tensiones, en función de las tolerancias físicas de los elementos que conforman los circuitos de las fuentes de alimentación, es conveniente la implementación de módulos de prueba, como complemento de las simulaciones realizadas. Para hacerlo, se implementa un proyecto PCB a partir de los diagramas esquemáticos elaborados para cada una de las fuentes. Los circuitos cuentan con los reguladores conmutados LT3501 y LTC3549, que especifican en sus hojas de datos reglas para la ubicación y trazado (*routing*) dentro de un PCB que los involucre.

Para el primer módulo de prueba, se realiza un proyecto PCB que integra dos de los circuitos de las fuentes que corresponden a los que proporcionan los voltajes de 3.3V/1.8V y 2.5V/1.2V. El primer circuito se utilizaría para alimentar las funciones del CPLD, mientras el segundo lo haría con el FPGA. El diseño de PCB define un conjunto de 4 capas (*stack-up*) mostrados en Figura 4.34, necesario para cumplir con las reglas de diseño especificadas en las hojas técnicas de los reguladores integrados [100], [101].

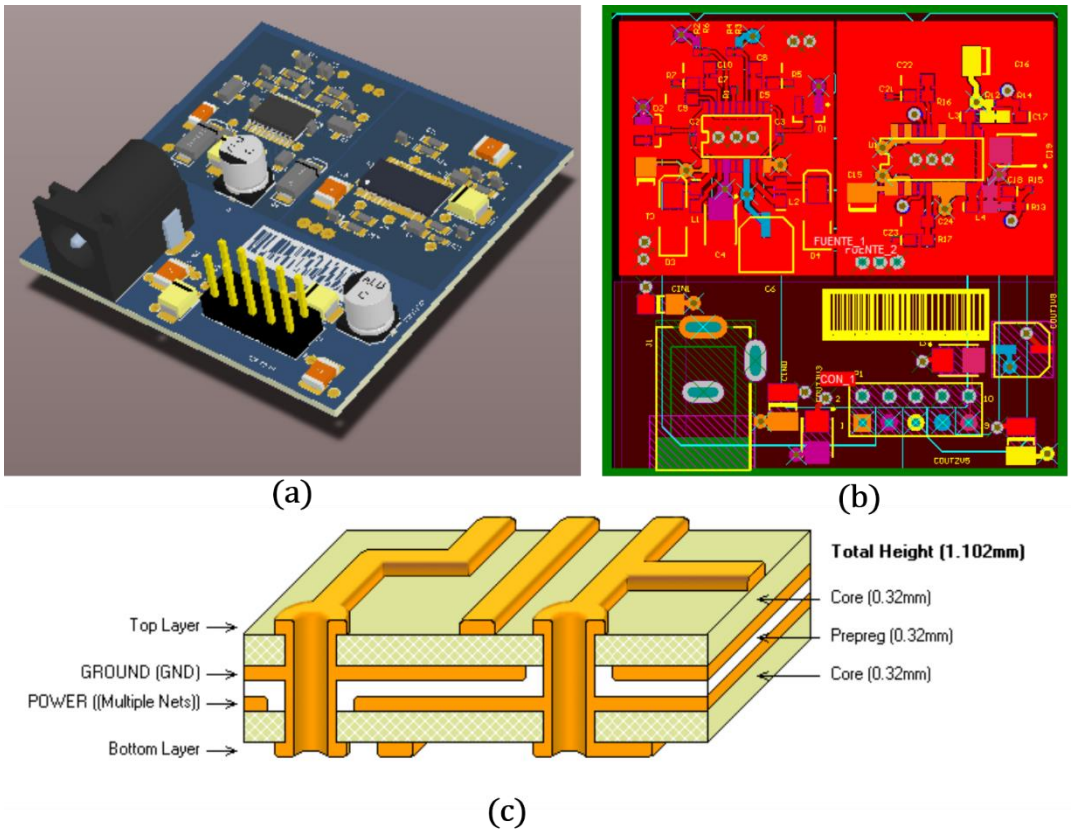


Figura 4.34 Vista del (a) Modelo 3D, (b) *Layaout* y (c) Organización de Capas

Teniendo en cuenta las limitaciones tecnológicas de las empresas del país con servicio de fabricación PCB, se diseña una tarjeta de dos (2) capas en su *stack-up*, ajustando con dificultad las reglas de diseño, pero con el fin de obtener al menos los valores nominales requeridos. Los resultados experimentales obtenidos validan los diseños esquemáticos propuestos, en cuanto a los niveles de voltaje; sin embargo, revela la necesidad de seleccionar inductores de mayor tolerancia a corriente.

4.2.3.1.2 Placa de prueba para microcontrolador

Esta placa se diseña con el objetivo de realizar pruebas de las funciones de control desde el planificador de tareas. La tarjeta cuenta con soporte para programación del microcontrolador con el dispositivo *pickit3*. El

PCB se diseña en 4 capas, de manera que se cumplan las reglas de diseño. Para el proyecto se tiene en cuenta el diagrama esquemático planteado en la Figura 4.21, en el que se plantea el uso de conectores ligados a los puertos del microcontrolador, para facilitar la realización de pruebas generales sobre esta unidad. La disposición física de los buses favorece las conexiones entre los módulos planteados y este procesador, para el caso específico de la tarjeta de entrenamiento, objeto de este trabajo.

Por razones de presupuesto y cumplimiento de cronograma, se diseña una segunda versión de la placa de control principal, considerando una tecnología de dos (2) capas y de acuerdo con las reglas esenciales de diseño de PCB. En la Figura 4.35 se muestra el modelo 3D de la tarjeta.

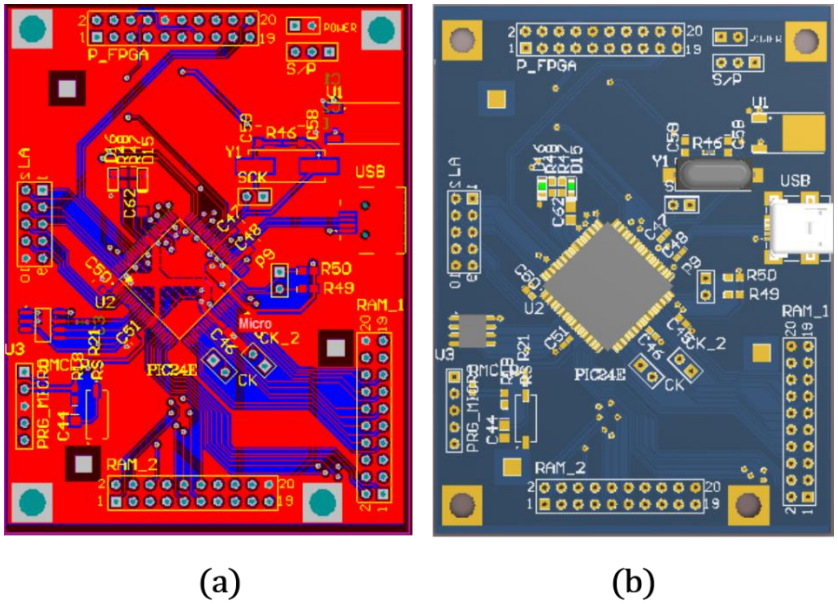


Figura 4.35 *Layout* del PCB y (b) Modelo 3D del prototipo para microcontrolador (visto con *Altium Designer*)

4.2.3.1.3 Placa de prueba para FPGA

Uno de los objetivos primordiales en la tarjeta de entrenamiento es la programación del FPGA. Por ello, el diseño propuesto soporta las interfaces de configuración: *Slave-Serial*, *SelectMAP* y como respaldo JTAG. La alimentación de este módulo es tomado a través de su respectiva interfaz de conexión a la placa de 3.3V. El encapsulado del FPGA seleccionada es de tipo CSP de 324 pines, razón por la cual deben proyectarse al menos seis (6) capas para usar el 100% de los pines de E/S. Sin embargo, las restricciones técnicas del fabricante de PCB más especializado a nivel nacional, sumado al costo sugerido por sus servicios, han obligado a diseñar una versión limitada que incluye únicamente las interfaces de programación y algunos pines de propósito general (Ver Figura 4.36).

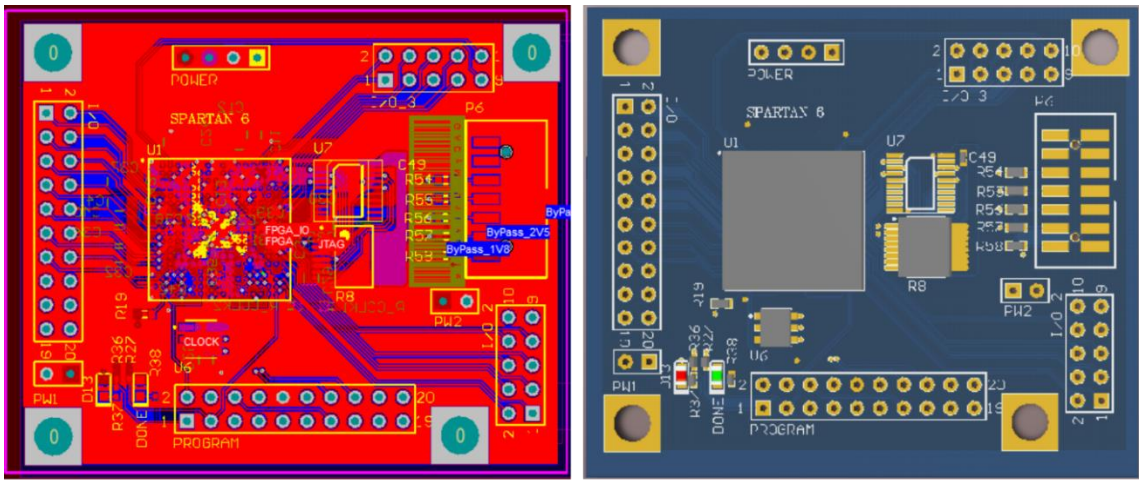


Figura 4.36 *Layout* del PCB y (b) Modelo 3D del prototipo para FPGA (visto con *Altium Designer*)

4.2.3.2 Prototipo parcial con FPGA y Microcontrolador

Una vez son verificados los diseños modulares funcionales más básicos, se integran en módulos más complejos manteniendo sus funciones esenciales.

Los módulos de alimentación, FPGA y microcontrolador se embebieron en una única tarjeta de 6 capas (*Signal/Power/Signal/Signal/GND/Signal*) (Ver Figura 4.32) diseñada con soporte adicional para dispositivos de E/S, de uso frecuente en un laboratorio de sistemas digitales. No obstante, antes de su implementación fue necesario realizar un análisis de integridad de señal en sus trazos y/o nodos más críticos. Uno de ellos corresponde a la señal de reloj de usuario, que constituye la base de temporización de los proyectos de los estudiantes y se muestra en la Figura 4.37.

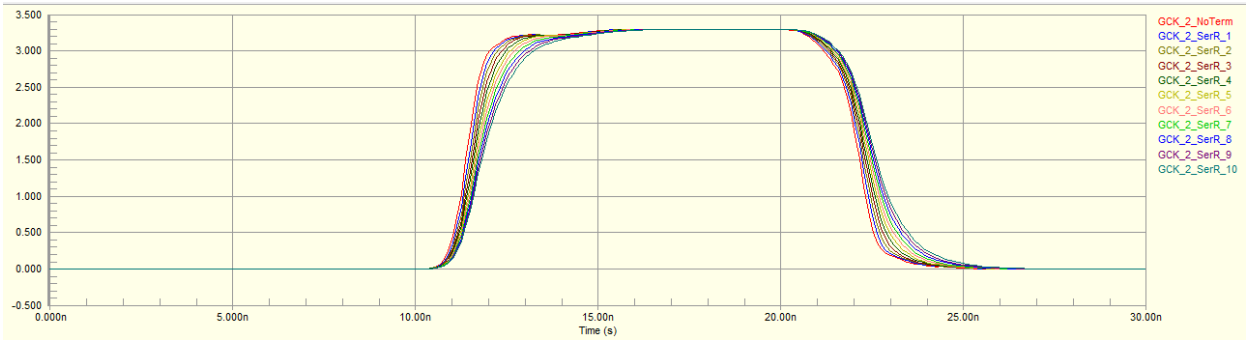


Figura 4.37 Análisis de integridad de señal para reloj de usuario

El análisis permite determinar la necesidad de resistores terminadores para la interfaz JTAG y la simetría del trazo de las líneas diferenciales para el reloj de usuario, con impedancias promedio controladas del orden de los 50 Ohm. Las reglas de diseño usadas para estas PCB se exponen en la Tabla 4.19. Los trazos con ancho superior a los 10mil demandan mayor impedancia para suministro de potencia o reloj. Así mismo, se usan separaciones y trazos de 3mil para optimizar el área bajo el encapsulado del FPGA y usar todas sus líneas a través de las capas de señal internas.

Tabla 4.19 Parámetros de Reglas de Diseño para los PCB

REGLAS DE DISEÑO	TARJETA CPLD		TARJETA FPGA-µc	
	min (mil)	max (mil)	min (mil)	max (mil)
Clearence	5,905	-	3	-
Width VCC5V0	20	-	20	-
Width VCC3V3	10	14	10	14
Width VCC1V8	10	14	-	-
Width all	5,095	10	3	15
Width GND	10	14	5	24
Width CCLK	-	-	16,881	-
Width GCK(0,1,2)	9,008	37,582	-	-
Via Diameter	15,748	-	20	-
Via Hole Size	7,874	-	10	-
Diff Pairs Routing	-	-	10	-
Plane Clearence	10	-	5	-
	Min. (ohms)	**Max. (ohms)**	-	-
Impedance GCK(0,1,2)	50	90	-	-

4.2.3.3 Prototipo parcial con CPLD y SRAM

El prototipo conformado por el CPLD y el banco de memorias SRAM (Ver Figura 4.33) debió verificarse en su capacidad de interacción la tarjeta del FPGA y el MCU. Así, el CPLD fue programado vía JTAG, con los subsistemas de comunicación y direccionamiento del analizador lógico.

El análisis de integridad de señal para la tarjeta CPLD-SRAM se centró en los buses de datos y direcciones de la unidad de memoria y la interfaz JTAG del CPLD. Según el análisis (Ver Figura 4.38), no fueron requeridos resistores de terminación para la frecuencia máxima, aunque preventivamente fueron

considerados en el diseño esquemático. Las trazas para el reloj del analizador son diferenciales, con impedancias controladas promedio del orden de 70 Ohm

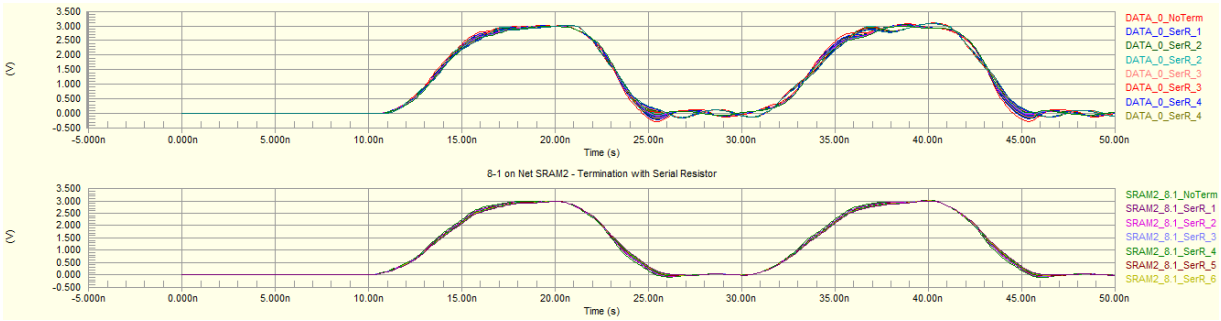


Figura 4.38 Análisis de integridad de señal para datos y direcciones de SRAM

4.3 Alternativa de Acceso Multiusuario

Como se describe en el capítulo 3, el uso de la plataforma *hardware* se limita a un solo usuario, quien dependiendo de los privilegios, debería solicitar una cita para acceder al recurso reconfigurable y analizador lógico.

El acceso simultáneo de dos o más usuario a la tarjeta de entrenamiento plantea un desafío en la asignación y administración de recursos de lógica programable de la FPGA y canales del analizador lógico para la validación en la práctica de laboratorio. Este objetivo se hace posible con el uso de la reconfiguración parcial dinámica de FPGAs, permitiendo que distintos usuarios tengan acceso el recurso programable.

4.3.1 Reconfiguración parcial de FPGA

La reconfiguración parcial es la capacidad de modificar dinámicamente bloques de la lógica, mediante la descarga de *bitstream* parciales, mientras que la lógica restante continúa funcionando sin interrupción. La tecnología de reconfiguración parcial de *Xilinx* permite a los diseñadores cambiar la funcionalidad en tiempo de operación, lo que elimina la necesidad de volver a configurar completamente y establecer vínculos, mejorando significativamente la flexibilidad que ofrecen los FPGAs. El uso de la reconfiguración parcial permite a los diseñadores reducir el consumo de potencia y mejorar la capacidad de actualización de *firmware* sobre el sistema.

Como se muestra en la Figura 4.39, la función implementada en “Bloque A” se modifica mediante la descarga de uno o varios archivos *.bit* parciales, *A1.bit*, *A2.bit*, *A3.bit* o *A4.bit*. La lógica en el diseño de FPGA se divide en dos tipos diferentes, la lógica reconfigurable y la lógica estática. El bloque etiquetado como *Bloque "A"* representa la lógica reconfigurable, mientras el área restante del FPGA corresponde a la lógica estática. Esta última permanece y no es afectada durante la carga de un archivo *.bit* parcial, pero sí lo hace con el bloque reconfigurable, que se sustituye en función de su contenido.

Hay muchas razones por las cuales la capacidad de acceder simultánea y dinámicamente a un conjunto complejo de bloques hardware, en un único dispositivo FPGA resulta ventajoso:

- Reducción del tamaño requerido de un dispositivo FPGA para una aplicación específica, con reducciones de coste y consumo de energía.

- Proporcionar la flexibilidad en la elección de los algoritmos o protocolos disponibles en una aplicación.
- Habilitación de nuevas técnicas en aplicaciones de seguridad
- La mejora de la tolerancia a fallos
- Acelerar la computación configurable

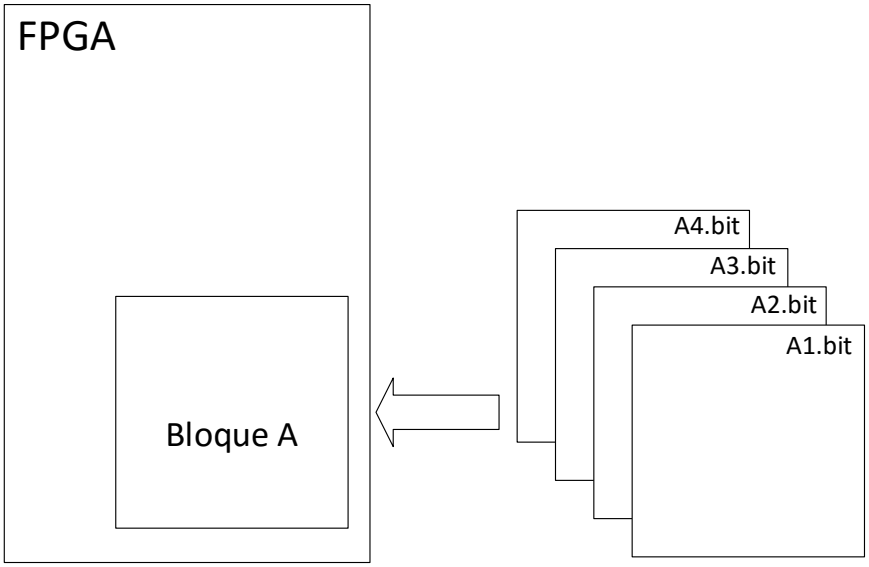


Figura 4.39 Esquema básico de la reconfiguración parcial

4.3.1.1 Terminología

La siguiente terminología es específica para la función de reconfiguración parcial y es importante conocerla para la consolidación de diseños parciales.

- **Síntesis *Bottom-Up*.** La síntesis de abajo hacia arriba es la síntesis del diseño por módulos, ya sea en un solo proyecto o varios. Requiere que una lista de conexiones separada esté escrita para cada partición, y no se realizan optimizaciones a través de estas fronteras, asegurando que cada parte del diseño se sintetiza de forma independiente. La lógica de nivel superior debe ser sintetizada con cajas negras para particiones.
- **Frames de Configuración**. Son los segmentos programables más pequeñas del espacio de memoria de configuración del FPGA. Los *frames* reconfigurables se construyen a partir de un número discreto de estos elementos.
- **Frames**. Representan la región reconfigurable más pequeña dentro de un dispositivo FPGA. El *bitstream* de tramas reconfigurables varía dependiendo de los tipos de lógica contenida dentro del *frame*.
- **Partición**. Una partición es una sección lógica del diseño, definido por el usuario en un límite jerárquico, para ser considerado en la reutilización del diseño. Una partición que se conserva mantiene no sólo una funcionalidad, sino también la aplicación idéntica.
- **Partición de Pines**. Establece las conexiones física y lógica entre la lógica estática y la reconfigurable. Los pines de particiones se crean automáticamente para todos los puertos en las particiones reconfigurables.

- **Lógica Proxy**. Es un elemento LUT1 único insertado automáticamente por el software para cada partición de pines, con excepción de las rutas dedicadas. La lógica proxy se requiere como un punto fijo conocido de interfaz entre la lógica estática y reconfigurable.
- **Lógica Reconfigurable**. Corresponde a cualquier elemento lógico que es parte de un módulo reconfigurable. Estos elementos lógicos se modifican cuando se carga un archivo .bit parcial. La mayoría de los componentes lógicos pueden ser reconfiguradas como LUTs, Flops, BRAM, bloques DSP, y IO.
- **Módulo Reconfigurable (RM).** Un módulo reconfigurable (RM) es la descripción de la lista de conexiones que se implementa cuando se crea una instancia de una partición reconfigurable. Puede haber múltiples módulos reconfigurables para una partición reconfigurable.
- **Partición Reconfigurable (RP).** Una Partición Reconfigurable (RP) es un conjunto de atributos en una instanciación que define esa instancia como reconfigurable. Las herramientas de software tales como *Ngdbuild*, *Map* y *Par* detectan el atributo de partición reconfigurable sobre la instancia y la procesan correctamente. El término partición reconfigurable se suele usar indistintamente con instancia, si ésta corresponde a una partición reconfigurable.
- **Lógica estática.** Representa cualquier elemento lógico que no forma parte de una partición reconfigurable. El elemento lógico nunca se reconfigura y siempre está activa cuando se están programando particiones reconfigurables. La lógica estática también se conoce como Lógica de Nivel Superior.

4.3.1.2 Criterios de diseño de reconfiguración Parcial

La Reconfiguración parcial (RP) en dispositivos *Spartan-6* no está sustentada en la herramienta EDA de Xilinx. Los usuarios deben comprender los siguientes requisitos antes de iniciar el diseño de un proyecto en la plataforma de entrenamiento.

- Se requiere *Floorplanning* para definir las regiones reconfigurables, según el tipo de elemento. Para mayor eficiencia, conviene alinear los *frames* con los límites definidos por las regiones de reloj, cuando sea posible.
- La síntesis de abajo hacia arriba (para crear varios archivos de listas de conexiones) y la gestión de los archivos de listas de conexiones de módulos reconfigurables son responsabilidad del usuario. Cualquier herramienta de síntesis puede ser utilizada.
- Se recomienda desacoplar la lógica para desconectar la región reconfigurable de la parte estática del diseño durante la reconfiguración parcial. Si el elemento reconfigurable es una salida del FPGA, el desacoplamiento debe ser realizado fuera del *chip*.
- No todas las opciones de implementación están disponibles para el flujo de PR, ya que algunas técnicas de implementación pueden requerir de optimización a través de todo el diseño.

4.3.1.3 Consideraciones de diseño de usuario

- La mayoría de los tipos de componentes pueden ser reconfigurados:
    - Relojes Globales y Reloj Modificación lógica deben residir en la región estática.
    - BUFG, MMCM, PLL, DCM y similares
    - Componentes de las funciones individuales de la arquitectura (como BSCAN, STARTUP, etc.) deben permanecer en la región estática del diseño
- Pueden producirse restricciones de IP, debido a los componentes utilizados para implementarlo. Los ejemplos incluyen:
    - ICON *ChipScope* (BUFG)

- Bloques de EDK con *buffers* globales
- Controlador de MIG (MMCM)

4.3.1.4 Asignación de recursos programables por usuario

El número límite de usuarios que podría acceder simultáneamente a la plataforma *hardware* está determinado por las especificaciones del FPGA. Bajo este supuesto, cada usuario podría implementar su diseño solo si se disponen de suficientes recursos en este dispositivo configurable. En este caso, se especifican las restricciones de área asignada para implementación, se enmascara con respecto al mapa global de configuración de bloques del FPGA, dado por el archivo *.bit* general, y se crea el *bitstream* de configuración parcial correspondiente.

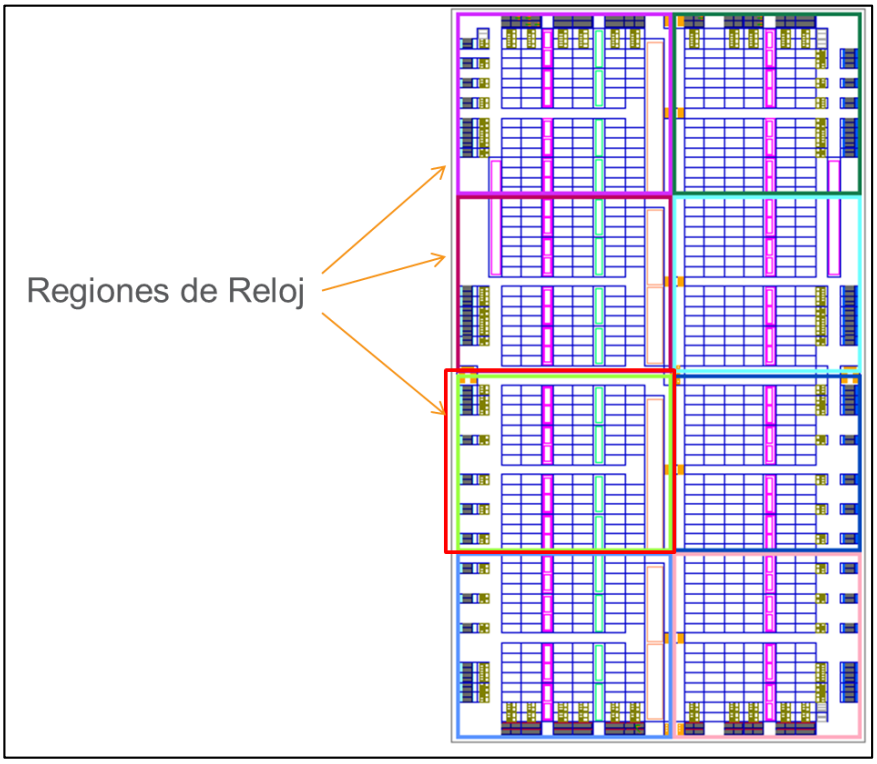


Figura 4.40 Esquema de distribución de bloques de reloj en FPGA Spartan-6

La mejor forma para dividir los recursos dentro de la FPGA es tomar las regiones de reloj (Ver Figura 4.40) con el fin de no crear conflicto con las señales de reloj de los diseños de otros usuarios. Además, se debe tener en cuenta que, hacer una segmentación de la FPGA significa reducir los recursos disponibles para que un usuario programe sus diseños. Debe considerarse también, que para la depuración solo se encuentra asociado un instrumento embebido, cuyos canales de muestreo se encuentran conectados a un conjunto de pines de E/S del FPGA, lo que limitaría el uso de este recurso dentro del proceso de diseño.

4.3.1.5 Configuración del dispositivo FPGA

En esta sección se describen las consideraciones de diseño del sistema al configurar el dispositivo FPGA con un archivo .bit parcial, así como las características arquitecturales de la FPGA que facilitan la reconfiguración parcial.

Esta sección se concentra en los detalles exclusivos de PR, pues la mayoría de los aspectos de la reconfiguración parcial no son diferentes de la configuración completa estándar.

En general, cualquiera de los siguientes puertos de configuración se puede utilizar para cargar el *bitstream* parcial: *SelectMAP*, *Serial*, JTAG, o ICAP (*Configuración interna del puerto de acceso*).

Para cargar un archivo .bit parcial mediante un modo *SelectMAP* o *serial*, algunos pines deben ser reservados para su uso después de la configuración inicial del dispositivo. Esto se logra mediante la restricción CONFIG_MODE en el *UCF* y la opción *-g persist* en el *Bitgen*.

Los *bitstream* parciales contienen todos los comandos de programación y los datos necesarios para la reconfiguración parcial. La tarea de carga de un archivo *.bit* parcial en una FPGA no requiere conocimiento de la ubicación física de la RM, porque la información de direccionamiento a los *frames* de configuración está incluida en tal archivo. No es posible que un *bitstream* parcial sea enviado a la parte equivocada del FPGA.

Un controlador de reconfiguración parcial recupera el flujo de bits parcial de la memoria no volátil y lo entrega a un puerto de configuración. La lógica de control de reconfiguración parcial puede residir en un dispositivo externo, como un microprocesador. El controlador carga el PR diseñado por el usuario en flujos de bits parciales a través de una interfaz *Serial slave* o *SelectMAP*. Como con cualquier otra lógica en el diseño estático, el circuito de control de reconfiguración parcial interna opera sin interrupción durante todo el proceso de reconfiguración parcial.

4.3.1.5.1 La descarga de un archivo completo Bit

En un sistema digital, el dispositivo FPGA se configura después de *Power On Reset* mediante la descarga de un archivo *.bit* completo, ya sea directamente desde una PROM o desde un espacio de memoria de uso general por un microprocesador. Un archivo *.bit* completo contiene toda la información necesaria para restablecer el dispositivo FPGA y configurarlo con un diseño. La Figura 4.41 ilustra este proceso.

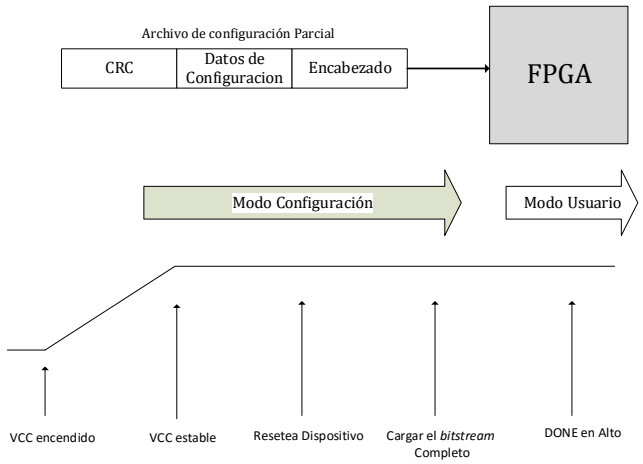


Figura 4.41 Configuración con un *bitstream* completo

Una vez finalizada la configuración inicial y verificada, el dispositivo FPGA entra en modo de usuario y el diseño descargado comienza a funcionar. Si se detecta un archivo *.bit* corrupto, la señal DONE nunca se establece, el dispositivo FPGA nunca entraría en modo de usuario y el diseño corrupto nunca comenzaría a funcionar.

4.3.1.5.2 La descarga del archivo de bits parcial

Un dispositivo FPGA parcialmente reconfigurado está en modo de usuario mientras se carga el archivo *.bit* parcial (ver Figura 4.42). Este archivo no tiene cabecera, ni hay una secuencia de inicio que lleve al dispositivo FPGA a modo de usuario. El archivo *.bit* contiene (esencialmente) sólo la dirección del marco y los datos de configuración, además de la suma de comprobación. Cuando toda la información en un archivo *.bit* parcial se envía al dispositivo FPGA, la señal DONE externa no se eleva para indicar la finalización.

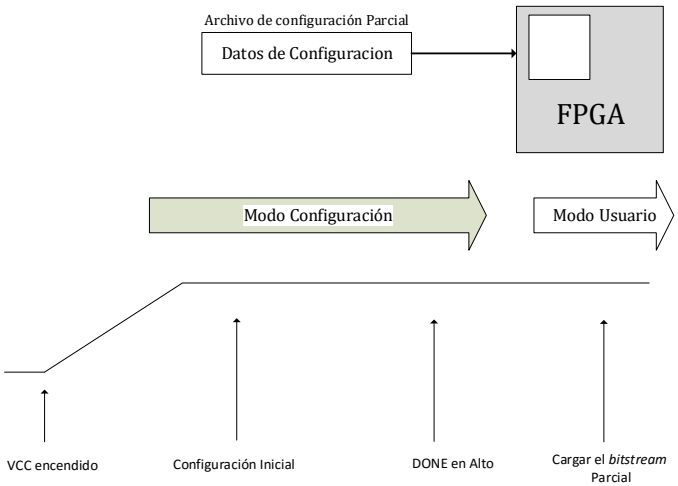


Figura 4.42 Configuración con un *bitstream* parcial

El archivo de bit parcial no tiene cabecera, ni hay una secuencia de inicio que trae el dispositivo FPGA en modo de usuario. El archivo contiene *bit* (esencialmente) sólo la dirección del marco y los datos de configuración, además de un valor final la suma de comprobación. Cuando toda la información en un archivo de *bit* parcial se envía al dispositivo FPGA, no hay señal DONE externa que se eleva para indicar la finalización.

Deben supervisarse los datos que se envían, a saber cuándo la configuración se ha completado. El final de un archivo *.bit* parcial tiene una palabra de desincronización de los servidores (0000000D) que informa al motor de configuración que el archivo *.bit* ha sido completamente entregado. Esta palabra se da después de una serie de comandos relleno, asegurando que una vez que se ha alcanzado la desincronización de los servidores, todos los datos de configuración ya han sido enviados a los marcos de destino en todo el dispositivo. Tan pronto como el archivo *.bit* parcial ha sido enviado completamente al puerto de configuración, es seguro liberar la región de reconfiguración para su uso activo.

4.3.2 Replanteamiento de tareas en el planificador

Al tener dos (2) o más usuarios usando la plataforma de entrenamiento, las funciones que se muestran en la Figura 4.43 se repetirían por cada usuario, y coexistirán múltiples tareas respondiendo a las solicitudes de los recursos. Teniendo en cuenta el modelo del sistema embebido, las tareas correspondientes a las solicitudes de usuario harán solicitudes repetidas de forma simultánea a los recursos de la tarjeta de entrenamiento. Esto implica el uso de las herramientas propias de un RTOS para la administración de recursos, como lo son los semáforos. Se debe de crear un semáforo binario para cada uno de los recursos. Es necesario evaluar el rendimiento del microcontrolador para realizar la ejecución de tareas muy

susceptibles al tiempo de ejecución, como el recibir tramas USB desde el servidor, para evitar problemas con pérdida de datos.

Figura 4.43 Estructura del sistema embebido en la tarjeta de entrenamiento

4.3.3 Propuesta esquemática

La selección del modo de configuración *SelectMAP* como una de las interfaces a soportar en el *hardware* de la placa de experimentación ha tenido una razón estratégica orientada al futuro inmediato de la plataforma, en su eventual adaptación para operación multiusuario. La configuración parcial es soportada por esta interfaz de configuración, de forma que las modificaciones que se deben hacer en el diseño de la PCB deben de ser las relacionadas con la distribución de los recursos por usuario (regiones asignadas) en la FPGA, en relación a los bloques que se pueden asignar como se muestra en la Figura 4.40. Los recursos de cada bloque asignado incluyen celdas IOB asignadas a pines de E/S. La reacomodación de estos pines debe hacerse para que cada bloque tenga la misma cantidad de interconexiones con el analizador lógico, dividiendo los 32 canales, en el caso de usar un solo dispositivo, para que el recurso de análisis se asigne a cada bloque en igual cantidad de canales. También existe conflicto para la utilización de los recursos como *Leds*, LCD y pulsadores, estos elementos también deberán de repartirse para cada bloque.

5. DISEÑO DEL ANALIZADOR LÓGICO INTEGRADO

Los analizadores lógicos han sido diseñados fundamentalmente para ayudar en la depuración de hardware a los ingenieros que diseñan y operan circuitos digitales. Para los ingenieros en formación, este tipo de instrumentos resulta esencial como apoyo a los procesos de verificación y depuración en el aprendizaje y apropiación del ciclo de diseño. En estos procesos, se requiere frecuentemente la medida e identificación de estados lógicos desde numerosos nodos y buses de circuitos digitales, de acuerdo con unos parámetros de adquisición especificados por el usuario. En esta sección, se exponen los aspectos esenciales de configuración y operación de estos instrumentos, así como los principales aportes realizados al respecto en un contexto académico. Así mismo, se presentan los detalles del diseño del analizador lógico que hace parte de la tarjeta de entrenamiento presentada en el capítulo 4. Este analizador lógico se constituye como uno de los principales aportes de este trabajo al servicio de la educación y desarrollo de competencias en ingeniería electrónica, fundamentándose en una arquitectura personalizada que involucra un subsistema de configuración para la comunicación, establecimiento de parámetros y operación del instrumento integrado; un generador de estímulos para la aplicación automática de señales en el circuito bajo prueba, de acuerdo con el *testbench* especificado por el estudiante; un circuito de disparo y sincronización, que detecta los niveles o transiciones especificadas por el estudiante para sincronizar el almacenamiento y visualización de muestras capturadas; un subsistema de muestreo, direccionamiento y almacenamiento, que organiza las muestras adquiridas y las almacena en una SRAM externa, conforme a los parámetros especificados por el usuario; finalmente, un subsistema basado en una máquina secuencial (FSM) para la generación de las señales de control, sincronización y operación del analizador lógico. La descripción de cada uno de estos bloques se presenta en este capítulo, con un soporte de gráficos, simulaciones y mediciones que muestran su concepción, diseño y operación.

5.1 Generalidades de los Analizadores Lógicos

Los analizadores lógicos surgieron como una respuesta a la evolución de los sistemas digitales y los microprocesadores, puesto que en los procesos de depuración del diseño de estos sistemas los osciloscopios son insuficientes, si se consideran aspectos como el número de entradas (canales) y las condiciones especiales de disparo. Entre las características más evidentes de los analizadores lógicos están: número elevado de canales, entradas digitales, disparo avanzado, frecuencia de muestreo y memoria de datos [107][108].

El funcionamiento de un analizador lógico se simplifica en cuatro pasos [107]: conexión, establecimiento, adquisición y análisis. El primero hace referencia a la conexión de las señales de interés del sistema bajo prueba (*SUT – System Under Test*) con los canales del instrumento. El establecimiento corresponde a la configuración de parámetros como el modo de reloj y el tipo de disparo, que definen condiciones importantes para la adquisición y análisis. El modo de reloj selecciona el tipo de adquisición entre temporización (asíncrona) o estado (síncrona), de acuerdo con la base de tiempo que se requiera para el muestreo, definida entre un reloj interno del analizador lógico o tomada desde el *SUT* respectivamente. El tipo de disparo es una de las características diferenciadoras con respecto a los osciloscopios y permite

seleccionar los datos que el analizador lógico captura con base en la evaluación de condiciones lógicas binarias. En la fase de adquisición se muestrean las líneas y buses del *SUT* conectados a los canales del instrumento de acuerdo con los parámetros configurados. La cantidad de señales que el usuario desea capturar simultáneamente está limitada por el número de canales del analizador lógico; pero el tiempo de adquisición de las mismas es función de la velocidad de muestreo y la profundidad de memoria de datos del instrumento en la que son almacenadas en tiempo real (longitud de registro). Finalmente, la información capturada y almacenada se puede mostrar en formas de onda, listados de palabras en diferentes formatos o archivos binarios durante la fase de análisis y presentación.

5.1.1 Tipos de disparo

El disparo de un analizador lógico está definido por condiciones que le permiten establecer los instantes de captura y almacenamiento de muestras; por tanto, la finalización del proceso de adquisición depende del cumplimiento de la condición de disparo. Los tipos de disparo más utilizados en una adquisición asíncrona se enumeran a continuación [109],[110]:

- Disparo por flanco (*Edge Triggering*): la detección de una transición en uno de los canales del analizador lógico suscita un evento en la adquisición.
- Disparo por tiempo de transición (*Slew-rate Triggering*): la activación del disparo está condicionada a la detección de tiempos de transición que están dentro de un intervalo predefinido. También puede condicionarse para hacerlo si el tiempo de transición ha superado este intervalo.
- Disparo por transitorio (*Glitch Triggering*): define un intervalo mínimo de ancho de pulso dentro del cual los pulsos que clasifiquen se consideran transitorios estrechos o *glitches*. El disparo se activa cuando se detecta un pulso con duración menor al definido.
- Disparo por anchura de pulso (*Pulse Width Triggering*): corresponde a un disparo generado cuando la señal del canal de interés presenta un pulso con duración entre dos valores predefinidos T_1 y T_2.
- Disparo por exceso de duración (*Timeout Triggering*): el disparo se activa cuando se detecta un pulso con duración mayor que un tiempo predefinido T en el canal seleccionado.
- Disparo por defecto de amplitud (*Runt Pulse Triggering*): la activación del disparo sucede cuando se detectan pulsos con amplitud menor que un nivel umbral predefinido.
- Disparo lógico (*Logic Triggering*): define la aplicación de una operación lógica sobre las señales de entrada. Cuando el resultado de la operación lógica está en el nivel activo se establece el disparo. También se conoce como disparo por nivel ya que las operaciones lógicas obedecen a los niveles lógicos de los canales seleccionados.
- Disparo secuenciado (*Setup-and-Hold Triggering*): el disparo se activa cuando se violan los tiempos de establecimiento (*setup*) y retención (*hold*) con respecto a una señal de referencia.

5.1.2 Adquisición

El almacenamiento de datos muestreados desde los canales del analizador lógico se realiza en función de la frecuencia de muestreo configurada, la profundidad de la unidad de memoria y el evento de disparo (*Triggering*). Este último es esencial para el control del proceso de captura y registro de muestras ya que define el cumplimiento de una condición que es de interés para el usuario en la depuración del *SUT* y sugiere la presentación de los niveles capturados en una ventana de tiempo alrededor del instante de disparo.

El proceso de adquisición en un analizador lógico puede dividirse en dos etapas: antes del disparo (predisparo – *pre-trigger*) y después del disparo (postdisparo – *post-trigger*). En la primera, se registran las muestras de datos en tiempo real y de manera continua mientras el evento de disparo no se ha presentado. De acuerdo con la frecuencia de muestreo (F_{SAMPL}) y la profundidad de memoria para los canales de

adquisición (*Memory_Depth*), puede ocurrir un desbordamiento de ésta si se supera el tiempo dado por la [Ec. 5.1]. Este suceso debe evitarse descartando las muestras más antiguas y conservando las más recientes, como en una estructura tipo *FIFO* (*First-In, First-Out*). Cuando el evento de disparo ocurre, la adquisición empieza su etapa final almacenando los datos posteriores a este evento hasta que la memoria esté llena.

$$t_{FULL} = \frac{1}{F_{SAMPL}} * Memory\_Depth \qquad \textbf{[Ec. 5.1]}$$

Se pueden establecer modos de adquisición en función del disparo y conforme a los requerimientos del usuario en la depuración del SUT. Por ejemplo, si se requiere capturar información sobre la causa de un comportamiento anormal del circuito, conviene almacenar las muestras anteriores al disparo; pero si lo importante es analizar las consecuencias de esa anormalidad, resulta más relevante una captura postdisparo [107]. Sin embargo, una posición adaptable para el disparo entre el 0% y el 100% de la longitud de registro permite al usuario ajustar el porcentaje de datos a almacenar durante el predisparo y el postdisparo (pre-postdisparo).

5.1.2.1 Muestreo Transicional (*Transitional Sampling*)

Durante una adquisición asíncrona normal, el analizador lógico almacena los datos presentes en sus canales por cada transición del reloj de muestreo. Esto no es muy favorable para el análisis de señales lentas o con pocas transiciones en intervalos largos, con respecto al período de muestreo, puesto que el registro gráfico que se presenta al usuario podría tener información poco relevante. Para optimizar el almacenamiento de los datos medido desde el SUT, muchos analizadores lógicos implementan el muestreo transicional. Este tipo de muestreo consiste en guardar en memoria sólo las transiciones, lo cual puede hacerse a través de un detector de flanco y un contador de ciclos del reloj de muestreo, que llevaría el control de la duración de los datos [109], [110]. El muestreo transicional garantizaría el almacenamiento de información esencial y la longitud de registro se vería extendida si se adquieren señales de baja frecuencia; sin embargo, para señales con frecuencias comparables con la mitad de la frecuencia máxima de muestreo es más conveniente el uso de una adquisición asíncrona normal.

5.1.3 Antecedentes principales en sistemas embebidos

Los analizadores lógicos que se distribuyen comercialmente como equipos especializados para los diseñadores digitales profesionales en la industria y algunos laboratorios de investigación son especialmente robustos y costosos. Estos instrumentos ofrecen frecuencias de muestreo hasta del orden de los Giga Hertz, usando relojes base de cientos de Mega Hertz y con arquitecturas de PC en los que hacen uso de recursos de los sistemas de cómputo. Sin embargo, los analizadores lógicos que se insertan como núcleos *hardware*/*software* funcionales o *cores* en sistemas embebidos ofrecen menores prestaciones por sus limitaciones en recursos de *hardware*, pero favorecen la portabilidad y facilitan los procesos de verificación de diseños, especialmente cuando se usan dispositivos configurables como las FPGAs y los PDLs.

Los principales reportes en la bibliografía con respecto a analizadores lógicos corresponden a patentes americanas establecidas para sistemas embebidos, como en [111] durante los años 2001 y 2004, [112] en el 2002, [113] en el 2004 y [114] en el 2008, así como desarrollos base en Colombia [115], [116] durante el 2004. La mayoría de estos diseños embebidos proponen la implementación del analizador lógico, como herramienta para la verificación de circuitos lógicos, sobre arquitecturas programables como CPLDs y FPGAs.

La arquitectura que se muestra en la Figura 5.1 corresponde a un analizador lógico embebido dentro de un PLD, que permite capturar señales lógicas antes y después de una condición de disparo (*Breakpoint*). Este esquema presentado en [111], requiere de una herramienta *software* de diseño electrónico asistido por computador (EDA) para que el usuario especifique las señales del PLD que serían monitoreadas, el *breakpoint*, el número de muestras postdisparo a almacenar, el número total de muestras a capturar y la señal de reloj del sistema. El usuario tiene que usar esta herramienta EDA para compilar y sintetizar el diseño del SUT y el analizador lógico en el mismo PLD. El *debug* y la comunicación se hacen a través de una interfaz JTAG, con base en unas celdas "estímulo" y "sensoras" que se sintetizan en el dispositivo configurable para suministrar la información de control al analizador lógico y obtener información de él respectivamente. Su arquitectura usa esencialmente tres contadores para el control del número de muestras capturadas (8 bits), el direccionamiento a memoria RAM (7 bits) y el retardo o control de muestras postdisparo (7 bits). La unidad de control principal es una máquina de estados finitos de cinco (5) estados y cuatro (4) salidas para controlar los contadores mencionados de acuerdo con las etapas de operación del analizador lógico y las señales de entrada. Las principales entradas de esta máquina secuencial corresponden a solicitudes de inicio/detención de adquisición, envío de muestras desde memoria hacia un sistema de cómputo y retroalimentación de condiciones de desbordamiento de los contadores. Finalmente, cabe mencionar que el circuito de disparo genera la señal de *Breakpoint* a través de un comparador que recibe señales de entrada y un registro con la condición de disparo.

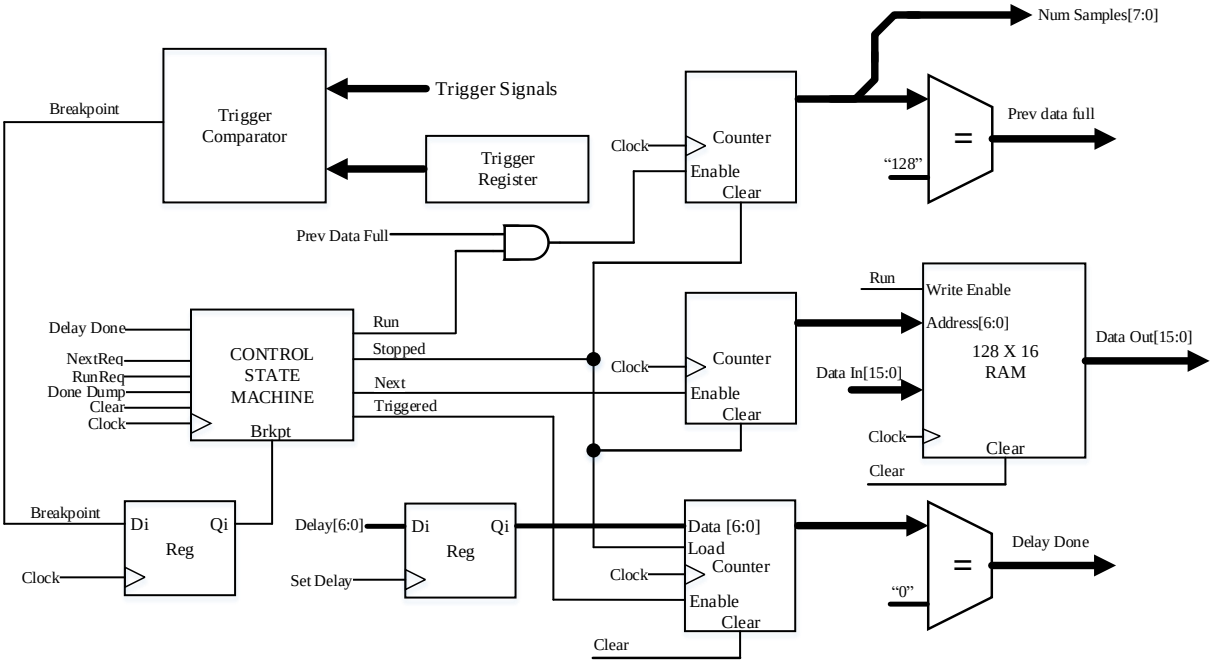


Figura 5.1 Arquitectura de analizador lógico propuesto en [111]

La arquitectura que se ilustra en la Figura 5.2 se describe en [112] con un contexto similar al presentado en [111], dado que su desarrollo en el marco de sistemas embebidos sugiere también su compilación y síntesis junto con el SUT usando una herramienta EDA; aunque algunos parámetros de configuración y operación

pueden ajustarse en tiempo de ejecución. En esta arquitectura se reduce el número de contadores a dos (2) para generar las direcciones de memoria en los ciclos de lectura y escritura respectivamente. Así mismo, se plantean dos máquinas de estado para independizar el control del disparo y la operación del analizador lógico La primera de ellas, etiquetada como "*TRIGGER STATE MACHINE*" en la Figura 5.2, está sincronizada con un reloj independiente y tiene cuatro estados: "nunca armado", "armado y esperando disparo", "disparado y ejecutando" y "completo, memoria llena". Con cada uno de estos estados se controla el proceso de adquisición y almacenamiento postdisparo en memoria, con base en las señales del comparador de disparo, el desbordamiento del contador de direcciones para escritura y el inicio de captura enviado desde la máquina de estados de modo depuración ("*DEBUG MODE STATE MACHINE*"). Esta última decodifica cuatro órdenes que se envían mediante una interfaz serial para ajustar disparo (palabra de condición de disparo), iniciar operación de captura, obtener datos de muestreo y leer estado. Obsérvese que en la Figura 5.2 este bloque tiene cuatro (4) salidas, que se activan individualmente de forma excluyente en caso de recepción de las secuencias correspondientes, y opera en función de un reloj de *debug*. Así mismo, la arquitectura de este analizador lógico incorpora un registro para almacenamiento de su estado de operación, un registro de desplazamiento y un multiplexor para la salida de datos en serie, desde memoria o desde el registro de estado. La independencia de los relojes de las máquinas de estado y sus circuitos dependientes podría introducir algunos inconvenientes de temporización por sincronización, por lo que se considera que probablemente se han omitido algunos aspectos de este tipo en la descripción de la patente.

Con respecto a los procesos de verificación de SoC (*System-on-Chip*), la incorporación de analizadores lógicos embebidos facilitaría la depuración de la actividad de buses y señales entre *cores* de procesamiento, memoria y otras unidades, debido a que la emulación apoyada en herramientas EDA a veces es insuficiente para realizarse en tiempo real. Este planteamiento es abordado en [113] con base en un bloque procesador o de procesamiento digital de señales (DSP), unidades de memoria y una unidad de captura, todos en un mismo *chip*. Su objetivo principal es conectar lógicamente el *core* DSP a una unidad de captura que usa algunos recursos de almacenamiento y de procesamiento del *chip* para su configuración y operación, de manera que obtenga datos de los ciclos de bus del procesador. Por tanto, se define un *host* y un formato de comunicación entre la unidad de captura y él, de manera que se defina qué tipo de información se capturaría mediante programación. Esto debe hacerse sin generar interrupciones al procesador para garantizar una mínima afectación al procesamiento de tiempo real. Hay que resaltar que la programación del *core* DSP y el *host* deben tener relación con la configuración de la unidad de captura, con respecto a los datos que debe adquirir y el tipo de transacción por realizar. Por tanto, la depuración que soporta está orientada a la programación con software de bajo nivel, con el propósito especial de obtener información directa de los ciclos de lectura/escritura del *core* de procesamiento dentro de un SoC.

En un contexto similar, la arquitectura propuesta en [114] sugiere su incorporación en un circuito integrado junto con el SUT, que puede ser un sistema basado en microprocesador. La motivación principal para hacerlo consiste en la complejidad que implica extraer los buses a analizar desde el interior del circuito integrado como puntos de prueba, que requieren el diseño de *pads* y canales exclusivos en la tarjeta donde se ensamble para la conexión de un analizador lógico externo. Su propuesta pretende eliminar estas interfaces mediante la adición de un bloque al interior del mismo *chip*, que permita la captura de señales con base en un patrón programable de disparo. La Figura 5.3 resume las funciones esenciales de dicha propuesta, en la que se resalta la capacidad de selección de la fuente de reloj, las señales de condición de disparo y las señales de entrada. En la figura se muestra también que el control de captura lo hace una máquina de estado en función de los circuitos de reloj y de control de disparo.

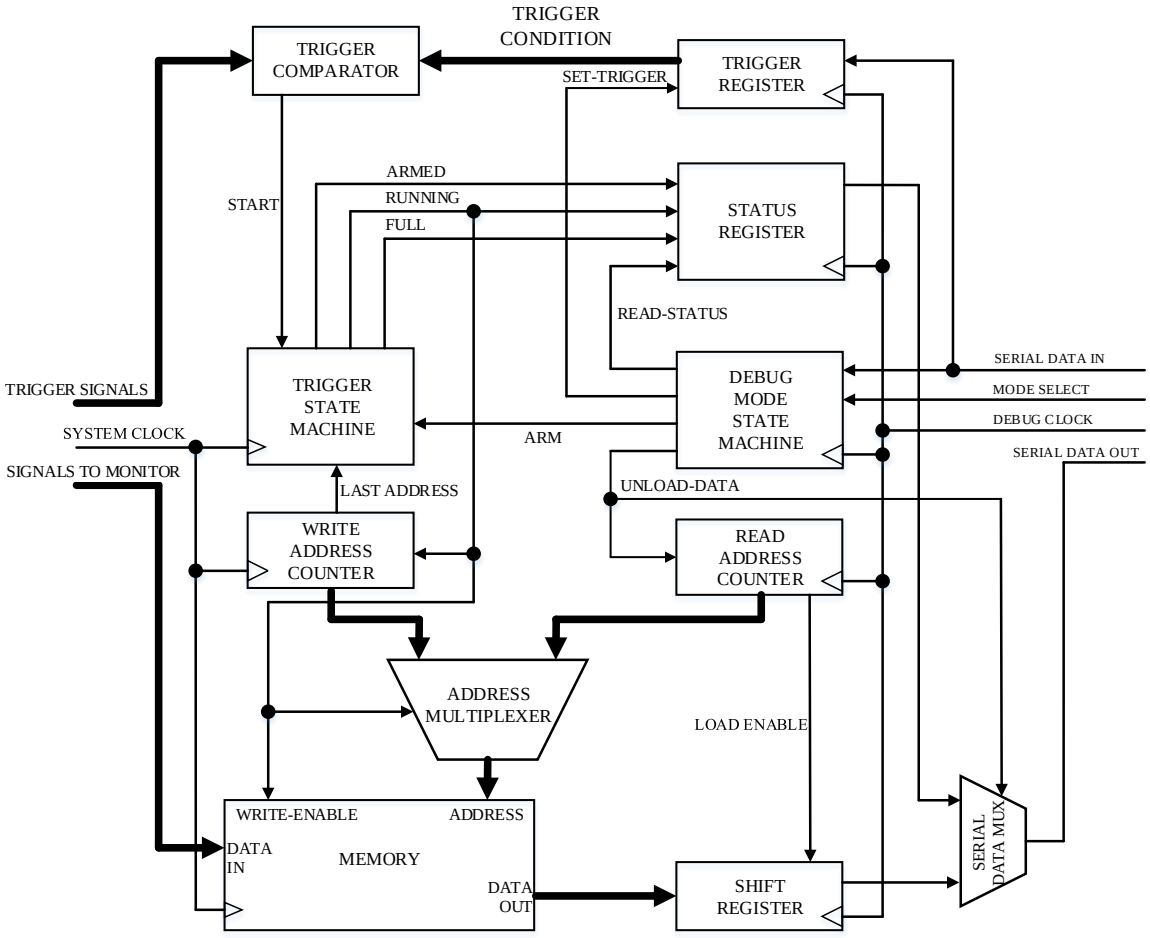


Figura 5.2 Arquitectura de analizador lógico propuesto en [112]

Debido a que los SoC incrementan cada vez más su complejidad en función de los *cores* que integren, los procesos de depuración también se hacen más complejos. Estos circuitos integrados conforman sistemas embebidos que crecen en complejidad al integrar controladores, memorias, interfaces, módulos de comunicación, entre otras unidades que tienen sus propios puertos y cuyo acceso se dificulta al estar en el mismo *chip*. El integrar una unidad de analizador lógico en este tipo de sistemas se ha consolidado debido a los requerimientos de depuración, demandando también flexibilidad en su configuración y operación. Esta tendencia se asume en [117], proponiendo una arquitectura orientada a la organización de datos de configuración y adquisición en paquetes. Para ello, se definen paquetes que faciliten el intercambio de información y se almacenan en memoria tramas con la información útil y encabezados (*headers*), que se introducen para la identificación de datos y sus unidades fuente. Además, el circuito de disparo introduce detección de transiciones y la información de las entradas pasa por un bloque de compresión basado en muestreo transicional. Aunque sería conveniente definir un *stack* o pila de procesos y protocolos para modelar el encapsulamiento de la información que se propone en [117], las solicitudes de información almacenada en memoria deben tener la garantía de extracción de la información útil y su direccionamiento.

Finalmente, el antecedente inmediato de la propuesta que se presenta en este capítulo lo establece un analizador lógico diseñado e implementado sobre un FPGA que hace parte de un kit de desarrollo comercial [115], [116]. Su interfaz de comunicación se basa en puerto paralelo, a través del cual se envían datos de

configuración hacia el analizador lógico y retornan los datos capturados, almacenados en unidades de memoria tipo FIFO. Para la adquisición se soportan las capturas predisparo, postdisparo y pre-postdisparo.

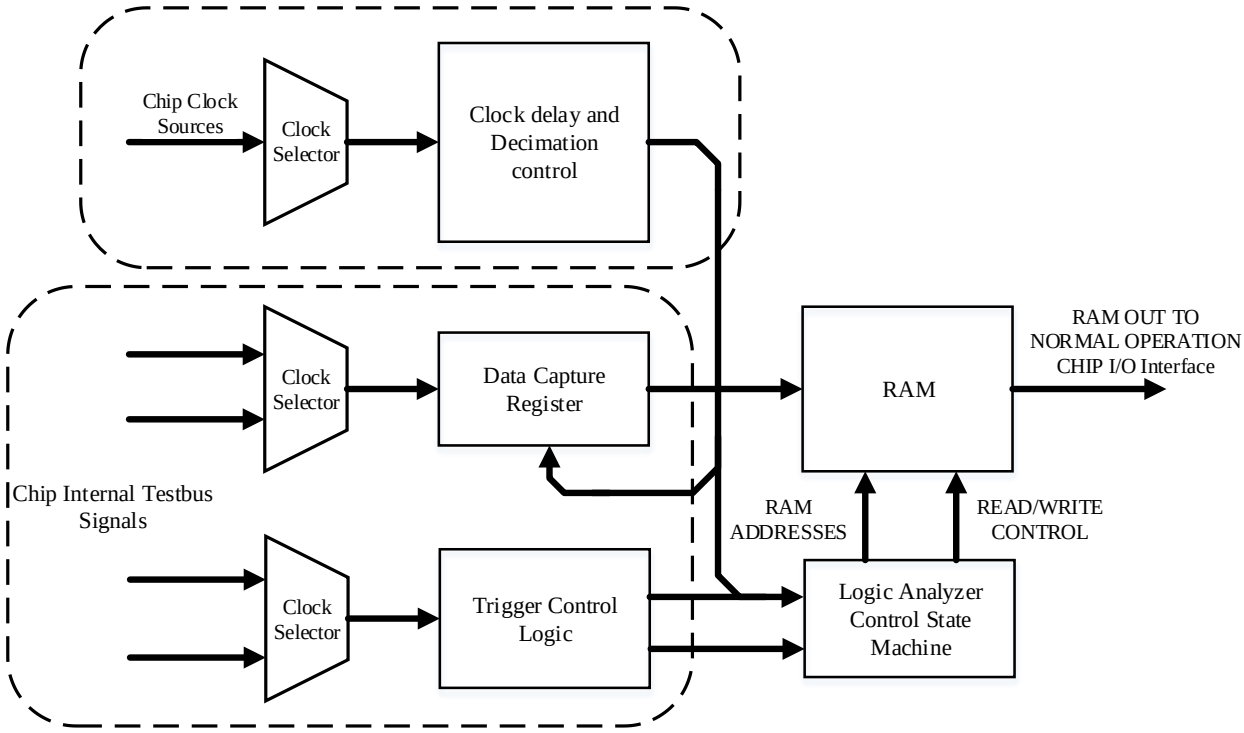


Figura 5.3 Arquitectura de analizador lógico propuesto en [114]

Con un propósito dentro del contexto educativo y considerando la plataforma hardware diseñada en el capítulo 4 y la interfaz de usuario del capítulo 3, el analizador lógico que se describe en este capítulo soporta las funciones básicas para los procesos de depuración que requiere un ingeniero en formación. Por tanto, se pueden identificar circuitos para el pre-postdisparo, unidad de memoria y un subsistema de control principal basado en una máquina de estados. Sin embargo, es importante hacer énfasis en el subsistema de configuración de este analizador lógico, que introduce una arquitectura novedosa con respecto a aquellas reportadas para sistemas embebidos y que se constituye en uno de los aportes principales de este trabajo en el contexto de los sistemas digitales. Entre estas características es importante citar: la incorporación de una estructura FIFO que almacena y facilita la aplicación de estímulos al SUT; un set de instrucciones o comandos con un formato cómodo y corto para facilitar la configuración de parámetros del analizador lógico y su operación; una unidad secuencial intérprete de estos comandos que se reciben a través de una interfaz de comunicación asíncrona de cuatro (4) fases y un banco de registros con la información de configuración y estado que constituye la base de operación del sistema de control principal del analizador lógico.

5.2 Subsistema de Configuración del Analizador Lógico

De acuerdo con las necesidades en el desarrollo de competencias en sistemas digitales, durante la fase de verificación en el ciclo de diseño, el usuario debe configurar el analizador lógico según los patrones que

desee aplicar y medir en sus circuitos de prueba. Los parámetros de configuración de este instrumento que se han identificado como los más relevantes para esta fase de la formación y que han sido considerados en el diseño del sistema propuesto son los siguientes:

- Número de canales a analizar
- Tipo de muestreo: puede ser transicional o normal
- Tipo de disparo: puede ser por nivel, flanco positivo, flanco negativo, o sin disparo (adquisición continua con detención manual)
- Frecuencia de muestreo
- Frecuencia de envío de estímulos
- Adquisición de muestras antes y después del disparo (pre-postdisparo): se establece por porcentaje de la memoria de muestreo

Desde el punto de vista de arquitectura, estos parámetros pueden ser enviados al analizador lógico para su almacenamiento en registros y garantizar su disponibilidad durante el proceso de adquisición; sin embargo, en la arquitectura propuesta se introduce un subsistema de comunicación y configuración basado en comandos, que permite el envío de cualquiera de estos parámetros en cualquier instante, incluso, durante el proceso de adquisición. Este subsistema está formado por cinco bloques principales, que consideran desde la interfaz de comunicación con un sistema externo hasta el banco de registros de configuración del analizador lógico, como se puede apreciar en la Figura 5.4. El diseño y descripción funcional de estos bloques se presenta a continuación.

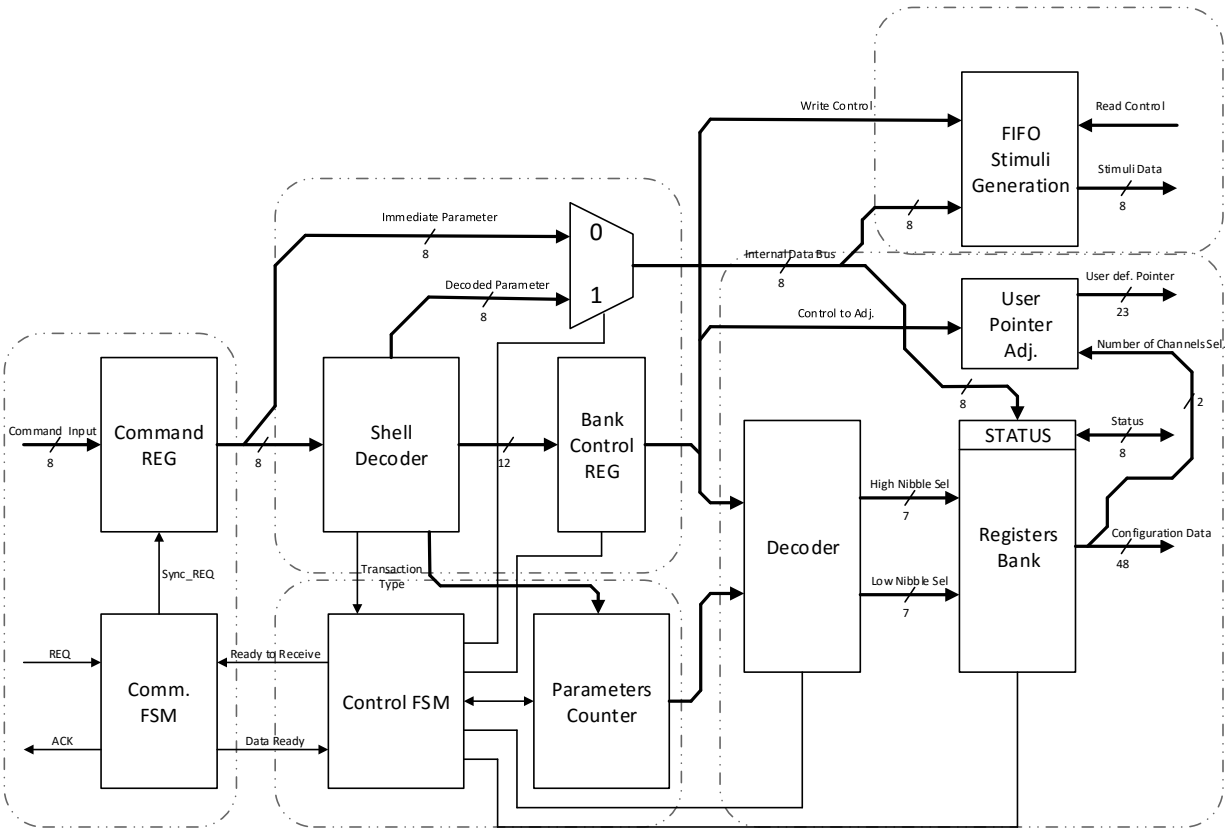


Figura 5.4 Diagrama de bloques del subsistema de configuración del analizador lógico

5.2.1 Comunicación asíncrona

El módulo propuesto para la configuración de analizadores lógicos tiene un bloque de comunicación que soporta la interfaz *hardware* para configurar el instrumento desde una MCU. Este bloque se basa en una interfaz de comunicación asíncrona que implementa el protocolo de *handshake* de cuatro fases (*4-phase handhake*) [118]. Dicho protocolo define dos líneas para control de flujo de datos, *REQ* (*Request*) y *ACK* (*Acknowledge*), y establece las siguientes etapas para el proceso de comunicación:

- La unidad maestro suministra los datos y activa la señal *REQ* (*REQ*=1).
- El receptor ve la señal *REQ* sincronizada y, si está activa, toma los datos y establece en alto su señal *ACK* (*ACK*=1).
- La unidad maestro ve la señal *ACK* sincronizada y, si está activa, desactiva su señal *REQ* (*REQ*=0).
- El receptor ve la señal *REQ* sincronizada y, si está en bajo, desactiva su señal *ACK* (*ACK*=0).

El subsistema de comunicación tiene también dos señales internas para la sincronización con el subsistema de control de la configuración del instrumento, un bus de datos interno de 8 bits que lleva comandos y parámetros sincronizados. La máquina secuencial de la Figura 5.5 es de tipo *Moore* y gobierna el control de esta etapa. Su vector de entrada está formado por las señales *RDY* (*Ready to receive* – listo para recibir) y *REQ* [*RDY, REQ*], mientras la salida es conformada por *DATA_READY* (*Data Ready* – Dato listo y disponible) y *ACK* [*DATA_READY, ACK*]. Durante el estado "*WAITING*", la señal *ACK* se mantiene en

bajo (*ACK*=0) mientras *REQ* también lo esté (*REQ*=0), esperando a que se solicite una transacción de datos sobre el bus de entrada de comandos *Command Input*. Cuando *REQ* es activada por la unidad maestro externa, la máquina entra en el estado "*PROCESSING*" y establece la señal *DATA_READY* para sincronizar el subsistema de control y procesar el dato entrante. Una vez esta unidad lo procesa, activa la señal *RDY* para que la máquina conmute a "*FINISHING*", establece *ACK* en alto, e informa que el *REQ* ha sido atendido y el dato se ha recibido. Este estado se mantiene mientras *REQ* permanece activo y la transición a "*WAITING*" ocurre cuando la unidad maestro desactiva esta señal (*REQ*=0), finalizando el ciclo de recepción de datos.

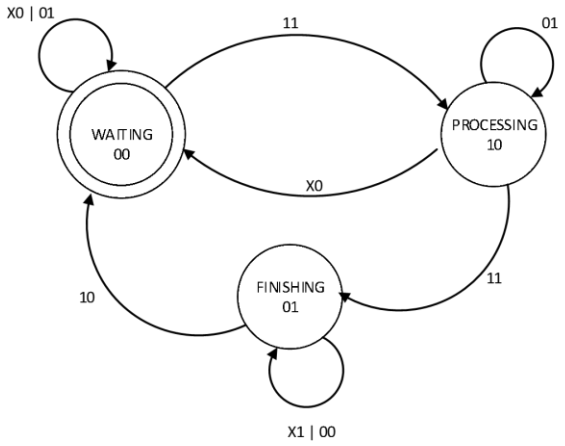


Figura 5.5 Diagrama de estado para la máquina secuencial del bloque de comunicación asíncrona

5.2.2 Configuración basada en comandos

La arquitectura propuesta para el subsistema de configuración del analizador lógico se basa en un conjunto de comandos que facilitan su operación. El formato de estos comandos y sus funciones se muestran en la Tabla 5.1, en la que se define también el tipo de transacción: tipo 0, cuando no se envían parámetros adicionales sobre el comando o están incluidos en el mismo byte, y tipo 1 en aquellas que implican transacciones adicionales para el envío de los parámetros.

Los comandos se reciben en un registro de 8-bit, que es habilitado por la señal *REQ* sincronizada. Cada comando recibido es decodificado por un bloque combinatorio "*Shell decoder*" que suministra las señales adecuadas de control y datos a otros bloques y/o etapas del sistema. Este bloque entrega una palabra de 12 bits para el control y decodificación del banco de registros de configuración, el cual se carga en paralelo sobre el registro de control de banco *Bank Control REG*; los bits utilizados se describen en la Tabla 5.2. El tipo de transacción también es decodificado y enviado a *Control FSM* en el subsistema de control, también la señal de selección de carga para el contador de parámetros adicionales se envía al control de un multiplexor. Un bus interno de 8 bits se establece a partir de este bloque de decodificación, en el que se ubican separadamente los datos inherentes a algunos comandos tipo 0, correspondientes a parámetros que serían almacenados en el banco de registros de configuración (*Registers Bank*); los parámetros adicionales enviados en comandos de tipo 1, que pueden ir a la cola de estímulos (*FIFO Stimuli Generation*) o al banco de registros. La selección de estos datos para el bus interno se hace a través de un multiplexor controlado desde el registro.

Tabla 5.1 Descripción de comandos

Comando	Significado	Tipo	Parámetros adicionales	Código de operación
CTRG	*Configure trigger* – Configurar disparo	1	1-byte para el registro CONF_TRIG	0001
TRGW	*Trigger Word* – Palabra de disparo	1	4-bytes para la palabra de disparo (32-bit)	0010
SFREQ	*Sample frequency* – Frecuencia de muestreo	0	-	0011
CHSAMP	*1-bit channels to sample* – Canales de 1-bit a muestrear	0	-	0100
STM2SND	*Stimuli to send* – Estímulos a enviar	1	256-bytes para estímulos	0101
STARTAQ	*Start Acquisition* – Iniciar adquisición	0	-	0110
STOPAQ	*Stop Acquisition* – Detener adquisición	0	-	1000
PNTADJ	Pointer Adjustment – Ajuste de puntero para almacenamiento pre-post disparo	0	-	0111
RST	*Reset system* – Reiniciar sistema	0	-	1001
RDSMEM	*Read sample SRAM* – Leer SRAM de muestreo	0	-	1010

Formato de comando

Dato - parámetro		*Código de operación del comando*	
7	4	3	0

Tabla 5.2 Descripción del registro de control de banco *Bank Control REG*

Bit	Function
H_REG (3-bit)	Selecciona el *nibble* más significativo de un registro del banco
L_REG (3-bit)	Selecciona el *nibble* menos significativo de un registro del banco
Stim_LD	Habilita el almacenamiento en la cola de estímulos
PTR_LD	Habilita la carga del puntero a memoria de muestreo de acuerdo con el número de canales a muestrear.
SMEM	Habilita la lectura de datos desde memoria de muestreo
DBus_Sel	Selecciona el dato para el bus interno de datos, puede ser del registro de comando o del bloque de decodificación "*Shell decoder*".
EN_Sel	Selecciona la fuente de decodificación del banco de registros, puede ser del registro de control de banco o del contador de parámetros adicionales
Dec_EN	Habilita el decodificador del banco de registros.

BNK_CTRL_REG

H_REG	L_REG	Stim_LD	PTR_LD	SMEM	DBus_Sel	EN_Sel	Dec_EN
11 9	8 6	5	4	3	2	1	0

5.2.3 Máquina de estados para el subsistema de control

Este bloque (*Control FSM*) se constituye como el más importante del subsistema de control al manejar el flujo de las señales de datos y control, de acuerdo con el tipo de comando y/o transacción, con una máquina secuencial de estados finitos (FSM). Esta máquina consta de cinco (5) estados: *WAITING4COMM*, *LOAD_CTRL*, *DATA_COUNTING*, *SIMPLE_COMM* y *REGISTERING*. Mientras ningún comando haya llegado, la FSM se mantiene en el estado *WAITING4COMM*, en el que espera por un nuevo comando o parámetro adicional. Cuando un comando tipo 1 ocurre, se alcanza el estado *LOAD_CTRL*, durante el que se carga el número de parámetros adicionales en un contador descendente, que a su vez informa a la FSM para leer el siguiente dato, bien sea como parámetro o como comando. Si se recibe un comando tipo 0, la transición es al estado *SIMPLE_COMM*, en el que se almacena información de control decodificada del comando recibido y se separan los datos implícitos, si los hay. El estado *REGISTERING* es para sincronizar el almacenamiento en los registros de configuración, antes de retornar de nuevo a la espera de parámetro o comando por defecto (*WAITING4COMM*). El diagrama de estado de la FSM se muestra en la Figura 5.6.

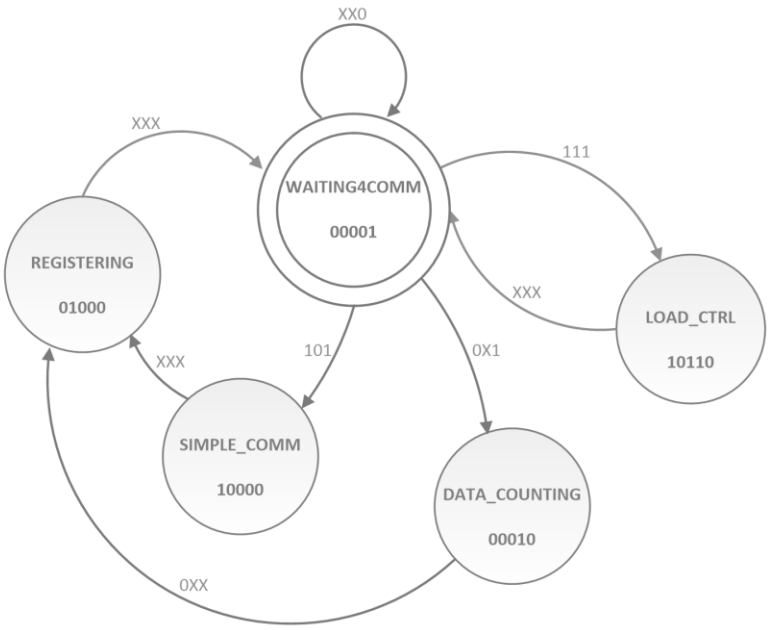


Figura 5.6 Diagrama de estado para el subsistema de control

Control FSM tiene tres (3) entradas: la más significativa es la señal generada por el contador de parámetros, en la que indica si su conteo ha llegado a cero; la siguiente corresponde al tipo de transacción o comando y la menos significativa es la señal *DATA_READY*, que es emitida desde el bloque de comunicación *Comm FSM*, indicando cuando un parámetro o comando está listo para ser decodificado.

Las salidas de *Control_FSM* son cinco: el bit más significativo habilita el registro de control *BNK_CTRL_REG* cuando llega un nuevo comando, el bit 3 sincroniza la habilitación de los decodificadores del banco de registros y la configuración del puntero de direcciones a memoria de muestreo; el bit 2 permite la carga del número de parámetros adicionales al contador correspondiente, que es previamente seleccionado a través de un multiplexor 4-1 en el bloque de decodificación *Shell decoder*. El bit 1 de la salida de la FSM habilita el decremento en uno del contador de parámetros, mientras el bit menos significativo indica la disponibilidad de este subsistema de control al bloque de comunicación *COMM_FSM*.

5.2.4 Banco de registros de configuración

La información de control para la configuración se almacena en un banco de siete (7) registros de 8-bit, que son escritos cada vez que un comando de configuración es decodificado y se extraen los parámetros. Por cada registro se puede acceder a su contenido de un *byte* o a cada uno de sus *nibbles* de manera independiente. El registro *BNK_CTRL_REG* tiene dos campos de 3-bit, que se muestran en la Tabla 5.2 como *H_REG* y *L_REG*, los cuales se usan para seleccionar el *nibble* alto o el *nibble* bajo de uno de los registros del banco. La decodificación se realiza a través de un decodificador 3-8, que es habilitado por los bits *EN_Sel* y *Dec_EN*. Los detalles del banco de registros se presentan en la Tabla 5.3.

Tabla 5.3 Descripción del banco de registros

Registro	Dirección	Función	Comando relacionado
TRIG_COND_WORD 32-bit (4 bytes)	0x03~0x00	Esta palabra de 32-bit define la palabra de condición de disparo, cuando un disparo por nivel es configurado	TRGW
CONF_TRIG	0x04	Define los parámetros principales del disparo	CTRG
AQ_FORMAT	0x05	Define los parámetros principales de la adquisición	SFREQ, CHSAMP
STATUS	0x06	Contiene las banderas relacionadas con el proceso y estado actual del analizador lógico	STARTAQ, STOPAQ

La configuración del disparo para el analizador lógico usa las cinco direcciones más bajas del banco de registros. Parámetros como el tipo de disparo (2-bit), posición del bit para disparo por flanco (5-bit) y tipo de muestreo (1-bit) son almacenados en *CONF_TRIG* (Ver Figura 5.7). Cuando se selecciona un disparo por nivel, se tiene que almacenar una palabra de condición de 32-bit en el registro *TRIG_COND_WORD*.

La frecuencia de muestreo y el número de canales de entrada son también parámetros de adquisición para el analizador lógico y son almacenados en el registro *AQ_FORMAT*. El primero de ellos reside en el *nibble* bajo, mientras la cantidad de canales de 1-bit a muestrear está en el *nibble* alto del registro (Ver Figura 5.7).

Los comandos que inician y detienen el proceso de adquisición modifican la parte baja del registro *STATUS*, cuyos bits constituyen las entradas principales para controlar el analizador lógico. Así mismo, el comando

de lectura de memoria modifica particularmente el *nibble* más alto, estableciendo el bit RDF (Ver Figura 5.7). No obstante, el subsistema de control principal del analizador lógico puede cambiar el *nibble* bajo del registro *STATUS* para reiniciar el bit *STRTAQ*, cuando haya terminado el proceso de adquisición.

CONF_TRIG

7	6	5	4	3	2	1	0
SAMP_TYP	*EDGB [4]*	*EDGB [3]*	*EDGB [2]*	*EDGB [1]*	*EDGB [0]*	*TRGTYP [1]*	*TRGTYP [0]*

SAMP_TYP: Tipo de muestreo

0: Muestreo normal continuo **1**: Muestreo transicional

EDGB[4:0]: Posición del canal (0 to 31) a considerar para disparo por flanco

TRGTYP[1:0]: Tipo de disparo

00: Continuo **01**: Nivel **11**: Flanco de subida **10**: Flanco de bajada

AQ_FORMAT

7	6	5	4	3	2	1	0
		CHNMB [1]	*CHNMB [0]*	*SFRQ [3]*	*SFRQ [2]*	*SFRQ [1]*	*SFRQ [0]*

CHNM[1:0]: Número de canales a muestrear

00: 32-bit **01**: 16-bit **10**: 8-bit

SFRQ[3:2]: Frecuencia de aplicación de estímulos (Basada en frecuencia del Sistema *Fosc*)

SFRQ[1:0]: Frecuencia de muestreo (Basada en frecuencia del Sistema *Fosc*)

STATUS

7	6	5	4	3	2	1	0
			RDF		*#WR*	*STATF*	*STRTAQ*

RDF: Bandera que indica que el usuario ha activado un proceso de lectura de la memoria de muestreo

#WR: Indica que hay un proceso de escritura activo sobre la memoria de muestreo. Este bit es activo bajo.

STATF: Indica que el analizador lógico está en el estado *TRIGGERED* o *READING*. Junto con el bit *#WR* indican el estado exacto post-disparo en que se encuentra.

Figura 5.7 Detalles de los registros *CONF_TRIG*, *AQ_FORMAT* y *STATUS*

5.2.5 Generador de estímulos y memoria de muestreo

Los analizadores lógicos son implementados como módulos independientes o instrumentos, pero la mayoría de aplicaciones educativas requieren el suministro de señales (*estímulos*) que alimenten las entradas de los circuitos bajo prueba de acuerdo con los requerimientos de usuario.

Una estructura tipo *FIFO* se usa en este sistema para almacenar los estímulos enviados por el usuario y cargarlos a las entradas seleccionadas del circuito digital bajo prueba. Esta *FIFO* es asíncrona por razones de temporización, ya que pueden ser usados distintos relojes en sus circuitos de lectura y escritura.

El bloque *FIFO* tiene dos puertos de 8-bit para la entrada y salida de datos. La entrada de estímulos es suministrada por el bloque *Shell decoder*, desde la interfaz de usuario hacia el bus de datos interno. Su

escritura se habilita a través del bit *Stim_LD* del registro *BNK_CTRL_REG* y el bit 1 de *Control_FSM*. El circuito de salida de esta estructura debe ser manejado por el módulo de control principal del analizador lógico, de acuerdo con *STATUS*.

La memoria de muestreo es indispensable en los analizadores lógicos para almacenar las muestras capturadas, pero su direccionamiento debe estar de acuerdo con los parámetros de disparo. En el diseño propuesto, la dirección al inicio de la adquisición está configurada por defecto sobre la base de la mitad más alta del espacio total de direccionamiento de la memoria de muestreo. Sin embargo, el tamaño de este espacio es de acuerdo con el número de canales configurados para muestrear. La máxima dirección de memoria es la más alta del espacio de direccionamiento total, cuando sea ajustado el número de canales más pequeño posible (8-bit). La Tabla 5.4 muestra el rango de direcciones para cada número de canales seleccionado, así como el puntero de inicio a memoria de muestreo establecido por defecto.

Tabla 5.4 Puntero de inicio por defecto a memoria de muestreo

Número de canales	Rango total de direccionamiento	Puntero de inicio por defecto
32-bit	0x000000~0x1FFFFF (2MB)	0x100000
16-bit	0x000000~0x3FFFFF (4MB)	0x200000
8-bit	0x000000~0x7FFFFF (8MB)	0x400000

El puntero de inicio por defecto debe establecerse de acuerdo con los bits *CHNMB[1:0]* del registro *AQ_FORMAT*. Estos bits seleccionan un multiplexor 4-1 de 3-bit, cuyas entradas son los tres bits más significativos de las direcciones por defecto para cada caso de los presentados en la Tabla 5.4. Sus bits de salida corresponden a las tres entradas más significativas de precarga del contador de direcciones, que tiene sus entradas más bajas fijas en cero (0).

Cuando un comando *PNTADJ* es enviado al analizador lógico, se incluye en su formato un bit para especificar el sentido en el que el contador de direcciones debe avanzar, sea ascendente o descendentemente, de manera que se ajuste el puntero a memoria de muestreo de acuerdo con la especificación porcentual de muestras post-disparo dada por el usuario y que el analizador debe capturar.

El bit *PTR_LD* (*Pointer Load*) de *BNK_CTRL_REG* se activa, originando que en el estado de *REGISTERING* de *Control_FSM* se habilite el contador y sea efectivo el ajuste del puntero requerido por el usuario.

5.3 Subsistema de Direccionamiento a Memoria de Muestreo

La generación de direcciones para acceder a la memoria de muestreo obedece a un circuito secuencial, que basado en un conteo progresivo, avanza conforme a una serie de parámetros, algunos de ellos establecidos por el usuario como son: el estado actual del analizador lógico; sus circuitos de disparo; la frecuencia de muestreo; el número de canales a muestrear y el ajuste del puntero dado por la cantidad de memoria definida para el post-disparo.

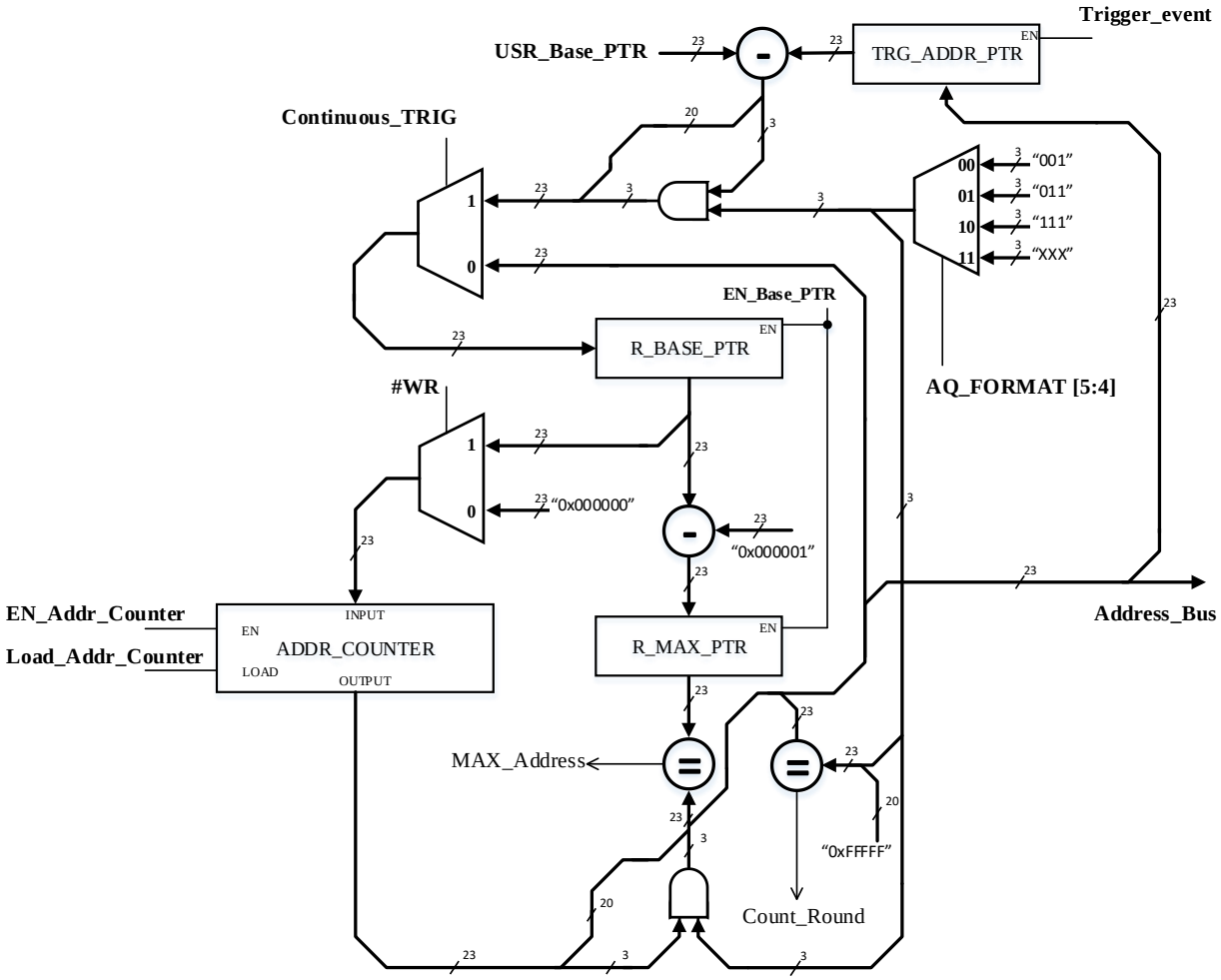


Figura 5.8 Diagrama en bloques lógicos del subsistema de direccionamiento

En la Figura 5.8 se muestra la conformación lógica del subsistema de direccionamiento a la memoria de muestreo. En esta ilustración se pueden identificar las siguientes señales de entrada:

- **USR_Base_PTR**: dirección de 32-bit ajustada por el usuario como base para el almacenamiento post-disparo. En la aplicación Web, este parámetro se indica en términos porcentuales que el microcontrolador traduciría en términos de comandos *PTRADJ*.
- **AQ_FORMAT[5:4]**: uno de los registros de configuración que define la frecuencia de muestreo y el número de canales a usar en el proceso de adquisición. En este subsistema se establecen como entrada los bits 5 y 4 de este registro, los cuales definen el número de canales a muestrear y el espacio total de direccionamiento.
- **Trigger_event**: señal sincronizada con el reloj del subsistema de direccionamiento que indica cuando una condición de disparo se ha alcanzado.
- **Continuous_TRG**: señal que indica cuando el usuario ha seleccionado una adquisición continua. Esta señal se obtiene de *CONF_TRIG*[*TRGTYP1*: *TRGTYP0*] (Ver Figura 5.7), activándose cuando ambos bits están en bajo.

- **EN_Base_PTR**: señal que habilita la carga de los registros puntero base y máximo (*R_BASE_PTR*, *R_MAX_PTR*), de manera que se actualicen en los instantes seguidos al evento de disparo.
- **EN_Addr_Counter**: señal enviada por el sistema de control principal del analizador lógico para habilitar el incremento del puntero a memoria de muestreo.
- **Load_Addr_Counter**: señal que permite la carga de entradas del contador de direcciones, ajustando el puntero en un valor que depende del ciclo que esté cumpliendo el analizador sobre la memoria (lectura o escritura).
- **#WR**: señal que define si el analizador se encuentra escribiendo o leyendo la memoria de muestreo (#WR=0 para indicar escritura sobre SRAM).

Las salidas del subsistema de direccionamiento son tres:

- **Address_Bus**: es la salida principal u objetivo de este bloque, pues corresponde al bus de direcciones de 23-bit que recorre el espacio global de direccionamiento. Este bus se conecta a la lógica de decodificación de bancos de memoria y a las propias unidades de memoria.
- **MAX_Address**: señal de advertencia que indica cuándo el contador de direcciones ha alcanzado la máxima dirección, de acuerdo con el puntero base, el puntero de disparo y el espacio post-disparo definido por el usuario.
- **Cont_Round**: señal de advertencia que se activa cada vez que se alcanza el máximo conteo binario en el contador de direcciones, en función del espacio de direccionamiento establecido de acuerdo con el número de canales a muestrear. Así, para una adquisición de 32 canales, esta señal de advertencia se activaría cuando el bus de direcciones alcance la dirección 0x1FFFFF; para 16 canales cuando se alcance 0x3FFFFF y para 8 canales cuando se tenga 0x7FFFFF.

En la arquitectura propuesta con la Figura 5.8 se presentan tres registros de tipo puntero de 23 bit: *TRG_ADDR_PTR*, *R_BASE_PTR* y *R_MAX_PTR*. El primero de ellos almacena la dirección apuntada por el bus de direcciones cuando ocurre un evento de disparo, razón por la cual su habilitación está definida por la señal *Trigger_event*. El segundo puntero tiene la dirección base, desde donde se almacenan las muestras correspondientes a la captura pre-disparo del analizador lógico. El tercer puntero almacena la dirección máxima que debe alcanzar el bus después del evento de disparo.

El contador binario *ADDR_COUNTER* es de 23-bit y genera las direcciones absolutas. Según la Figura 5.8, los 20-bit menos significativos del bus que se obtienen de este contador se consideran como las 20 líneas más bajas del bus de direcciones que se conectarían directamente a las entradas de las unidades de memoria. Los tres bits más significativos de la dirección absoluta se enmascaran, mediante una operación AND, de acuerdo con el número de canales que el usuario ha seleccionado y obedeciendo los rangos válidos de direccionamiento que se deben garantizar de acuerdo a la Tabla 5.4. Para esto, se usa un MUX 4-1, cuyas entradas tienen las máscaras a aplicar según *AQ_FORMAT[CHNMB1:CHNMB0]* (Ver Figura 5.7 y Figura 5.8). La salida de la operación AND junto con los 20-bit más bajos de la salida del contador, conforman el bus de direcciones (*Address_Bus*). En 6.2.4.1.1 se define la interfaz con la memoria externa de muestreo y se expone la conformación de las unidades de 1Mx16.

Un MUX 2-1 de 23-bit selecciona la dirección que debe cargarse a *R_BASE_PTR*, mediante la señal *Continuous_TRG*, de manera que cuando el usuario ha seleccionado un disparo por nivel o flanco se cumpla la ecuación [Ec. **5.2**]. En esta expresión, *Mask* resulta de la concatenación de los 3-bit de salida del MUX 4-1, en la parte más significativa, y 0xFFFFF. El valor que se carga sobre el registro es, en este caso, la diferencia entre el puntero especificado por el usuario y la dirección en la que ocurre el evento de disparo, lo que corresponde a un desplazamiento del primero en función de este evento. Esto se hace para garantizar la continuidad en el almacenamiento de las muestras alrededor del disparo, actualizando el puntero base de

manera que direccione la primera muestra de la sección porcentual de memoria definida por el usuario para el pre-disparo; por tanto, este registro apunta a la primera muestra que debe ser enviada al servidor en el proceso de lectura de la memoria de muestreo.

$$R\_BASE\_PTR = Continuous\_TRG \cdot (USR\_Base\_PTR - TRG\_ADDR\_PTR) \cdot Mask + \overline{Continuous\_TRG} \cdot Address\_Bus \quad \textbf{[Ec. 5.2]}$$

Cuando el usuario ha seleccionado una adquisición continua, *R_BASE_PTR* se actualiza permanentemente con el valor del bus de direcciones, como si se generase un disparo por cada instante de muestreo. Sin embargo, dado que el valor cargado en *R_MAX_PTR*, como lo muestra [Ec. **5.3**], depende del puntero base y señaliza la finalización del recorrido por el espacio de direccionamiento, el usuario debe terminar el proceso de adquisición mediante el envío de una instrucción *STOPAQ*.

$$R\_MAX\_PTR = R\_BASE\_PTR - 1 \quad \textbf{[Ec. 5.3]}$$

5.3.1 Decodificación de direcciones y organización de datos

El espacio total de almacenamiento del sistema es de 8MB y se implementa mediante cuatro unidades SRAM de 2MB (*IS61WV102416BLL*) externas al módulo analizador lógico, como se indica en la Figura 5.9. En la sección 6.2.4.1.1, se describen las características más relevantes de estas unidades y se presenta la tabla de decodificación (Tabla 6.4), que relaciona el número de canales seleccionado por el usuario con los tres bit más significativos del bus de direcciones, para generar las señales de control y habilitación a cada una de las unidades de memoria. El decodificador de direcciones considera este parámetro, obtenido del registro *AQ_FORMAT* para habilitar los bancos de memoria por Word o por Byte, estableciendo los rangos de direcciones presentados en la Tabla 5.4. Esto implica que al seleccionar 32 canales, el rango comprendido entre 0x200000 y 0x7FFFFF queda por fuera del espacio de direccionamiento. Cuando son 16 canales de adquisición, el rango no implementado es de 0x400000 a 0x7FFFFF; mientras que si se seleccionan 8 canales, se utiliza todo el espacio de memoria. Así, durante un proceso de muestreo normal, el espacio ocupado en memoria por cada muestra es directamente proporcional al número de canales seleccionado, lo que significa que cuando éste es más grande se requieren más bits por posición de memoria y se alcanzaría su ocupación total en una dirección más baja comparado con un menor número de canales. En la Tabla 5.5 se puede apreciar el comportamiento descrito.

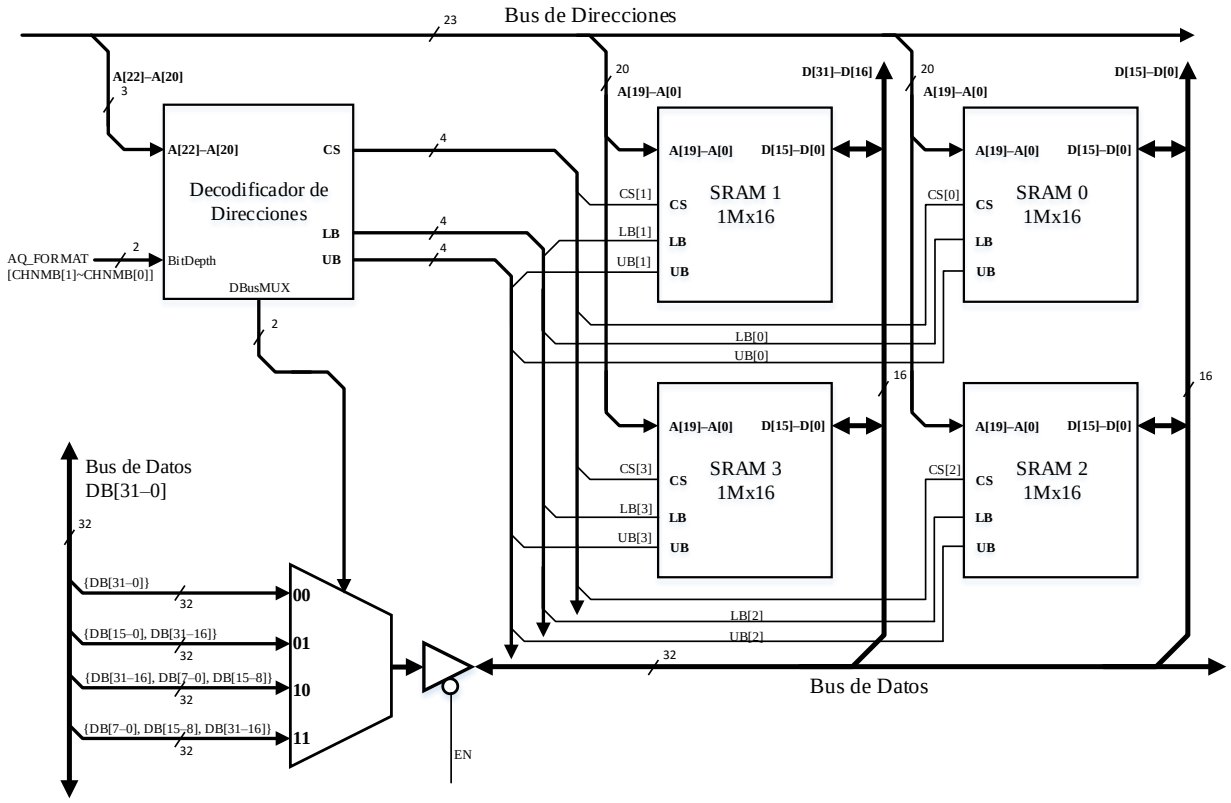


Figura 5.9 Bloque de decodificación de direcciones a memoria de muestreo y organización de datos

Tabla 5.5 Relación de número de canales, capacidad de direccionamiento y organización de bus de datos

Número de canales	Bytes por muestra	Espacio total	Banco activo	DBusMUX
32	4	2Mx32 (64Mbit)	SRAM0 – SRAM2	00
			SRAM1 – SRAM3	00
16	2	4Mx16 (64Mbit)	SRAM0	00
			SRAM1	01
			SRAM2	00
			SRAM3	01
8	1	8Mx8 (64Mbit)	SRAM0 – LB=0, UB=1	00
			SRAM0 – LB=1, UB=0	10
			SRAM1 – LB=0, UB=1	01
			SRAM1 – LB=1, UB=0	11
			SRAM2 – LB=0, UB=1	00
			SRAM2 – LB=1, UB=0	10
			SRAM3 – LB=0, UB=1	01
			SRAM3 – LB=1, UB=0	11

El uso de los 32 canales por parte del usuario hace que los datos relevantes ocupen todo el bus, por lo que las memorias deben activarse por parejas. En caso del uso de 16 u 8 canales, las unidades de memoria se habilitarían individualmente según este parámetro y la dirección del bus. No obstante, en este caso el usuario

debe ubicar sus datos en los canales o bit menos significativos, pero esto dejaría fuera de funcionalidad a las memorias SRAM1 y SRAM3, que se conectan al Word más alto del bus de datos (Ver Figura 5.9). Para garantizar el uso de todas las unidades de memoria se conecta un bloque MUX que intercambia los Word y/o bytes del bus de datos antes de conectarse a ellas. Este intercambio se hace de manera que se lleven los datos de los canales seleccionados (los menos significativos) hacia la sección del bus donde está conectada la memoria habilitada por el decodificador de direcciones. Por tanto, este bloque debe generar una señal de 2-bit para el control de selección del MUX (*DBusMUX*) en función de la memoria habilitada por cada dirección (o sea, acorde a las señales CS, UB y LB).

En la Tabla 5.5 se muestran los valores de *DBusMUX* para cada caso de selección de número de canales y por cada unidad de memoria habilitada. Obsérvese que cuando *DBusMUX*= "00", el bus de datos no cambia; pero cuando se seleccionan 16 canales, el MUX debe llevar los datos del Word menos significativo hacia la palabra más alta si se habilitan SRAM1 o SRAM3. El caso de adquisición con 8 canales requiere atención sobre las señales UB y LB, que habilitan el byte más significativo o el menos significativo del bus de datos respectivamente, para cada unidad de memoria. Con estas condiciones, cada unidad de memoria atendería el acceso a un segmento de 8-bit del bus de datos que la conecta, según sea el estado de estas señales de control. De esta manera, si se habilitan SRAM0 y SRAM2, solamente debe llevarse el byte menos significativo del bus a la parte más alta del Word bajo; esto significa, intercambiar los dos bytes del Word menos significativo. Por otro lado, si se habilitan SRAM1 y SRAM3, se deben intercambiar los Word, de manera que los 16-bit menos significativos sean llevados a la parte más alta del bus; pero si se habilita simultáneamente UB, debe hacerse adicionalmente un intercambio también entre los bytes que conforman esta palabra, de manera que los 8-bit más bajos se sitúen en la parte más alta del bus (Ver las entradas del MUX en la Figura 5.9).

5.4 Subsistema de Disparo

El subsistema de disparo garantiza la detección de cualquiera de las condiciones seleccionadas por el usuario para que el analizador lógico controle su adquisición. Los tipos de disparo soportados para este diseño, definidos en la Figura 5.7, se configuran en los bits *TRGTYP[1:0]* del registro *CONF_TRIG*. Para el cumplimiento y detección de las condiciones de disparo configuradas se requieren esencialmente dos circuitos: un detector de transiciones y un comparador de niveles lógicos. Estos circuitos se presentan en la Figura 5.10 y la Figura 5.11 respectivamente.

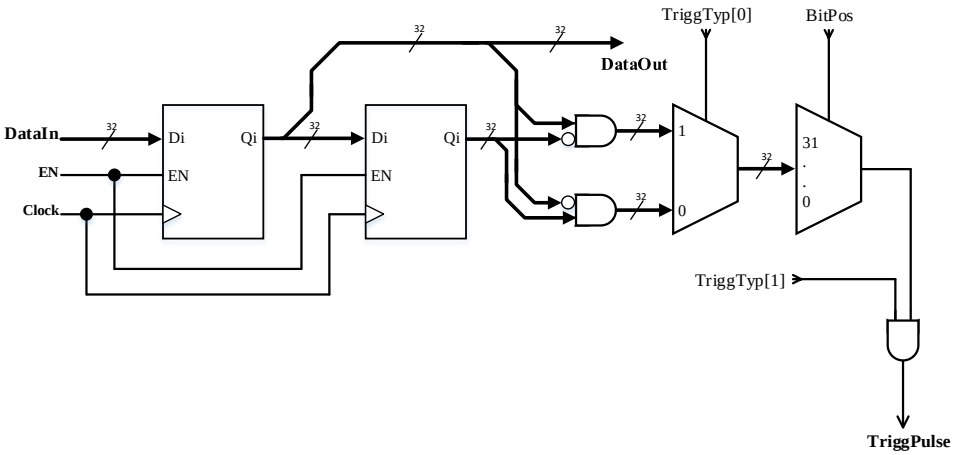


Figura 5.10 Bloque de detección de disparo por transición

El circuito de detección de disparo por transición (Ver Figura 5.10), se basa en un detector de flanco de dos niveles para 32 bits de acuerdo con el máximo número de canales. Este detector tiene dos tipos de salida para indicar, mediante un pulso alto, la presencia de una transición positiva o negativa. La salida que detecta transiciones positivas se conecta a la entrada 1 de un MUX 2-1 de 32-bit, mientras la que detecta flancos negativos se lleva a la entrada 0 del mismo. Este multiplexor selecciona una de estas salidas, en función del tipo de transición de disparo especificado por el usuario en *CONF_TRIG[TRGTYP[0]]*, y entrega los 32 canales a un MUX 32-1 que permite el paso del bit posicionado por *CONF_TRIG[EDGB[4:0]]*. Este último parámetro de 5-bits también lo define el usuario (Ver sección 5.2.2), de acuerdo con el canal que quiere tomar como referencia para el disparo del analizador lógico. Así, si el canal seleccionado presenta una transición activa, habrá un pulso positivo en *TriggPulse*, siempre que *CONF_TRIG[TRGTYP[1]]* esté establecido en '1'; esto es, que el usuario haya configurado un disparo por flanco.

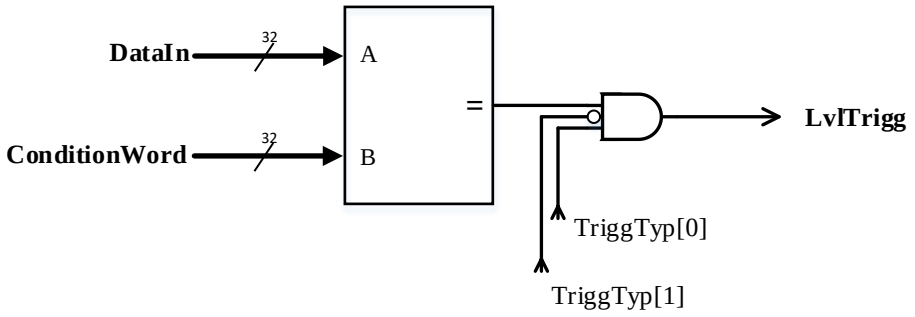


Figura 5.11 Bloque de detección de disparo por nivel

Por otro lado, si se ha configurado un disparo por nivel, se utiliza una palabra de referencia de 32-bits que se compara con el dato muestreado. Esta palabra se define como *TRIG_COND_WORD* (Ver Tabla 5.3), o palabra de condición de disparo en el banco de registros de configuración. Como se presenta en la Figura 5.11, se usa un comparador de 32-bits, cuya salida se activa en *LvlTrigg*, si *TRGTYP[1:0]* selecciona este tipo de disparo.

La palabra de condición de disparo se envía mediante comandos al analizador lógico de acuerdo con los niveles lógicos que el usuario establezca para cada canal. Cuando se configura una adquisición de 8 o 16 canales, se asume que los bits más significativos de los canales descartados están fijados a '0', puesto que el

microcontrolador de la plataforma hardware (Ver sección 4.1.2) aplica una máscara a este parámetro antes de su envío al analizador lógico.

5.5 Subsistema de Muestreo de Datos

El subsistema de muestreo de datos opera conforme a la selección de usuario de usar un muestreo normal o uno transicional. En el primer caso, todas las muestras tomadas por cada ciclo de reloj de usuario deben ser almacenadas; mientras en el segundo, los datos que se repiten o persisten en los canales de adquisición se almacenan solo una vez, seguido por el número de ciclos de reloj de muestreo durante los cuales se mantienen invariantes. El muestreo transicional es una alternativa que usan los analizadores lógicos para optimizar el uso de la memoria en la que residen los datos adquiridos. En la Figura 5.12 se presenta el diseño del subsistema considerando ambos tipos de muestreo.

Los datos que ingresan a este subsistema, que se asume han sido registrados en una etapa anterior, son almacenados en un registro para garantizar la comparación entre la muestra entrante y la inmediatamente anterior. Esta comparación alimentada de registros ($Q_n - Q_{n-1}$), equivalente a considerar un detector de flancos secuencial, es necesaria para el muestreo transicional. Sin embargo, se usa un MUX 2-1 para seleccionar entre la muestra almacenada y el conteo de ciclos de muestreo durante los que se mantiene un dato. En el caso transicional, debe considerarse que este MUX alternaría de manera controlada entre estas dos entradas. El conteo de ciclos para datos que se mantienen es basado en un contador ascendente de 32-bits con entradas predeterminadas, cuyo modo depende del número de canales configurado por el usuario. No obstante, tanto ese contador como el MUX deben controlarse por una máquina secuencial, en función del tipo de muestreo, los límites de conteo y la salida del comparador *EqualSamples*.

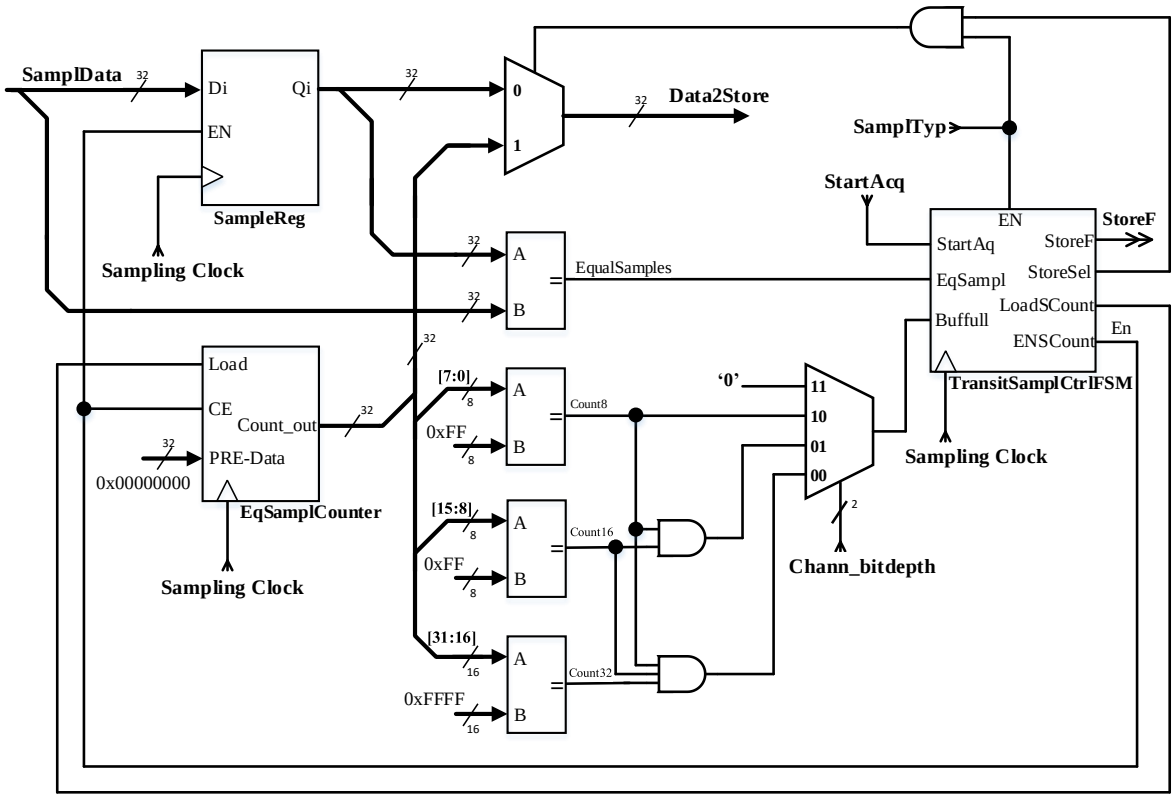


Figura 5.12 Bloque de muestreo de datos

La FSM para el control de las unidades del subsistema de muestro es de tipo *Moore* [119] y tiene cinco (5) estados, como se muestra en la Figura 5.13. Esta máquina secuencial tiene cuatro (4) entradas, que se citan a continuación:

- **EN** (*Enable*): habilita la operación de la FSM. Corresponde a la señal *SamplTyp* (*Sampling Type* – Tipo de muestreo), que se activa en '1' cuando se ha seleccionado muestreo transicional.
- **StartAq** (*Start Acquisition*): indica cuándo ha iniciado el proceso de adquisición (*StartAq*= '1') y el momento de su finalización (*StartAq*= '0').
- **EqSampl** (*Equal Samples*): indica cuándo la muestra n es igual a la anterior $(n-1)$. Corresponde a la salida de un bloque comparador.
- **Buffull** (*Full Buffer*): indica cuándo el contador de ciclos de muestreo de muestras iguales ha alcanzado el máximo valor, de acuerdo con el número de canales especificado por usuario (*Chann_bitdepth* está dado por *AQ_FORMAT[CHNMB[1:0]]*). Esto último se controla mediante un MUX 4-1, en el que se usan tres de sus entradas para indicar el alcance de los valores 0xFF, para 8-bit; 0xFFFF, para 16-bit y 0xFFFFFFFF para 32-bit. Cada una de estas entradas se obtiene mediante aplicación de lógica combinacional, basada en tres comparadores, sobre la salida del contador.

Las salidas de la FSM controlan las señales de habilitación y carga del contador, que son entradas síncronas para él, así como la selección del MUX de datos a almacenar. No obstante, también se emite una señal de advertencia para el generador de direcciones, que, durante un proceso de escritura en memoria de muestreo,

debe detener el puntero mientras arriben datos o muestras iguales. A continuación se describen las salidas de la máquina secuencial:

- **Bit 3 - ENSCount** (*Enable Samples Counter*): habilita el contador de muestras del mismo valor, para su incremento en uno.
- **Bit 2 - LoadSCount** (*Load Samples Counter*): carga las entradas predeterminadas del contador a su salida, haciendo *Count_out*=0x00000000.
- **Bit 1 - StoreSel** (*Storing Selection*): selecciona el dato que debe ser almacenado en memoria de muestreo, controlando un MUX 2-1. *StoreSel* tiene sentido solo durante un muestreo transicional, pues alterna entre los canales 0 y 1 del multiplexor para el almacenamiento de una muestra o de la salida del contador respectivamente.
- **Bit 0 - StoreF** (*Storing Flag*): señal de advertencia que indica cuando un dato está listo para ser almacenado. Así mismo, este indicador debe enviarse al generador de direcciones para que, durante un muestreo transicional, el puntero a memoria de muestreo solo cambie cuando se tenga un nuevo dato para almacenamiento. Por tanto, mientras *StoreF*=0, el bus de direcciones no debe cambiar.

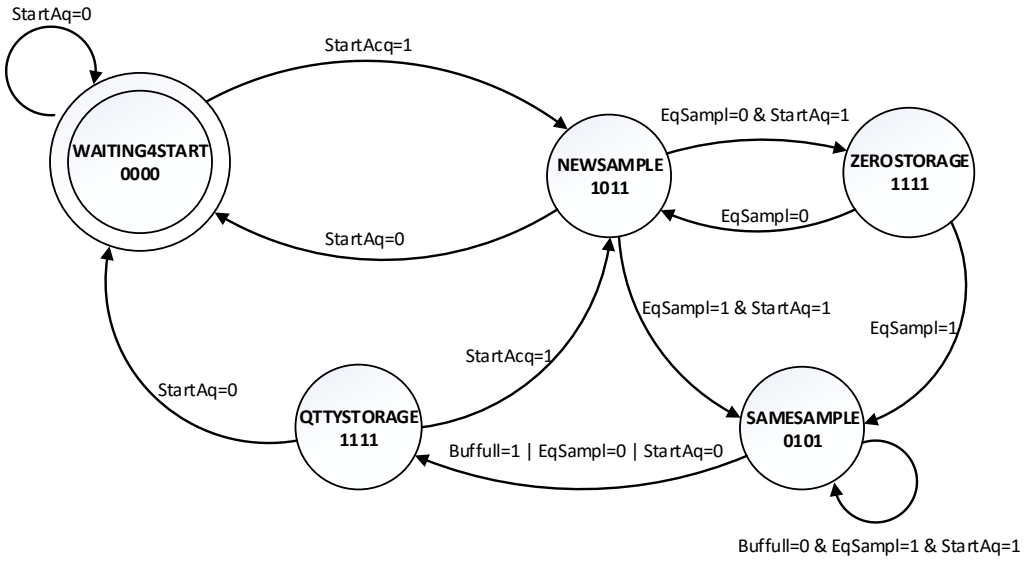


Figura 5.13 Diagrama de estado de FSM para control de muestro

Mientras no haya un proceso de adquisición activo, la FSM se mantiene en el estado *WAITING4START* y con todas las salidas inactivas, lo que inhibe inclusive al almacenamiento en el registro de muestreo *SampleReg*. Cuando se inicia una nueva adquisición (*StartAcq*=1), se alcanza el estado *NEWSAMPLE*, que a través de sus salidas selecciona el dato del registro de muestreo para almacenamiento, dado que se trata de una muestra nueva; así mismo, el contador de ciclos de muestreo se inicializa con la carga de 0x00000000 a sus salidas, preparándose para un eventual conteo en caso de arribar el mismo dato en los siguientes ciclos. Si durante el proceso de adquisición, para el próximo período del reloj de muestreo el dato obtenido es diferente, la salida del comparador *EqualSamples* se coloca en bajo y la FSM pasa al estado *ZEROSTORAGE*, en el que se mantienen la carga del contador y la señal de advertencia de almacenamiento; pero se selecciona la salida del contador para almacenamiento. Nótese que este contador tiene sus salidas en cero para este estado, lo que significa que un cero sería almacenado en la siguiente posición de memoria de muestreo, indicando que la muestra anterior no se repite para el siguiente ciclo de muestreo.

5.6 Subsistema de Control Principal

El subsistema de control principal del analizador lógico basa su operación en una máquina secuencial de seis (6) estados de tipo *Moore*, en la que se establecen las señales de control más relevantes para los procesos de direccionamiento, escritura y lectura de memoria de muestro, actualización de registro de estado, disparo y habilitación de envío de estímulos (Ver Figura 5.14).

La FSM controla todas las fases de operación del analizador lógico a través de 6-bit de salida, que se citan a continuación:

- **Bit 5 - StatUpd** – *Status Update* – señal de advertencia para actualización de registro de *STATUS* del analizador lógico, que define su bit *STATUS[STATF]*. Junto con la señal #WR establecen el estado en que se encuentra, fundamentalmente después del disparo, como lo muestra la Tabla 5.6. Estas señales conforman un bus de cuatro líneas (4-bit) que se escribe sobre el *nibble* bajo del registro STATUS, actualizando también el bit *STATUS[STRTAQ]*, una vez ha terminado la adquisición de muestras (Ver Figura 5.7). El bus está definido por [Ec. **5.4**].

$$FSMStat\_Updt = \{'0', \#WR, StatUpd, '0'\} \qquad \textbf{[Ec. 5.4]}$$

- **Bit 4 - TRGF** – *Trigger Flag* – señal de advertencia de disparo del analizador lógico. Se activa en alto para indicar que se ha alcanzado la condición de disparo establecida por el usuario.
- **Bit 3 - AddrLD** – *Address Load* – señal para la carga del contador en el generador de direcciones, de manera que se prepare para el inicio de una operación de escritura o lectura en memoria de muestreo.
- **Bit 2 - AddrEN** – *Address Enable* – señal de habilitación del contador en el generador de direcciones. Este bit establece el permiso para su carga o incremento del puntero en el bus de direcciones.
- **Bit 1 - #WR** – *Write* – Señal de escritura que se activa en bajo (#*WR*='0') cuando se requiere una operación de escritura en memoria de muestreo. Si #*WR*='1' establece una operación de lectura.
- **Bit 0 - StimEN** – *Stimuli Enable* – Señal de habilitación de aplicación de estímulos. Esta señal se activa en alto cuando inicia el proceso de adquisición, de manera que se habilite la salida del patrón de bits establecido por el usuario, para ser aplicado al sistema bajo prueba.

Tabla 5.6 Reporte de estados post-*triggered* con base en *WR* y *StatUpd*

#WR	***StatUpd (STATUS[STATF])***	**Estado**
0	**0**	ADDR_PRELOAD STOPPED RUNNING
0	**1**	TRIGGERED
1	**0**	ADDR_LOAD
1	**1**	READING

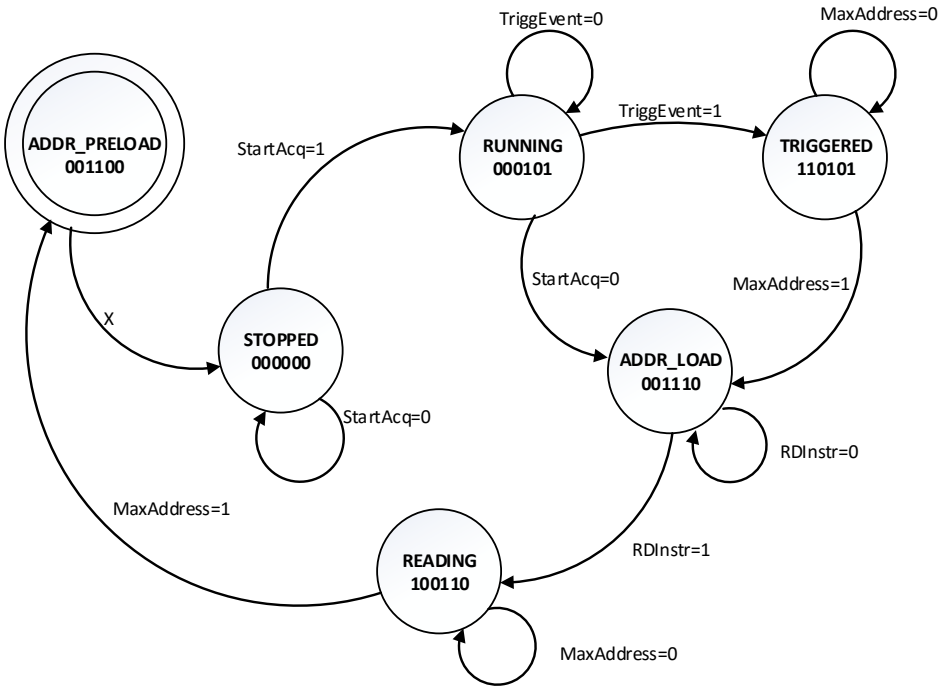


Figura 5.14 Diagrama de estado de la FSM de control principal del analizado lógico

Las entradas de la FSM de control principal se definen a continuación:

- **StartAcq**: señal proveniente de *STATUS[STRTAQ]*, que indica a la FSM cuándo un proceso de adquisición ha sido iniciado por el usuario (*StartAcq*=‘1’).
- **TriggEvent**: señal procedente del circuito de disparo, que informa a la máquina secuencial cuándo se ha alcanzado el evento de disparo especificado por el usuario (*TriggEvent*=‘1’).
- **MaxAddress**: señal emitida desde el subsistema de direccionamiento a memoria de muestreo, que indica a la FSM cuándo se ha alcanzado la máxima dirección, de acuerdo con la configuración realizada por el usuario (Ver sección 5.3).
- **RDInstr**: señal obtenida de *STATUS[RDF]*, para que la máquina secuencial conozca cuándo el usuario ha solicitado la lectura de la memoria de muestreo.

El estado *STOPPED* de la FSM de control principal es en el que opera la mayor parte del tiempo, dado que solo conmuta a otro estado cuando se inicia un proceso de adquisición desde *STATUS[STRTAQ]*. Esta acción se deriva de la aplicación Web discutida en la sección **Error! Reference source not found.** y su nvío hacia la plataforma hardware sólo se hace cuando se alcanza la fase de *debug*. Previamente, el usuario ha seguido un proceso de configuración de parámetros del analizador lógico y, posteriormente a la adquisición, realiza el análisis de los datos obtenidos o ajusta los parámetros necesarios para otro ciclo de adquisición; por tanto, la ventana de tiempo en la que opera el analizador con mayor demanda de potencia es muy pequeña con respecto al tiempo que permanece a la espera de un nuevo requerimiento del usuario. Durante este estado de detención y espera por un nuevo ciclo de adquisición, las salidas de la FSM son bajas, inhabilitando los módulos derivados y reiniciando los buses de manera que se disminuya el consumo de potencia del sistema.

Por otra parte, el estado *ADDR_PRELOAD* se convierte en un estado transitorio cuyo objetivo es restablecer los punteros en el subsistema de direccionamiento, una vez se ha concluido un ciclo de operación del usuario; esto es, configuración, adquisición y lectura- visualización de datos obtenidos. Así, este estado establece las salidas *AddrLD* y *AddrEN*, lo que reinicia el contador de direcciones a la memoria de muestreo, y conmuta incondicionalmente al estado *STOPPED*, en el que espera por una instrucción ***STARTAQ***.

Cuando el usuario ha iniciado un nuevo ciclo de pruebas con el analizador lógico y solicita el inicio de la adquisición (*StartAcq*='1'), la FSM pasa al estado *RUNNING*. Durante este estado, se habilitan la aplicación del patrón de estímulos sobre el sistema bajo prueba, el incremento del puntero a memoria de muestreo y la señal de escritura, mientras no se presenta algún evento de disparo. Si esto ocurre (*TriggEvent*='1'), se presenta una transición al estado *TRIGGERED*, en el que se mantienen en sus estados activos las salidas anteriores y se establecen también el bit de actualización de estado y la bandera de disparo, para indicar a los demás módulos que el evento de disparo se ha alcanzado y la FSM ha asumido el control de tal estado.

Con un proceso de adquisición activo, independientemente de haber alcanzado el evento de disparo, el usuario podría terminar el ciclo mediante el envío de un comando *STOPAQ*, lo que llevaría a la FSM al estado *ADDR_LOAD*, para detener el puntero a memoria de muestreo y esperar por la solicitud de lectura de los datos almacenados. No obstante, si la adquisición está corriendo bajo el estado *TRIGGERED*, solo lo hará hasta que llene la memoria de muestreo. En este instante, se presenta también una transición hacia *ADDR_LOAD*, se desactivan la bandera de disparo y la señal de escritura (#*WR*='1') y se restablece el puntero a memoria de muestreo a su dirección base, de acuerdo con la configuración de usuario y el evento de disparo (Ver sección 5.3). Cabe indicar también, que la aplicación de estímulos se suspende tras la terminación de la adquisición de muestras.

Cuando la solicitud de lectura de datos de la memoria de muestreo es enviada al analizador lógico (*RDInstr*='1'), se habilitan nuevamente el incremento del puntero en el sistema de direccionamiento y el bit de actualización de estado. Esto se hace en el estado *READING*, con el fin de recorrer secuencialmente la memoria de muestreo desde la dirección base (correspondiente al porcentaje de memoria establecido por usuario para el pre-disparo) hasta la máxima dirección donde se detuvo la adquisición. Además, el registro *STATUS* se actualiza con la desactivación de *STRTAQ* y el establecimiento de #*WR* y *STATF*.

5.7 Subsistema de Sincronización de Señales de Control

Las principales señales de control del analizador lógico surgen en un dominio de reloj rápido, del que se derivan señales de reloj de más baja frecuencia que el usuario puede seleccionar para el proceso de adquisición. Esto puede causar la invisibilidad de los niveles importantes de activación de estas señales en el dominio de reloj de usuario, razón por la cual es esencial sincronizarlas de acuerdo a este último.

La señal *USRFreqCtrl[1:0]* tiene la configuración especificada por el usuario para la frecuencia de muestreo, de acuerdo como reside en los bit *SFRQ[1:0]* del registro *AQ_FORMAT*. La relación de frecuencias, con referencia en la frecuencia base de reloj de la plataforma hardware *Fosc*, y la combinación de bit de este registro se muestra en la Tabla 5.7. Obsérvese que la relación de división de frecuencias es válida también para la aplicación de estímulos en los bits *SFRQ[3:2]*.

Tabla 5.7 Frecuencia seleccionada por el usuario para muestreo y aplicación de estímulos

Frecuencia de usuario	SFRQ[1:0]/SFRQ[3:2]
Fosc	00

Fosc/2	01
Fosc/4	10
Fosc/8	11

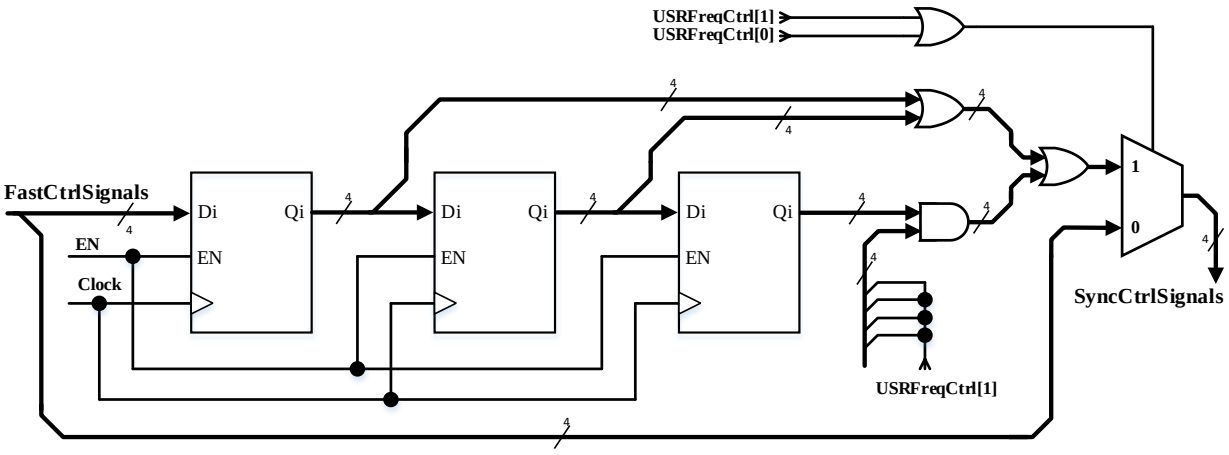


Figura 5.15 Bloque de sincronización de señales de control rápidas

El bloque de sincronización que se presenta en la Figura 5.15, se basa en tres niveles de registros Q_n, Q_{n-1}, Q_{n-2}, que definen la salida así:

$$SyncCtrlSignals = (USRFreqCtrl[1] + USRFreqCtrl[0]) \cdot \left(Q_n + Q_{n-1} + (Q_{n-2} \cdot USRFreqCtrl[1])\right) + FastCtrlSignals \cdot \overline{(USRFreqCtrl[1] + USRFreqCtrl[0])} \quad \textbf{[Ec. 5.5]}$$

El retraso intencional de las señales rápidas $FastCtrlSignals$ que ingresan al sistema consiste en aprovechar su activación (en alto) para prolongar la duración del pulso durante más de un período del reloj de los registros, usando operaciones OR entre cada retraso. Observe que para frecuencias más bajas, cuando *USRFreq_Ctrl[1]*='1', se introduce Q_{n-2} a la operación OR para prolongar la duración del pulso. El MUX 2-1 daría paso a las señales rápidas que ingresan al subsistema cuando el usuario ha seleccionado la frecuencia más alta del sistema (la misma del reloj de control) como frecuencia de muestreo, de manera que no se aplique retardo alguno. En los otros casos, se selecciona la combinación de los retrasos Q_n, Q_{n-1}, Q_{n-2}, como lo define [Ec. **5.5**].

5.8 Interconexión de los Subsistemas del Analizador Lógico

El analizador lógico, como módulo a implementar en un solo chip, dispone de entradas para su configuración y operación, así como las interfaces con el sistema bajo prueba y salidas de información de su estado de funcionamiento. Cada uno de los subsistemas descritos en este capítulo han sido instanciados usando *Verilog* [120] en una descripción estructural, que tiene como unidad de diseño principal el módulo que se muestra en la Figura 5.16. En esta unidad se identifican cinco (5) entradas:

- **Command_In**: corresponde al bus de 8-bit *Command Input*, presentado en 5.2.1, por el que se ingresan los comandos y parámetros de configuración al analizador lógico.

- **Sample2Store**: es un bus de 32-bits donde se conectan los canales a analizar del sistema bajo prueba.
- **Hw_RST**: la activación de esta señal reinicia las unidades internas a sus estados de configuración y operación predeterminados. Esta línea permite la conexión en hardware de un circuito basado en *switch* que asegure el estado de activación cuando se requiera aplicar un *reset* al sistema.
- **REQ**: como se definió en 5.2.1, la activación de esta señal inicia una transacción hacia el analizador lógico, a través de *Command_In*.
- **Sys_Clk**: es el reloj principal del sistema que controla especialmente la temporización de registros y máquinas secuenciales dentro del analizador lógico.

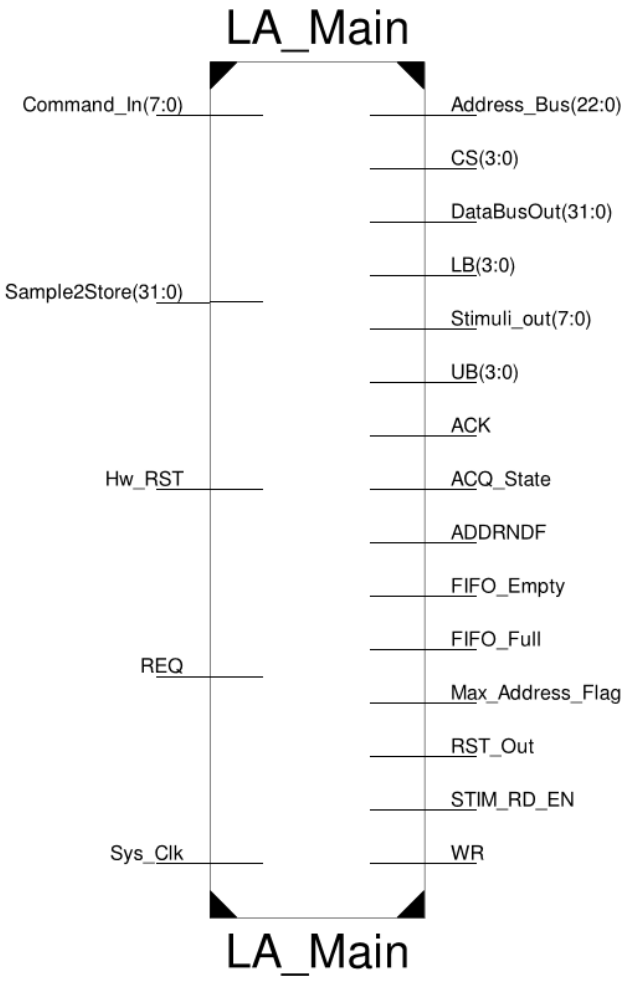


Figura 5.16 Bloque de sincronización de señales de control rápidas

Las salidas del analizador lógico controlan los procesos de direccionamiento y escritura de datos a memoria de muestreo, comunicación con microcontrolador y aplicación de estímulos, así como también ofrecen información de estado para el sistema que requiera conocerlo. Estas salidas se resumen a continuación:

- **Address_Bus**: Bus de direcciones de 23-bit, cuyos 20-bits menos significativos se conectan a las entradas de direccionamiento de las unidades de memoria SRAM externa (Ver sección 5.3).

- **CS**: Bus de 4-bits que conecta a las entradas de CE(*Chip Enable*) y OE(*Output Enable*) de las unidades de memoria SRAM externa (Ver sección 5.3.1).
- **LB**: Bus de 4-bits que conecta a las entradas de LB(*Lower-byte Control*) de las unidades de memoria SRAM externa (Ver sección 5.3.1).
- **UB**: Bus de 4-bits que conecta a las entradas de UB(*Upper-byte Control*) de las unidades de memoria SRAM externa (Ver sección 5.3.1).
- **WR**: Señal de escritura emitida por la FSM de control principal, que controla el almacenamiento en memoria SRAM externa e informa a otros subsistemas del analizador lógico el proceso activo.
- **DataBusOut**: Bus de datos de 32-bits, que conecta las unidades de memoria SRAM externa y el microcontrolador. A través de este bus circulan los datos que se almacenan en memoria y los que son leídos (Ver *Data2Store* en 5.5).
- **Stimuli_out**: Bus de 8-bits en el que se aplican los patrones de estímulos al circuito bajo prueba (Ver sección 5.2.5).
- **STIM_RD_EN**: Señal de estado que informa cuándo se está aplicando el patrón de estímulos sobre el circuito objetivo.
- **ACK**: Señal de respuesta del analizador lógico en el control de comunicación asíncrona presentada en 5.2.1.
- **ACQ_State**: Señal de estado que indica cuando el analizador lógico ha alcanzado sus estados *ADDR_LOAD* o *READING*.
- **ADDRNDF**: Señal de estado que anuncia cuándo se ha alcanzado la máxima dirección absoluta de memoria y se ha reiniciado el puntero durante un proceso de escritura (estados *RUNNING* o *TRIGGERED*). Esta señal se usa para indicar si se ha tomado una cantidad de muestras suficientes para completar los rangos de almacenamiento pre y post- disparo especificados por el usuario. Su restablecimiento a bajo se hace durante el estado *ADDR_PRELOAD*. Estas condiciones son chequeadas mediante dos puertas AND, que controlan el estado de la bandera *ADDRNDF* a través de un *flip-flop* (Ver parte inferior de la
- Figura 5.18).
- **FIFO_Full**: señal de advertencia que indica cuándo se ha llenado la cola de almacenamiento de estímulos (Ver secciones 5.2.5 y 6.3.2).
- **FIFO_Empty**: señal de advertencia que indica cuándo la cola de almacenamiento de estímulos se encuentra vacía (Ver secciones 5.2.5 y 6.3.2).
- **Max_Address_Flag**: señal de advertencia que indica que se ha alcanzado la máxima dirección de memoria de muestreo, en función de los parámetros de configuración dados por el usuario (Ver sección 5.3).
- **RST_Out**: señal de *reset* que se activa siempre que se envía un comando RST o se establece por *hardware* el bit *Hw_RST*.

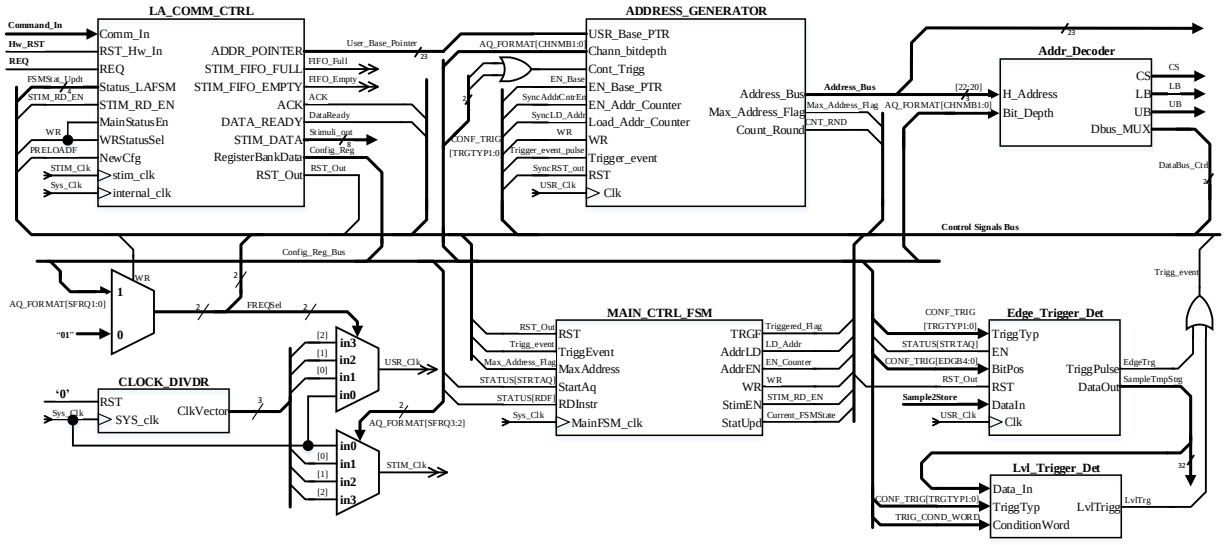


Figura 5.17 Diagrama en bloques del analizador lógico – Interconexión de subsistemas de comunicación, generación de direcciones, circuito de reloj, detección de disparo y control principal

En las Figura 5.17 y

Figura 5.18 se muestra el Diagrama en bloques del analizador lógico propuesto y la interconexión de todos los subsistemas que lo conforman. Obsérvese la dependencia del bus *Config_Reg_Bus*, que conecta el banco de registros de configuración para los subsistemas que lo requieren. Así mismo, se ha usado un bus *Control_Signals_Bus* para simplificar gráficamente la interconexión de las señales de control en los diagramas. En la primera figura aparecen fundamentalmente los bloques de configuración y comunicación (*LA_COMM_CTRL*), generación de direcciones (*ADDRESS_GENERATOR*), direccionamiento a memoria de muestreo (*Addr_Decoder*), detección de disparo por flanco (*Edge_Trigger_Det*), detección de disparo por nivel (*Lvl_Trigger_Det*), máquina de estados de control principal (*MAIN_CTRL_FSM*) y control de base de tiempos. Este último bloque está basado en un divisor de frecuencia (*CLOCK_DIVDR*), con reloj base *Sys_Clk* de frecuencia F_{OSC} y con tres (3) salidas de frecuencias $\frac{Fosc}{2}$, $\frac{Fosc}{4}$ y $\frac{Fosc}{8}$. La frecuencia seleccionada por el usuario para la operación de los subsistemas encargados del muestreo y almacenamiento se obtiene en *USR_Clk*, a través de un MUX 4-1; así mismo, sucede con la frecuencia del reloj para aplicación de estímulos que se define en *STIM_Clk*. Para el control del multiplexor que entrega *USR_Clk*, se selecciona entre los bits *AQ_FORMAT*[*SFRQ1*:*0*] y la palabra "01", según se trate de una operación de escritura o lectura respectivamente; esto permite fijar el reloj de manera predeterminada para el proceso de lectura a una frecuencia $\frac{Fosc}{2}$, que no depende de la configuración dada por el usuario. El selector de reloj para la aplicación de estímulos se controla directamente por *AQ_FORMAT*[*SFRQ3*:*2*], que es definido por el usuario.

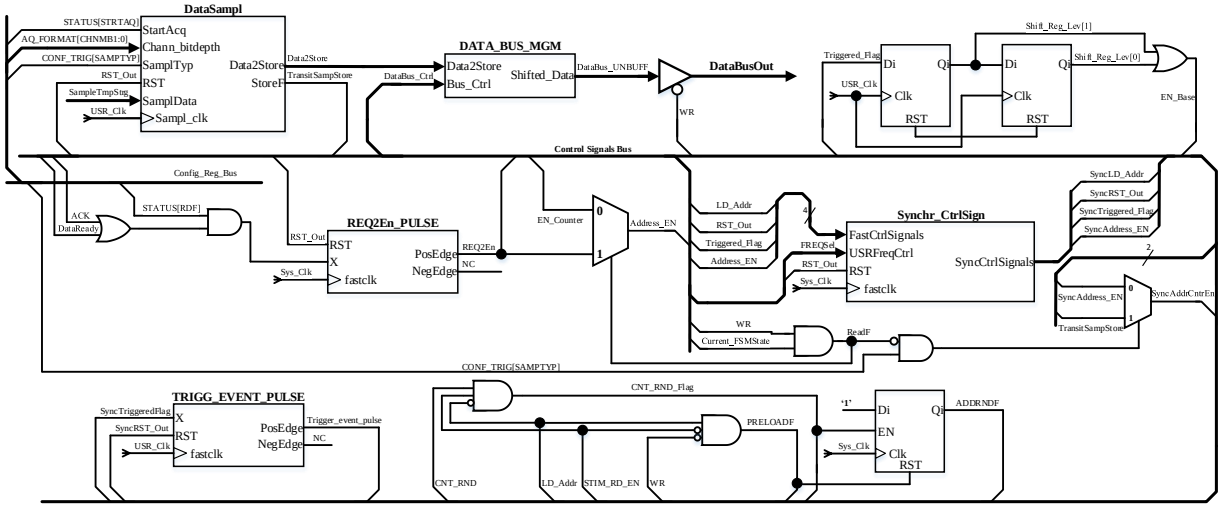


Figura 5.18 Diagrama en bloques del analizador lógico – Interconexión de subsistemas de muestreo, administración de bus de datos y sincronización

En la Figura 5.18 se muestra la interconexión de los bloques de muestreo (*DataSampl*), organización de datos (DATA_BUS_MGM) y sincronización (*Syncr_CtrlSign* – *REQ2En_Pulse* – *TRIGG_EVENT_PULSE*). Los bloques de sincronización cumplen un papel fundamental en este sistema debido a que los bloques de generación de direcciones, muestreo y detección de disparo operan a la frecuencia de muestreo definida por el usuario, pero algunas de sus señales de control, especialmente para habilitación de registros, son generadas por el sistema de control principal, que utiliza la frecuencia más alta del sistema. Por ejemplo, el bloque *Syncr_CtrlSign* tiene como entradas las señales de carga para el contador de direcciones, *reset*, bandera de disparo y habilitación del contador de direcciones (*LD_Addr*, *RST_Out*, *Triggered_Flag*, *Address_EN*), las cuales se originan en subsistemas que operan a la frecuencia F_{osc}; sin embargo, de acuerdo con la frecuencia seleccionada por el usuario para el muestreo, sus pulsos de activación son prolongados (*SyncLD_Addr*, *SyncRST_Out*, *SyncTriggered_Flag*, *SyncAddress_EN*) antes de conectarse a los subsistemas correspondientes.

Por otra parte, los subsistemas *REQ2En_Pulse* y *TRIGG_EVENT_PULSE*, basados en detectores de transición secuenciales de dos niveles, se usan para generar pulsos activos alto de acuerdo con las frecuencias de reloj del sistema y de usuario respectivamente. El primero de ellos lo hace cuando se ha enviado un comando ***RDSMEM***, de manera que se genere un pulso de habilitación para la iniciación de punteros del generador de direcciones durante el inicio de un proceso de lectura, como apoyo a la señal que envía la FSM de control principal para tal fin. El bloque *TRIGG_EVENT_PULSE* genera un pulso alto cada vez que la bandera *SyncTriggered_Flag* está también activa; ésto se hace para la habilitación de punteros en el subsistema de direccionamiento a memoria de muestreo, en su bloque *ADDRESS_GENERATOR* (Ver sección 5.3). La señal de habilitación del puntero base, que está dentro de este bloque, se genera también con base en la prolongación del estado de activación de la señal de disparo, mediante la combinación de salidas de dos *flip-flop* conectados en cascada (Ver parte superior derecha de la Figura 5.18).

6. RESULTADOS

Este capítulo presenta los principales resultados encontrados durante el desarrollo del sistema informático, resaltando la pertinencia y funcionalidad de la aplicación web a través de una interfaz gráfica de usuario, los detalles de la implementación del prototipo de la plataforma hardware y la síntesis e implementación del analizador lógico que facilita el proceso de depuración de las funciones lógicas que implementa el usuario.

6.1 Resultados de Implementación de la Aplicación Web

A continuación se presentan los resultados obtenidos en cada etapa de la construcción del sitio y algunas pruebas realizadas con la aplicación.

6.1.1 Resultados del Diseño del Sitio Web

Considerando el proceso de diseño descrito en la sección 3.4 se presenta el resultado obtenido en cada etapa de la construcción del sitio Web.

6.1.1.1 Resultados de Gestión de Usuarios

En 3.4.2 se describe el diseño de la página que posibilita que un docente adicione o registre usuarios. La Figura 6.2 presenta el resultado de esta página y la Figura 6.3 presenta la página en la que un usuario puede modificar su perfil, insertando los campos *nombre, usuario, correo, contraseña*, entre otros.

6.1.1.2 Resultados Gestión de Citas

Para la gestión de citas descrita en 3.4.3, se han desarrollado una serie de páginas que le informan a un usuario lo sucedido con las citas. En la Figura 6.4 se presenta la página diseñada para el estudiante que ingresa al sitio sin haber solicitado una cita, por lo que no aparece en la base de datos. Por otra parte, en la Figura 6.5 se presenta el caso para cuando un usuario ha tenido citas previas pero no tiene una cita ni actual ni futura y finalmente, la Figura 6.6 presenta la interfaz para cuando el usuario tiene una cita asignada.

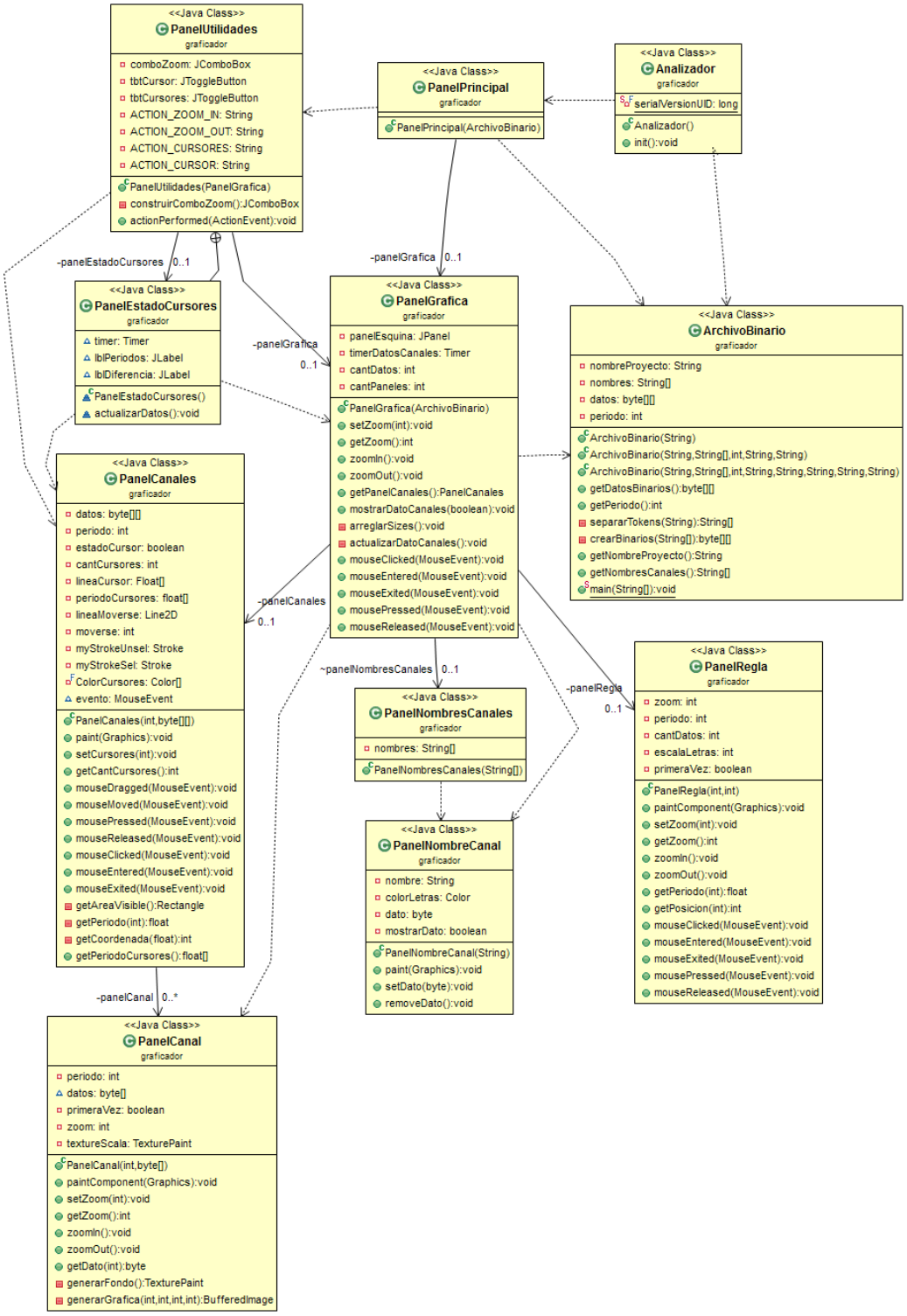


Figura 6.1 Diagrama de clases del *applet* construido para visualización de resultados

Figura 6.2 Registro de usuarios por parte de un docente

Figura 6.3 Campos para que docentes y estudiante editen sus perfiles de usuario

Figura 6.4 Usuario que no aparece en la base de datos

Figura 6.5 Usuario con citas ya cumplidas, pero sin cita futura

Figura 6.6 Usuario que ingresa horas previas a la de la cita solicitada

En esta etapa de gestión de citas, se diseñó también una página que contiene un calendario conectado con la base de datos para especificar disponibilidad de horarios en el laboratorio en cuanto a fechas y horas. En la Figura 6.7 está el resultado de la página con el calendario, en la que el estudiante debe elegir la fecha más conveniente para él. En la Figura 6.8 muestra la interfaz con disponibilidad de horas en la que, una vez escogido el día de la cita, se debe seleccionar la hora de la misma.

Figura 6.7 Calendario para solicitud de citas por parte de los usuarios

Figura 6.8 Disponibilidad de horarios en el laboratorio

Una vez que el usuario tiene definido el día y la hora de su cita, debe pasar a confirmar la misma; por eso, la aplicación presenta la información de la cita agendada al usuario y el botón para confirmar la misma (Ver Figura 6.9).

Figura 6.9 Información personal sobre la cita solicitada

6.1.1.3 Resultados de Integración de Herramientas Web 2.0 y LMS

En la sección 3.4.4 se describió el proceso de gestionar contenido en el CMS. Todas las páginas diseñadas se han construido con este sistema de gestión de contenidos; por lo tanto, todos los resultados presentados en la sección 6.1 reflejan su operatividad.

De acuerdo con la inclusión de las herramientas Web 2.0 descrita en 3.4.5, se presentan las funciones de la red social *facebook* a las que un usuario puede acceder. En la Figura 6.10 se observa el resultado de la acción de un usuario que desea ingresar al sitio con su cuenta personal de *facebook*, una vez que ha iniciado sesión con la red social, puede hacer uso de algunas funciones como subscribirse a la página en *facebook* del grupo de investigación GADyM, como se observa en la Figura 6.11o calificar determinado contenido con el botón "*Me Gusta*", como se observa en la Figura 6.12.

Figura 6.10 Aplicación de *login* con *Facebook*

Figura 6.11 Función de subscribirse de *facebook*

Herramientas de la Web 2.0 diferentes de *facebook* también han sido adicionadas a la aplicación Web. En la sección 3.4.6 se ha descrito el proceso para integrar *Joomla* con *Moodle* que permitió la integración de herramientas como las que se observan en la Figura 6.13.

Una vez realizada la integración entre los gestores de contenido y de aprendizaje, el resultado obtenido al realizar la creación de un curso es la que se observa en la Figura 6.14.

Figura 6.12 Función "*Me gusta*" para valorar contenido de *Facebook*

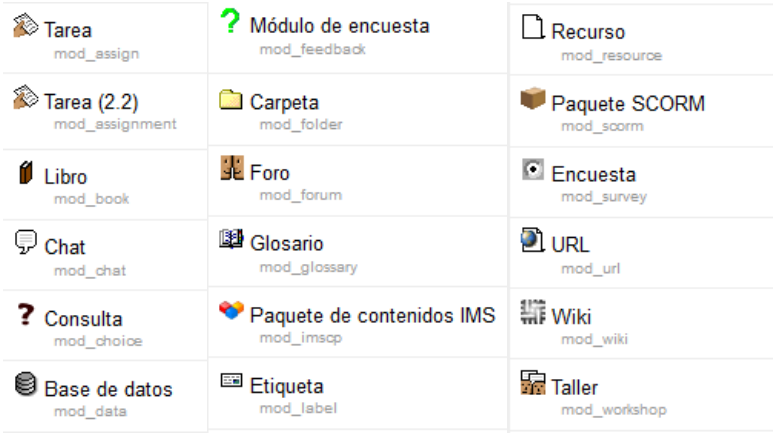


Figura 6.13 Funciones disponibles gracias a la integración con el LMS.

6.1.1.4 Resultados de la Herramienta de Configuración

En la Figura 6.15se presenta el diagrama de flujo que describe el proceso que un usuario debe seguir para realizar una práctica remota en el laboratorio; adicionalmente, tanto procesos como subprocesos se han enumerado para realizar una descripción detallada de cada uno.

El proceso enumerado como (1) y cuyo diseño se ha descrito en la sección 3.4.8.1, es la página realizada para que el usuario cargue el archivo de configuración de la FPGA. La Figura 6.16 presenta el resultado de este diseño. Adicionalmente, se observa en (2) de la Figura 6.15 que si el archivo cargado no es de la extensión que el sistema reconoce (**.bit*), se lanza un mensaje de alerta como se observa en la Figura 6.17.

El proceso enumerado como (3) en la Figura 6.15es el siguiente paso en el laboratorio. La página diseñada para que el usuario cargue el archivo que define la asignación de pines que el usuario seleccionó. El diseño de esta página es el presentado en la Figura 6.18 y tiene asociado su respectivo mensaje de alerta, al igual que el caso anterior. Este mensaje está enumerado como (4) en la Figura 6.15.

La siguiente etapa del proceso enumerada como (5) es la configuración del analizador lógico. La página diseñada para este proceso, descrito en la sección 3.4.8.2, se presenta en la Figura 6.19.

Este mismo proceso tiene varios pasos, dependiendo de las acciones que tome el usuario. Si el usuario le entrega a la aplicación el *testBench*, la aplicación toma este archivo, lo modifica y se lo entrega al usuario a través de una descarga, tal y como se observa en la Figura 6.20.

Una vez que el usuario descarga el archivo y realiza nuevamente la simulación, se le entrega un archivo llamado *simulación.txt.* El archivo resultado de esta simulación (**.tit*) tiene el formato presentado en la Figura 6.21, en el que se listan las entradas y el tiempo en el que ocurre cada una de las transiciones que sufren las señales.

Por último, en lo que a esta página se refiere, se tiene la elección del tipo de configuración de la FPGA. Esta etapa se presenta en la Figura 6.22. Esta página también se diseñó con las alertas necesarias para guiar al usuario en el proceso de configuración y así minimizar la posibilidad de errores. La Figura 6.23 presenta las variadas alertas que se pueden generar. Esto hace parte de la etapa (6) de la Figura 6.15.

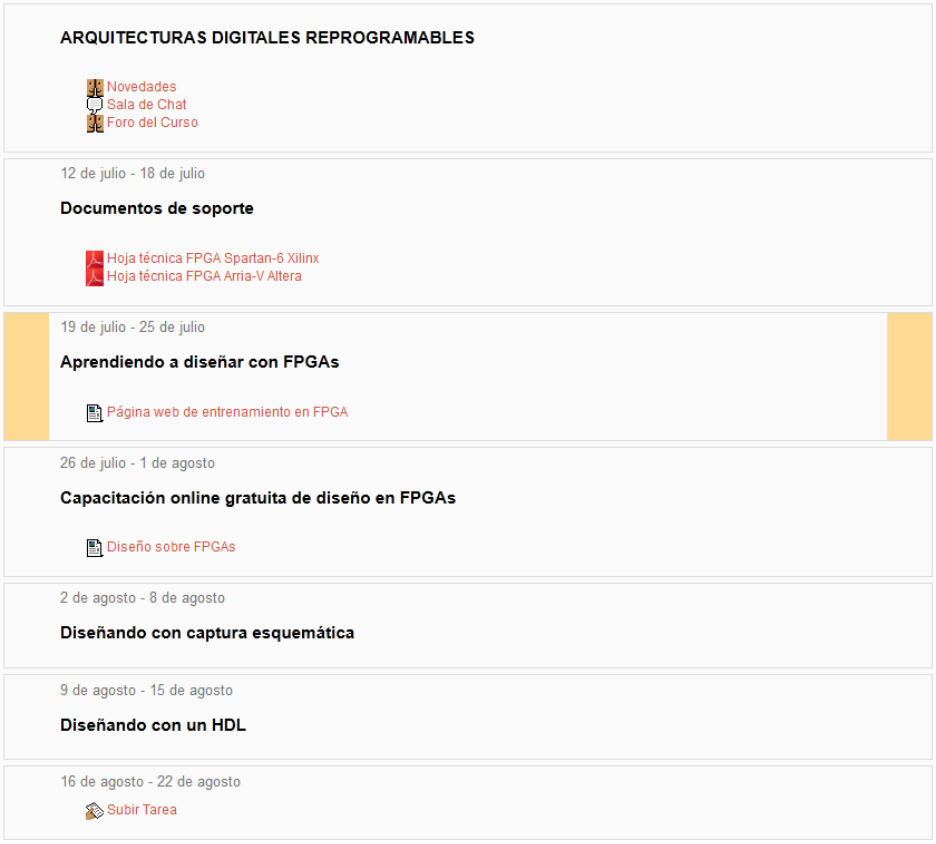


Figura 6.14 Herramientas vinculadas a la aplicación desde el gestor de aprendizaje

Una vez que el usuario avanza este paso (6), el *Servlet* encargado de atender esta petición, procesa el archivo subido por el usuario y el resultado es otro archivo, a partir del cual se generarán los estímulos. El formato del archivo para la generación de estímulos se presenta en la Figura 6.21.

El proceso (7) de la Figura 6.15 es la página en la que el usuario debe proceder con la ejecución del proceso pulsando el botón que se presenta en la Figura 6.25.

Una vez se avanza al proceso (8), en el que el sistema se ejecuta en el segundo servidor y se encuentra a la espera de los resultados, pueden ocurrir errores como se menciona en la etapa (9). La Figura 6.26 indica un error en la conexión con el servidor *FTP*.

Por otra parte la Figura 6.27 presenta el caso en el que el *hardware* no se encuentra conectado al servidor y, por lo tanto, no está disponible.

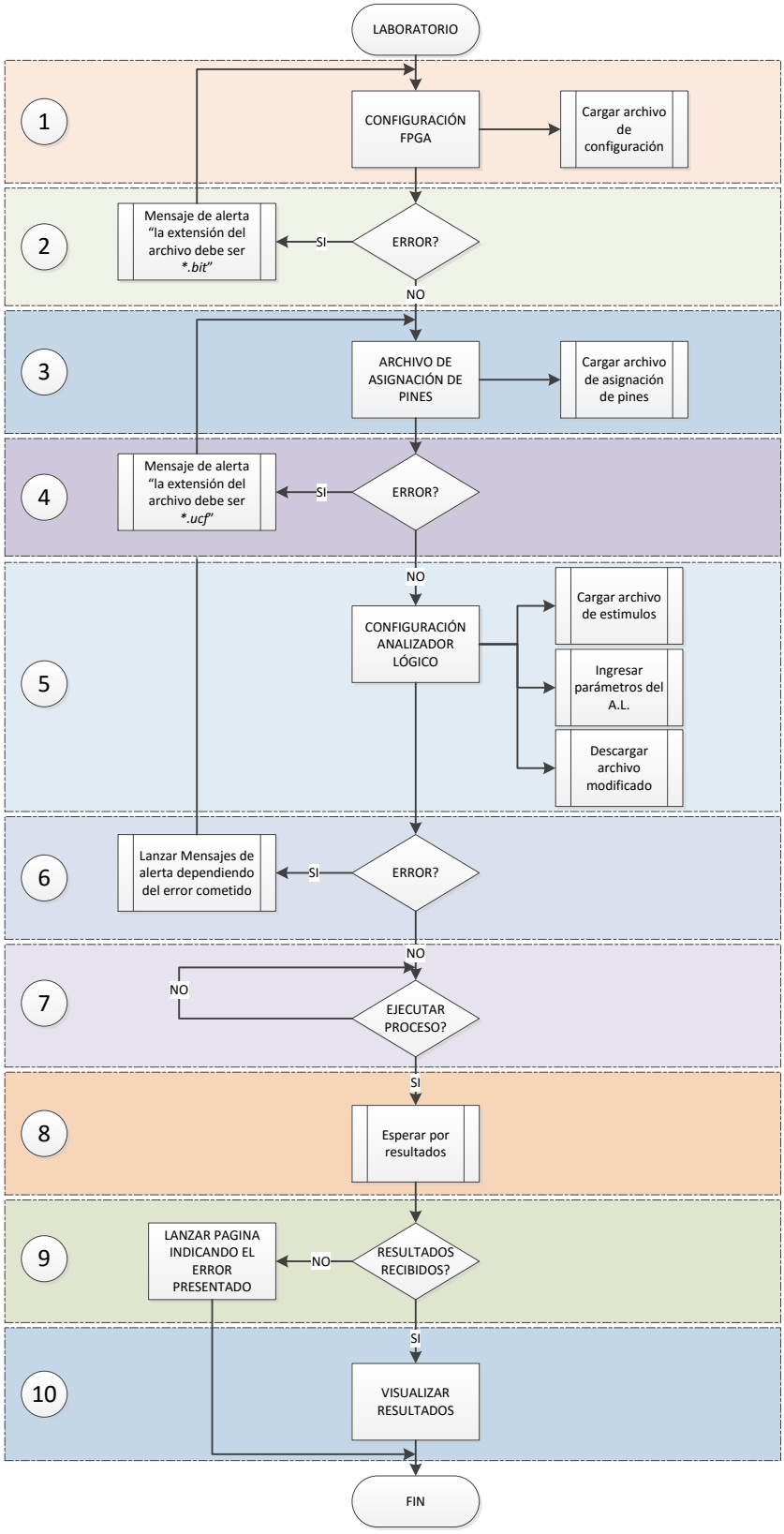


Figura 6.15 Diagrama de flujo del funcionamiento del laboratorio

Figura 6.16 Pagina para cargar archivo de configuración de la FPGA

Figura 6.17 Mensaje de alerta, si se carga un archivo de extensión diferente de *.bit*

Finalmente, la etapa (10) es cuando ha terminado exitosamente el proceso de captura de resultados y éstos se le presentan al usuario. La Figura 6.28 presenta la página diseñada para redirigir al usuario a la aplicación de visualización de resultados.

La aplicación diseñada para presentar los resultados a los usuarios es un *Applet* inmerso en la página Web. Esta aplicación se presenta en la Figura 6.29.

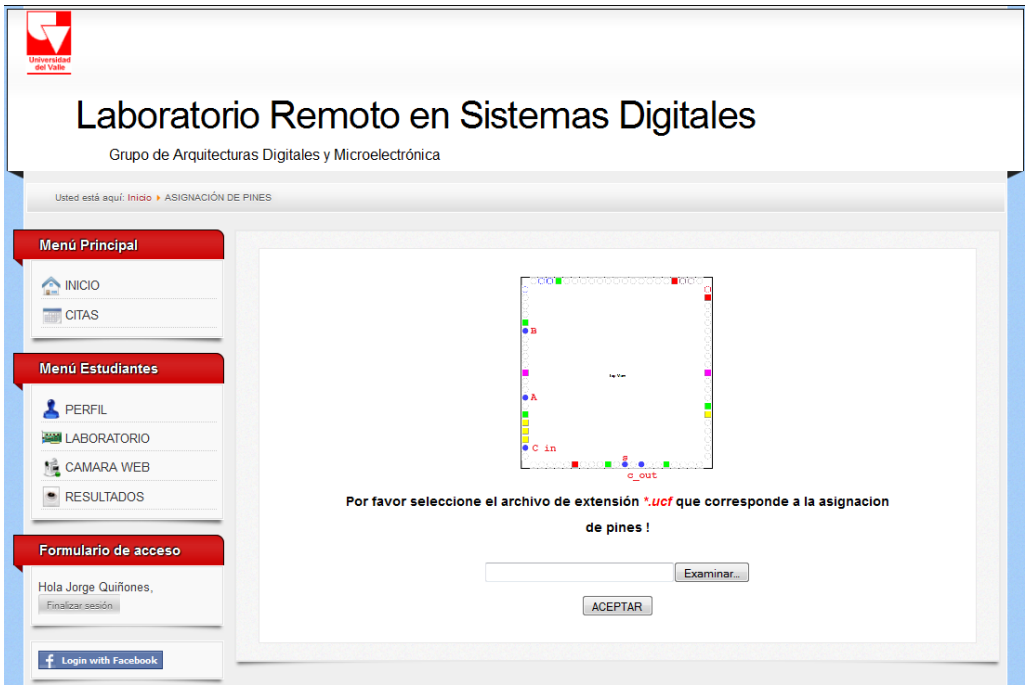


Figura 6.18 Página para cargar el archivo de asignación de pines *.ucf*

Figura 6.19 Página para realizar la configuración del sistema

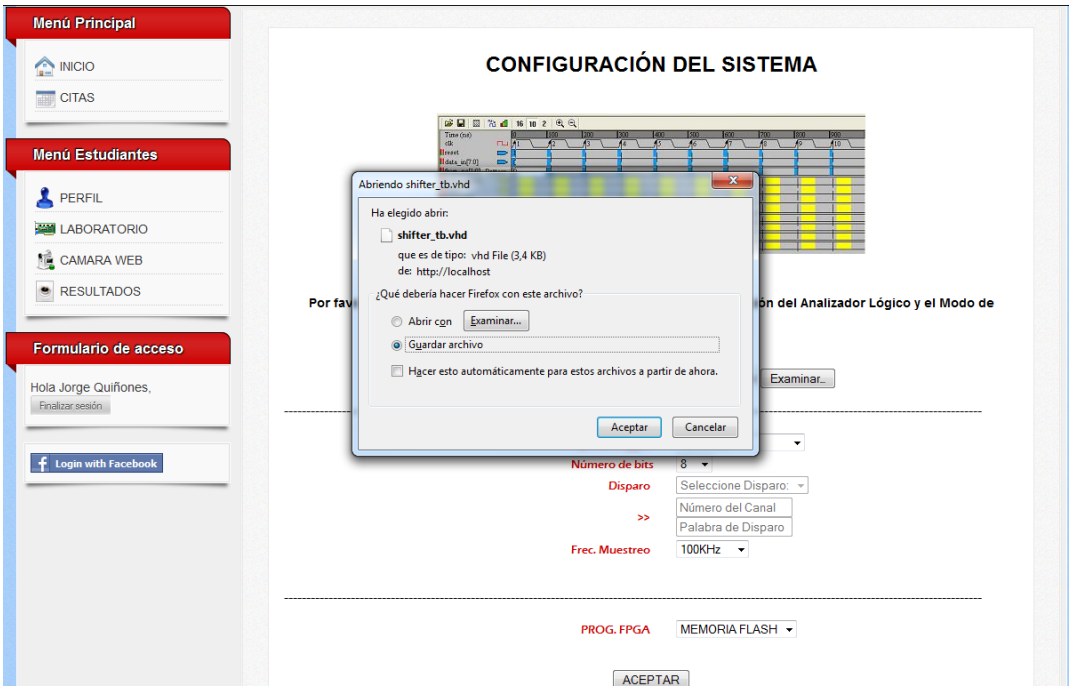


Figura 6.20 Aplicación para descargar archivo *TestBench* modificado por el sistema

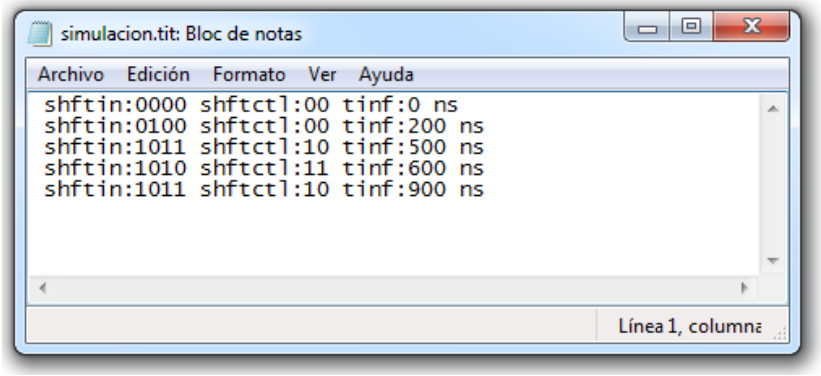


Figura 6.21 Formato del archivo que entrega el *ISE Design* para la generación de estímulos

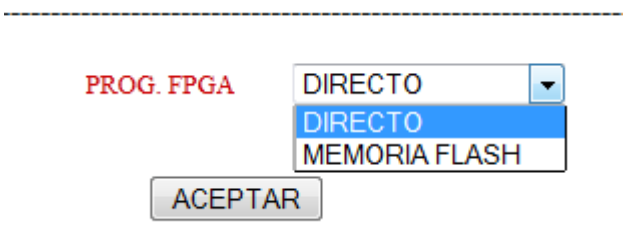


Figura 6.22 Opciones de configuración de la FPGA

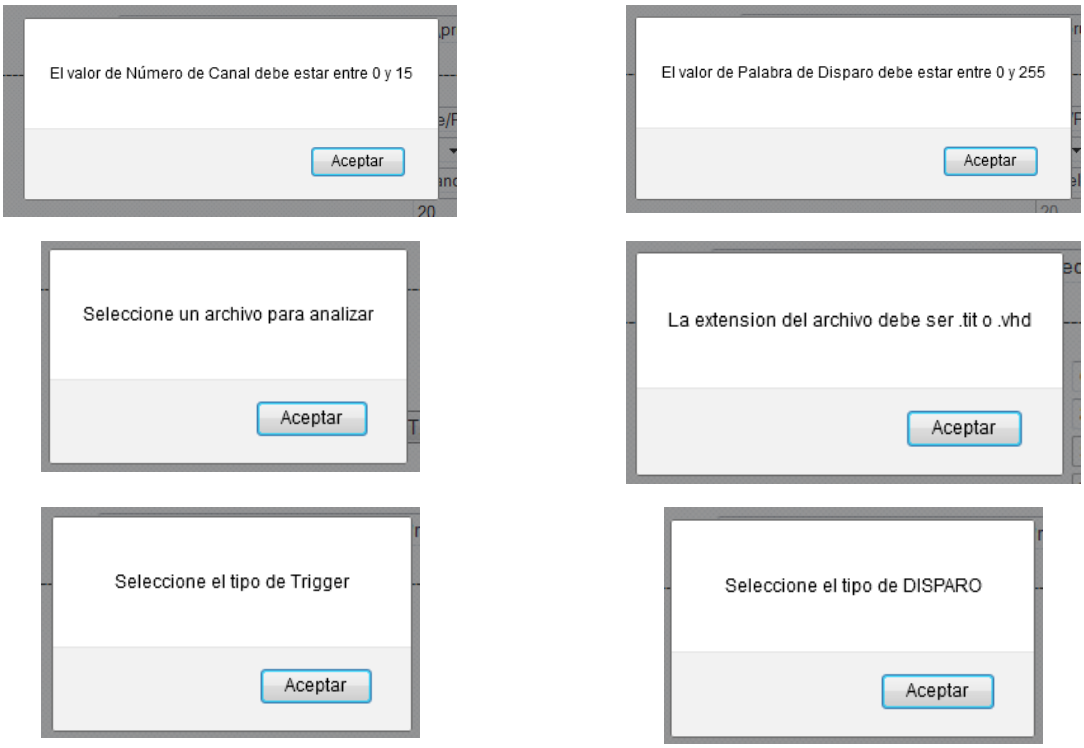


Figura 6.23Mensajes de alerta proporcionado por la página de configuración del sistema para evitar errores

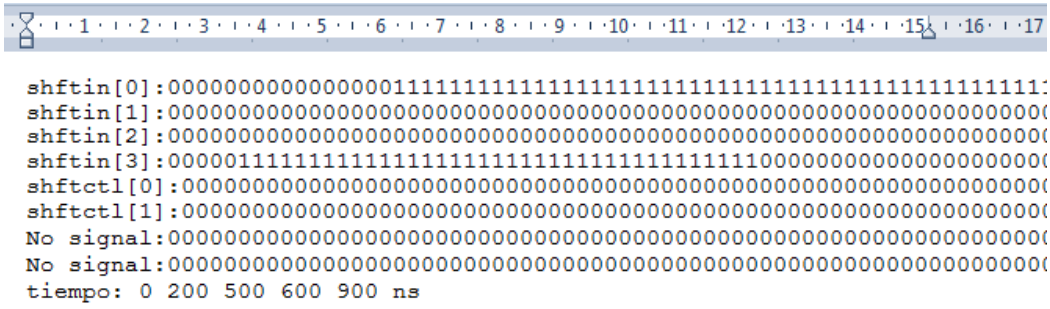


Figura 6.24 Formato del archivo procesado por el sistema para la generación de estímulos a partir del archivo de extensión *.tit*

Figura 6.25 Página presentada al usuario con la que se procede en la ejecución del proceso de pruebas

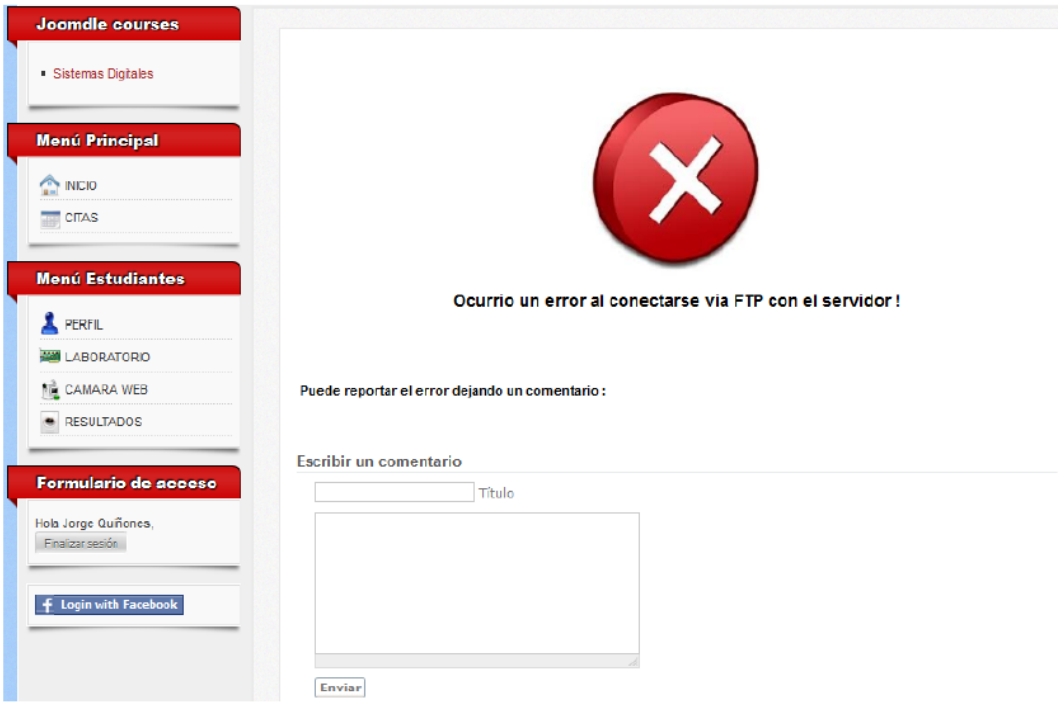


Figura 6.26 Página que se presenta si ocurre un error con la transferencia vía FTP

Figura 6.27 Página que se presenta si el hardware no se encuentra disponible

Figura 6.28 Página de finalización exitosa del proceso

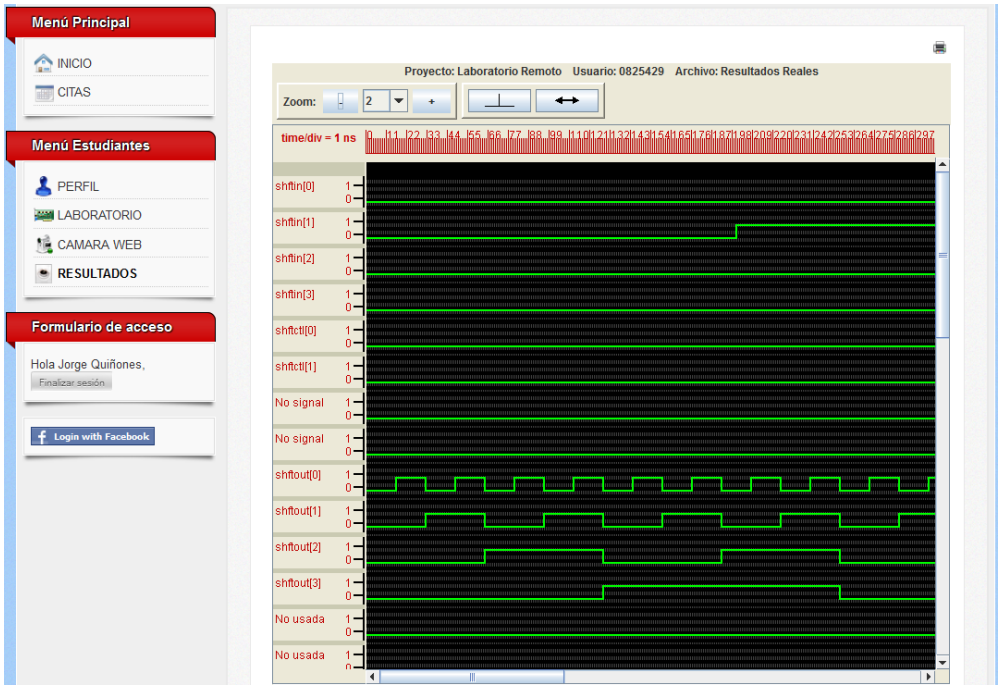


Figura 6.29 Página que carga el *applet* para la visualización de los resultados obtenidos

6.1.2 Resultados de Funcionalidad de la Aplicación

Con el fin de evaluar la funcionalidad de la aplicación desarrollada se usa un conjunto de herramientas de usabilidad [121]que definen la pertinencia de los servicios ofrecidos por la plataforma y su operatividad.

La usabilidad es la cualidad que tiene un sistema por la que permite a sus usuarios alcanzar objetivos específicos (como escribir una carta o enviar un mensaje) con efectividad, eficiencia y satisfacción. Es decir, que cuanto mejor permita hacer algo un sistema mayor usabilidad tendrá. Si el sistema ayuda a que el usuario incurra en la menor cantidad de errores o se recupera de ellos fácilmente, si permite hacer la tarea lo más rápidamente posible y además el usuario queda satisfecho con la labor realizada, el sistema tiene una buena usabilidad [122]. Con la aplicación, el objetivo es que esté diseñado como una herramienta que encaja con la forma de trabajar del usuario y le permita realizar aquello que éste pretende hacer de la mejor forma posible, dando soporte a docentes y estudiantes dentro de un entorno educativo.

En el Anexo F se presenta el modelo de encuesta utilizado, que fue aplicado inicialmente sobre un grupo de 15 usuarios entre estudiantes y docentes. No obstante, antes de la prueba se ha realizado una inducción sobre el manejo básico de la plataforma.

Analizando los resultados obtenidos con la encuesta (Ver Figura 6.30), se observa una perspectiva positiva de parte de los usuarios en cuanto a la interfaz gráfica y operatividad. No obstante, se sugieren mejoras en la claridad de los títulos de las secciones y formularios de la aplicación Web, que deberán ser explicados o reformulados para ser más intuitivo. La dependencia de *JavaScript* es definitiva y existe un proyecto de migración a *html5* para el soporte a plataformas móviles y mejora en la potencia demandada a los navegadores de los clientes.

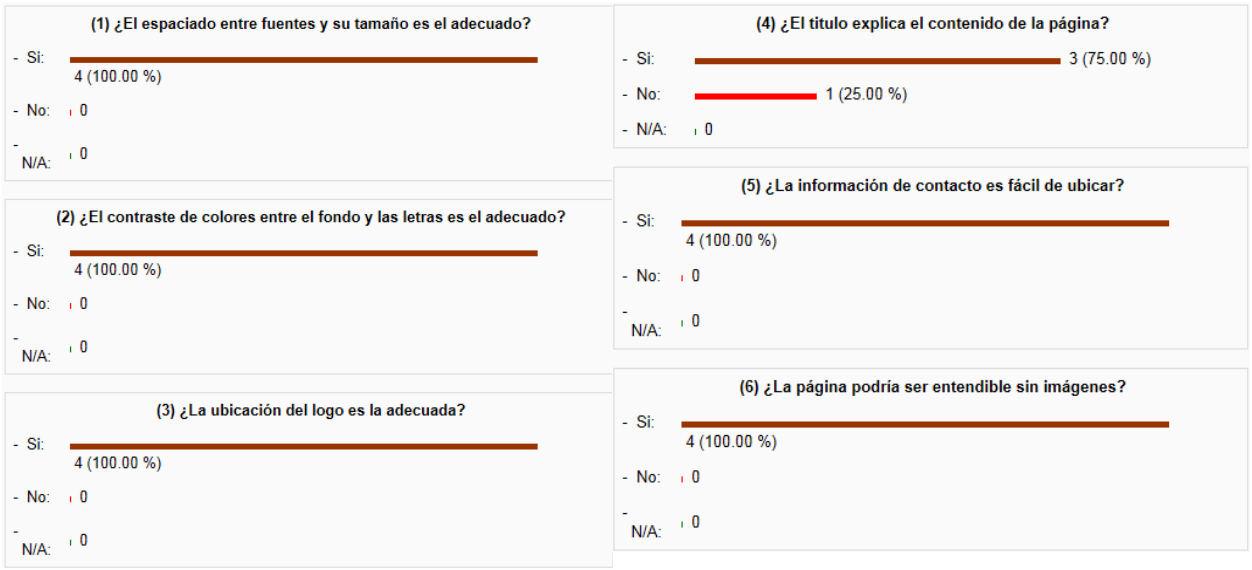


Figura 6.30 Resultado de prueba de usabilidad

6.2 Resultados del Desarrollo de la Placa de Experimentación

En esta sección se presentan los resultados obtenidos en cada una de las etapas de desarrollo de la plataforma *hardware*.

De acuerdo con la metodología adoptada, se realizan implementaciones modulares que debieron ser validadas funcionalmente para dar continuidad y soporte al desarrollo de los subsistemas más complejos. A continuación se describe el proceso realizado en cada uno de estos módulos.

6.2.1 Placa de prueba para fuentes de alimentación

Las placas de prueba correspondientes a las fuentes de alimentación del numeral 4.2.3.1.1 fueron sometidas a un proceso de verificación de tensiones intermedias y de salida, de acuerdo con los requerimientos del diseño. A pesar de que el módulo de alimentación de salida 3.3V/1.8V se diseñó atendiendo a las sugerencias técnicas de *layout* del fabricante, presentó inconvenientes en su primera versión, debido a un ajuste de tierras en el ensamble de los reguladores de conmutación integrados, y los niveles de tensión en la salida eran insuficientes, ubicándose por debajo del 50% de la tensión nominal. La placa de la Figura 6.31 es el resultado de este proceso, en el que se verificaron algunos puntos flotantes que debieron ser conectados a tierra y posteriormente fueron extraídos sus componentes para una prueba local.

Figura 6.31 Placa de prueba de fuente de alimentación 3.3V/1.8V defectuosa

La tensión de 3.3V es una de las de mayor demanda en los circuitos que conforman la tarjeta, pero los requerimientos de regulación son significativamente exigentes (con tolerancias del orden de 5%), por lo que se ha estimado conveniente la adaptación de un circuito similar con dos salidas de ese nivel, lo que le permitiría manejar mayor capacidad de corriente. Metodológicamente, esto se ha llevado a cabo mediante el rediseño del diagrama esquemático (Ver Figura 6.32), diseño del PCB y montaje de los componentes, estos últimos correspondían en su mayoría con los de la primera versión del módulo (Ver Figura 6.33).

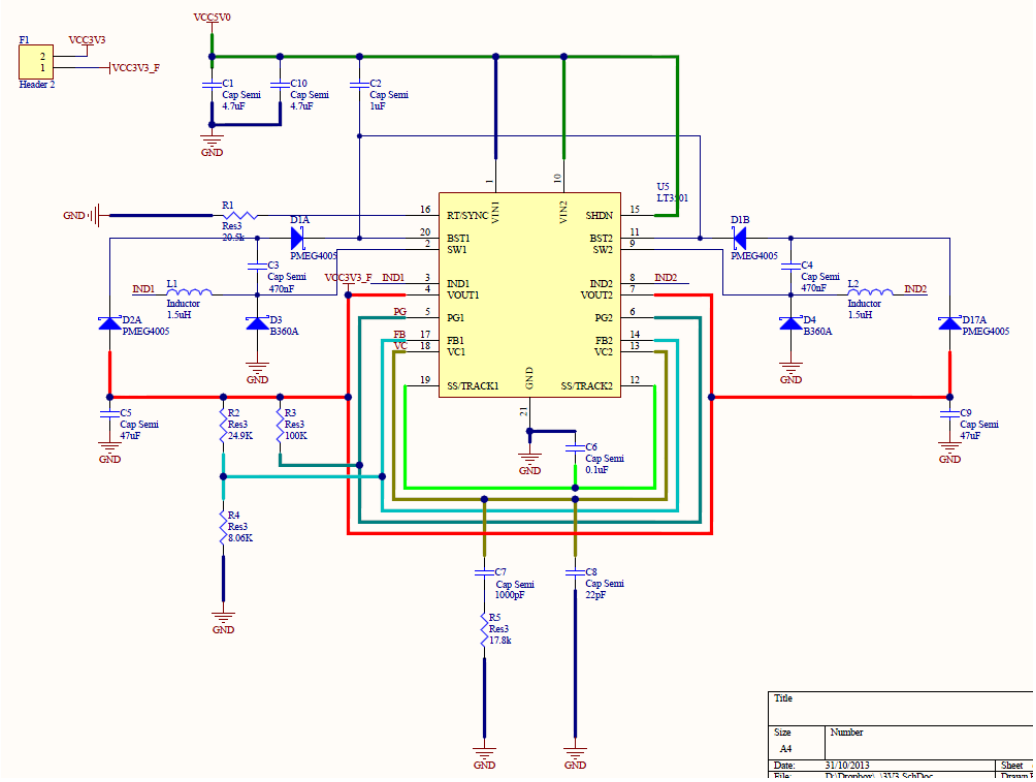


Figura 6.32 Diagrama esquemático modificado para fuente de 3.3V

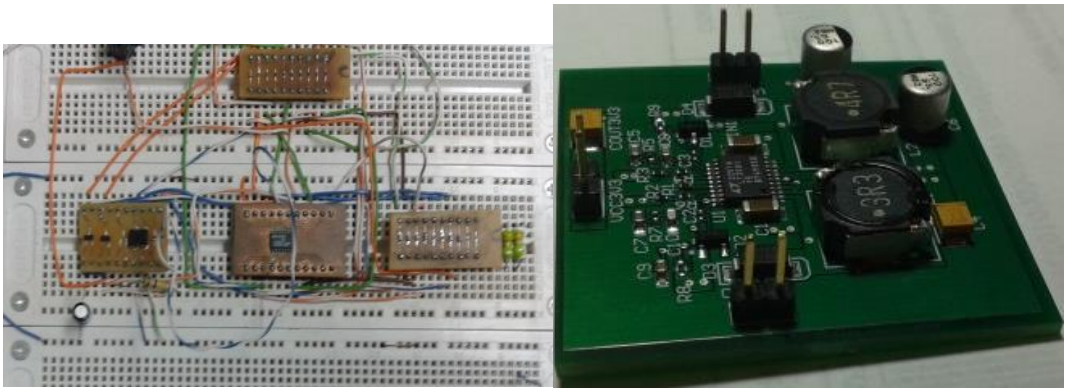


Figura 6.33 Pruebas locales y placa de prueba corregida

Tabla 6.1 Relación Voltaje/Corriente en Fuente de 3.3V

Voltaje [V]	***Corriente[mA]***
3.30	0 (Abierto)
3.29	10
3.22	190

3.18 400

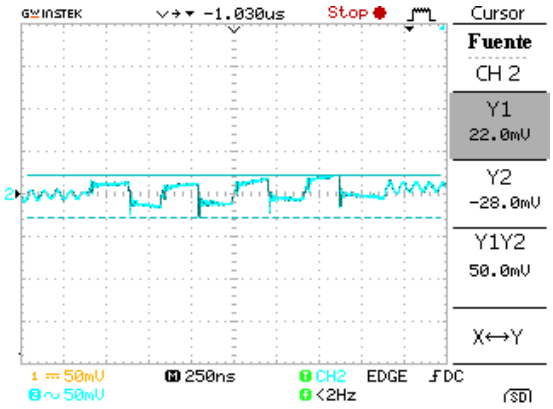


Figura 6.34 Máximo rizado de la señal de voltaje de 3.3 V a una corriente superior a 400mA

Para verificar la regulación de la fuente de 3.3V, que soporta la mayor carga en el sistema propuesto, se midió la relación de Corriente/Voltaje y estabilidad de la señal para diferentes cargar resistivas. Esto se muestra en la Tabla 6.1 y la Figura 6.34. El rizado debe verificarse en los *pads* de cada dispositivo que demanda el suministro de esta fuente, pues se han fijado capacitores de desacoplo cerca de ellos. No obstante, los niveles de tensión se encuentran en el rango de trabajo de los dispositivos de la placa.

Teniendo en cuenta la Tabla 6.2, el consumo esperado de corriente es menor a 400mA con un voltaje suministrado a la tarjeta por encima de 3.18V, lo que garantizaría la polarización correcta para el funcionamiento de todos los componentes.

El consumo de carga para las fuentes de 1.2 V, 1.8 V y 2.5 V es menor debido a que son voltajes para funciones específicas en los dispositivos de la FPGA y del CPLD. En la Tabla 6.3 se muestra el consumo estimado y la variación de voltaje/corriente medida en las pruebas.

Tabla 6.2 Cargas estimadas de los dispositivos principales en la tarjeta

Dispositivo		**Corriente**
FPGA (VCCO)	LX9	1mA
	LX45T	3mA
uC PIC24EP		80mA
CPLD XC2C256		45.54mA
SRAM IS61WV102416		50mA
Mem. Flash M25P16		15mA
12 Leds		60mA
LCD LCM1602A		4mA

Total	**LX9**	**255.54mA**
	LX45T	**257.54mA**

Tabla 6.3 Relación Voltaje/corriente y máxima corriente estimada para fuentes secundarias

Fuente		Max Corriente Estimada	V/I de prueba máx.
1.2V FPGA (VCCINT)	LX9	4mA	1.18V/100mA
	LX45T	15mA	
1.8 V CPLD (VCCAUX)		19.4mA	1.78V/100mA
2.5V FPGA (VCCAUX)	LX9	2.5mA	2.44V/0.8A
	LX45T	5mA	

6.2.2 Placas de prueba para validar las funciones del planificador

Para la validación del diseño esquemático del módulo de microcontrolador se implementa una tarjeta de 4 capas con soporte de conexión USB y conectores de expansión a los demás módulos dispuestos en puertos para la medición. Para el proceso de fabricación de la tarjeta se contrata con una empresa nacional y el proyecto se ajusta a sus especificaciones técnicas de capacidad de fabricación. En la Figura 6.35 se muestra la placa fabricada, en la que algunos de los elementos se han retirado para reutilizarse en una nueva tarjeta, ya que se detectó un defecto de fabricación en un corto entre una capa de señal y un polígono de un plano de alimentación interno, a través de un pin *through-hole*.

Debido a los inconvenientes de fabricación, se diseñó una tarjeta de dos capas con el mismo propósito para el microcontrolador (Figura 6.36). Las pruebas que se realizan en esta tarjeta deben validar el correcto funcionamiento del microcontrolador en la fase de programación y ejecución de sus tareas como se muestra en Figura 6.38. Mediante la interconexión de las tarjetas de prueba de alimentación, microcontrolador y FPGA (Ver Figura 6.37) se deben evaluar las funciones del planificador de tareas, usando un analizador lógico, para verificar la temporización de las señales de control, y el *debugger* de *pickit 3* para la depuración del sistema programado. En la Figura 6.38 se muestra el proceso de reconocimiento y programación de la tarjeta con microcontrolador en la herramienta MPLAB 8; de esta forma se comprueba el funcionamiento de las conexiones para el inicio y configuración del microcontrolador.

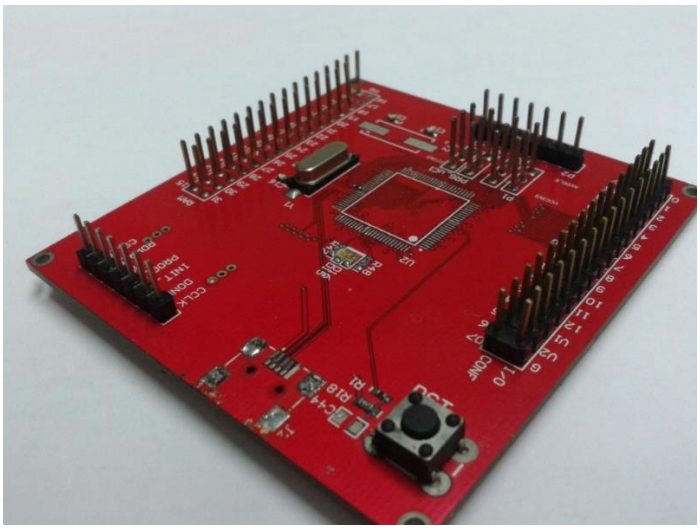


Figura 6.35 Tarjeta de operación general de microcontrolador con defectos de fabricación

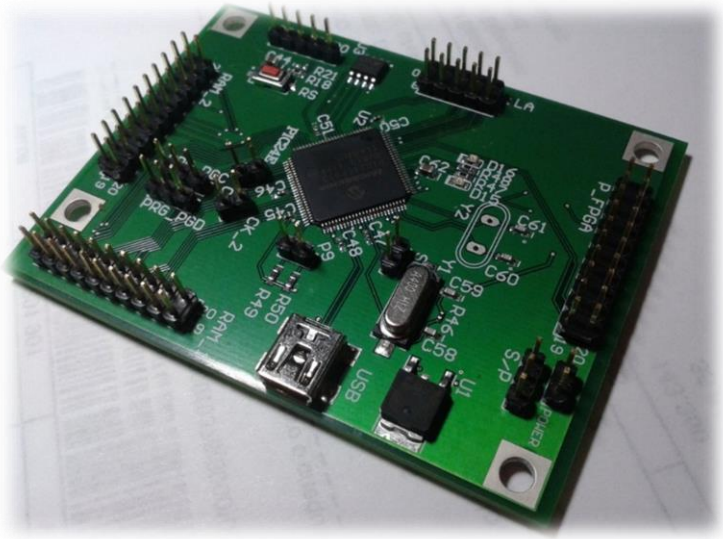


Figura 6.36 Tarjeta funcional para pruebas con el microcontrolador

Posteriormente, se valida la ejecución del planificador y la comunicación USB. Para ello, se diseñan 2 tareas básicas llamadas "*Mem*" y "*Check*" que encendieran los *leds* dispuestos en la tarjeta, la tarea "USB" encargada de inicializar la comunicación USB y la tarea "FPGA" que se encargara de enviar paquetes de 8-bits por el puerto *SelectMAP*; los resultados son exitosos como lo muestra la herramienta *RTOSviewer* de MPLAB en la Figura 6.39, obtenida en la depuración con el programador *pickit 3*. Además, como se muestra en la Figura 6.40, la inicialización del USB es exitosa y se logra la conexión entre el PC y la tarjeta.

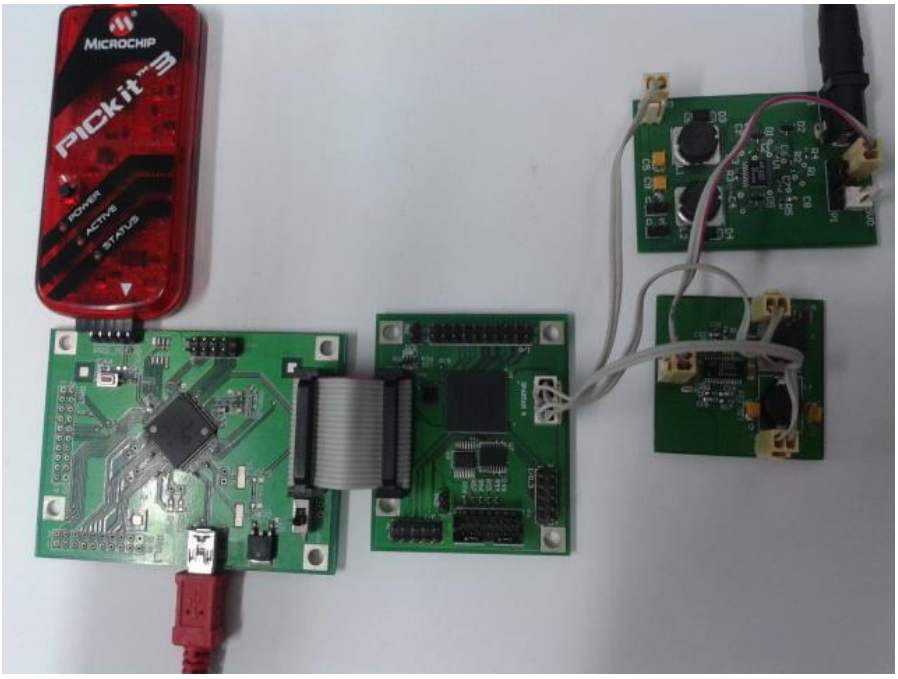


Figura 6.37 Conexión de módulos de prueba para validar la configuración del FPGA

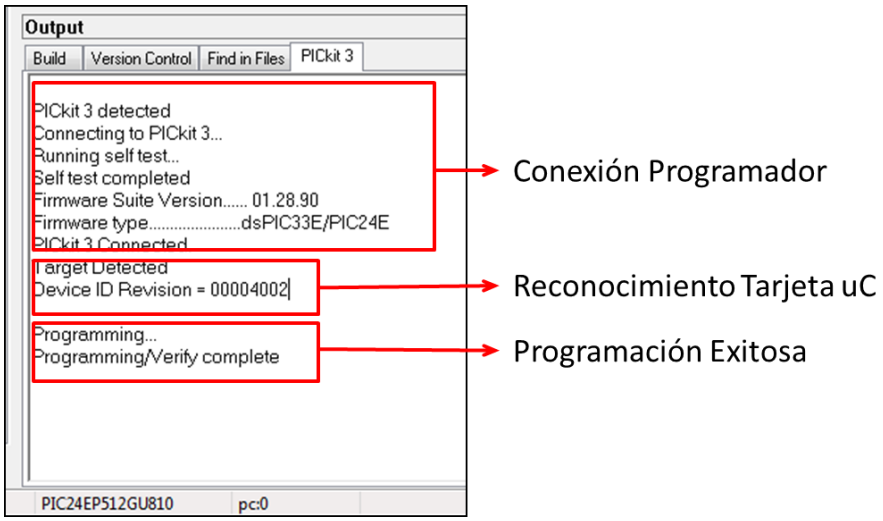


Figura 6.38 Conexión de la tarjeta con la herramienta MPLAB a través del programador *pickit 3*

Para validar la tarea de programación del FPGA, una aplicación *java* en el PC envía el archivo *.bit* para realizar la configuración usando la interfaz *SelectMAP*, siendo el método más rápido entre los soportados en la tarjeta para cargar el archivo al FPGA. La tarea "FPGA" toma la carga útil del archivo *.bit* y lo envía por el puerto *SelectMAP* para cargar los datos al dispositivo, como se describe en la sección 4.1.1.1.13.1. En la Figura 6.41 se muestra el envío del archivo de configuración por este puerto.

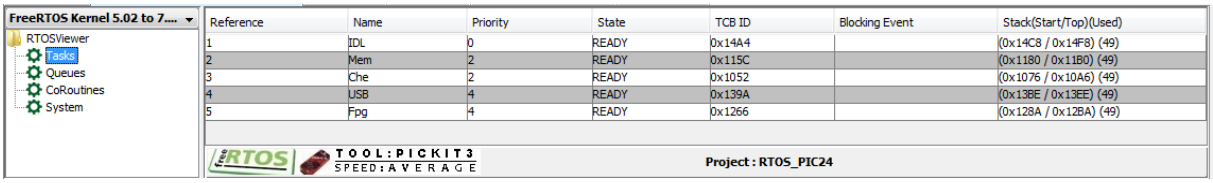


Figura 6.39 Análisis de ejecución del planificador con *RTOSviewer*

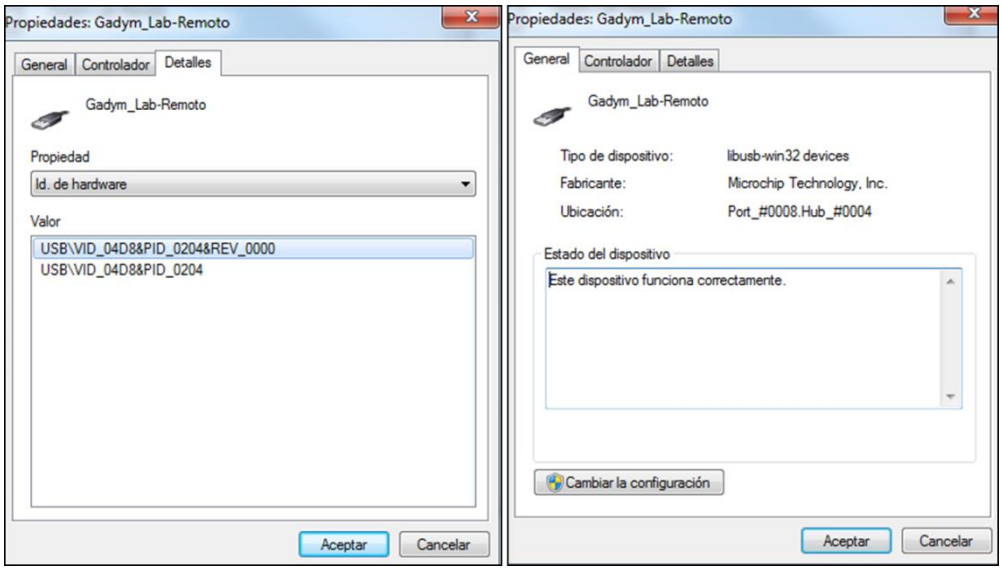


Figura 6.40 Conexión USB entre la tarjeta y el PC

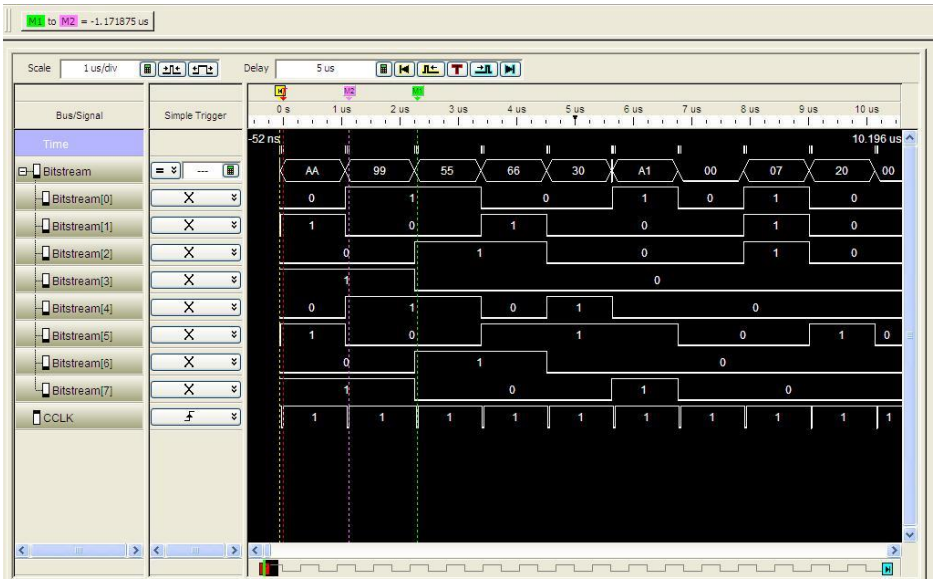


Figura 6.41 Datos de configuración en el puerto *SelectMAP* y CCLK

Para verificar el envío de los datos necesarios para la configuración y su correspondencia, se realiza el conteo de los *bytes* enviados por el puerto *SelectMAP*, iniciando desde la palabra de sincronización hasta encontrar la palabra *DESYNC-WORD* del *bitstream* (Ver sección 4.1.1.1.13.3) y el resultado del conteo se compara con el número de *bytes* de carga útil en el *bitstream.* El número total de bytes en la carga útil del *bitstream* se obtiene con la visualización del contendido del archivo *.bit* para la FPGA XC6LX9, con la herramienta *binviewer 2.0* (Ver Figura 6.42).

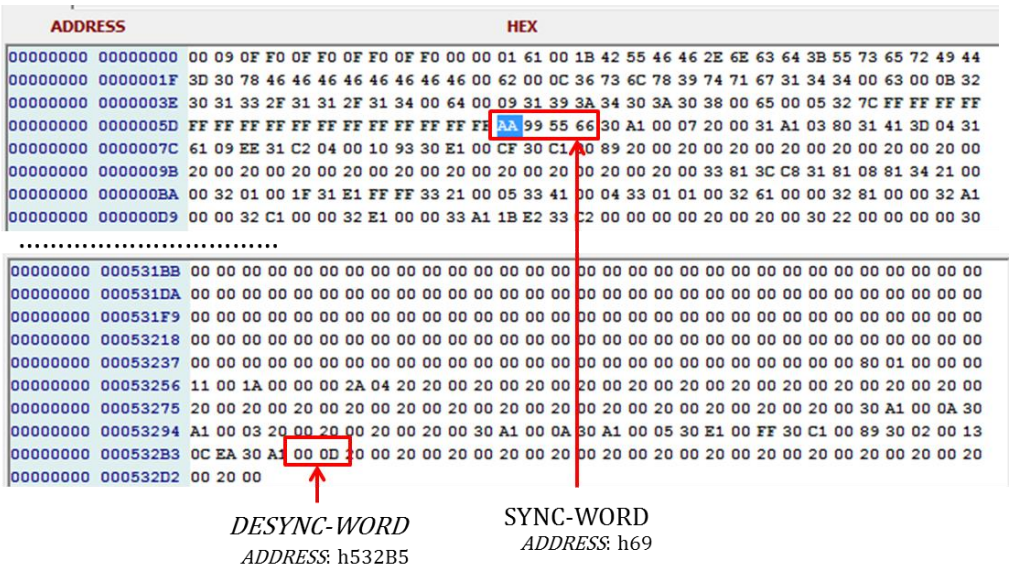


Figura 6.42 Datos de *bitstream* vistos con *binviewer* 2.0

Durante la prueba se detectó que, cuando transcurre el proceso de envío de datos al FPGA, mientras se presenta el cambio de contexto en el planificador de tareas, se dejaba de leer una trama de 64 bytes, lo que entregaba un *bitstream* incompleto a la interfaz de configuración del FPGA. Esto se reportaba como una

falla en la programación, después de advertir la alerta del pin INIT_B por un error en el Chequeo de Redundancia Cíclica (CRC) posterior a la llegada de la palabra *DESYNC_WORD*.

Este problema estaba ligado directamente a la prioridad de la tarea encargada de la programación de la FPGA, lo que sugiere que esta tarea sea de máxima prioridad durante su ejecución y que se suspenda a las tareas que no requieran de tomar el control durante este proceso o se disminuya su prioridad. Además, resulta conveniente explotar una mayor velocidad de reloj en el microcontrolador con el uso del PLL interno, pasando desde un reloj interno de 8MHz a un reloj de 80 MHz para su funcionamiento.

6.2.3 Tarjeta Prototipo con FPGA y Microcontrolador

La tarjeta propuesta en la sección 4.2.3, con los componentes necesarios para validar la configuración y funcionamiento de la FPGA con los respectivos puertos de E/S, fue fabricada por una firma extranjera y presento fallas posteriores al montaje de los componentes (ver Figura 6.43) [106]. Estas fallas hicieron más difícil la validación del sistema y requirió la reparación parcial de vías que aumentaban su resistencia en cada prueba. Por tanto, los trazos aislados por defectos en la fabricación y ensamble fueron sustituidos por conductores externos dispuestos manualmente, aunque esto comprometía la integridad de la comunicación USB, programación de microcontrolador y programación del FPGA.

Este prototipo usa principalmente el modo de configuración *SelectMAP*, por ser el método más rápido y con soporte para reconfiguración parcial, evacuando los datos que llegan desde el servidor por un puerto de 8-bit, a diferencia del método *slave-serial* que evacua por un canal de 1-bit. La configuración exitosa se da cuando la señal DONE se establece en alto, como se muestra en la Figura 6.43. La primera función programada en el FPGA es un *buffer*, cuya entrada se conecta a uno de los botones disponibles en la tarjeta y su salida se lleva a un *led* de la misma. Así mismo, se implementó un divisor de frecuencia que permitiera la validación de interfaces de reloj y bloques secuenciales que fueron probadas con una frecuencia de reloj de hasta 200MHz desde el lado del FPGA.

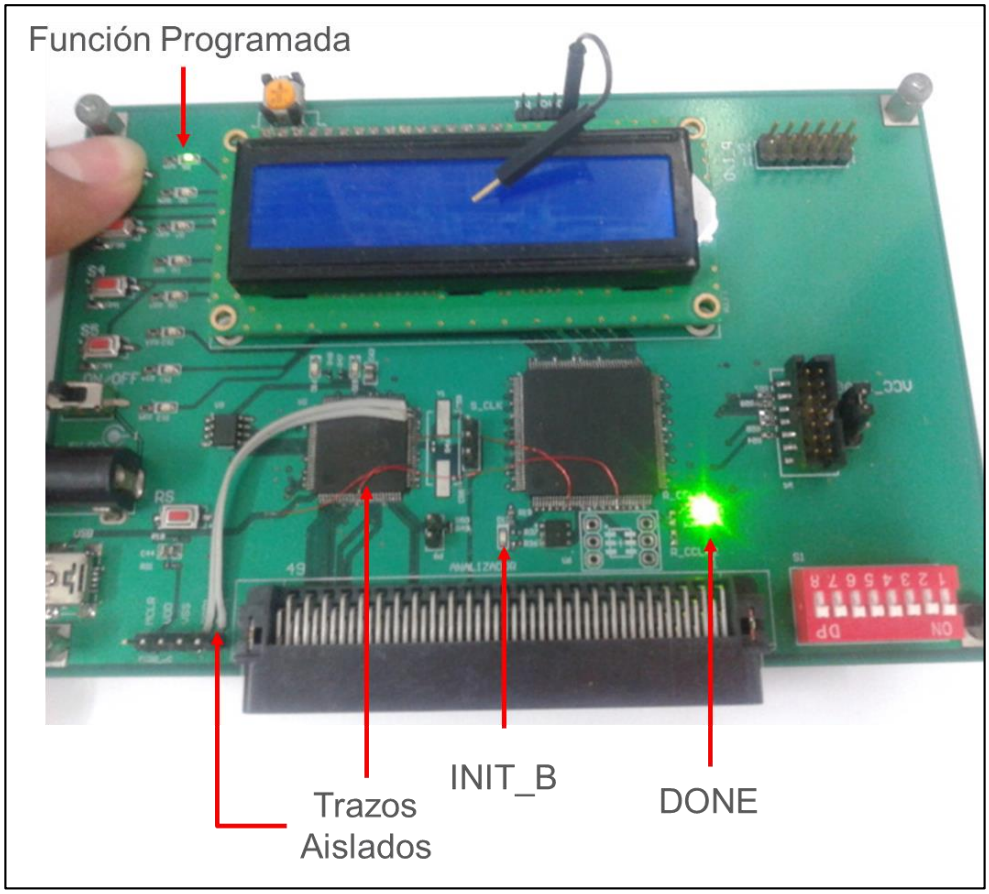


Figura 6.43 Prueba de configuración en Tarjeta con microcontrolador y FPGA

6.2.4 Tarjeta Prototipo con CPLD y SRAM

Las pruebas realizadas para la validación del funcionamiento en esta tarjeta corresponden al funcionamiento del analizador lógico, cuyo diseño se presenta en el capítulo 5. No obstante, antes deben verificarse la programación del CPLD, el direccionamiento de memoria y comunicación con el microcontrolador.

6.2.4.1 Programación CPLD

La programación del CPLD se hace mediante interfaz JTAG (ver Figura 6.44 - izquierda), la cual fue considerada en el diseño de la tarjeta y es utilizada solo para esta función. Después de la programación del CPLD, es posible hacer las pruebas pertinentes para validar las funciones del analizador lógico con la memoria de muestreo y con el microcontrolador, como se describe en las secciones 6.2.4.1.1 y 6.2.4.1.2.

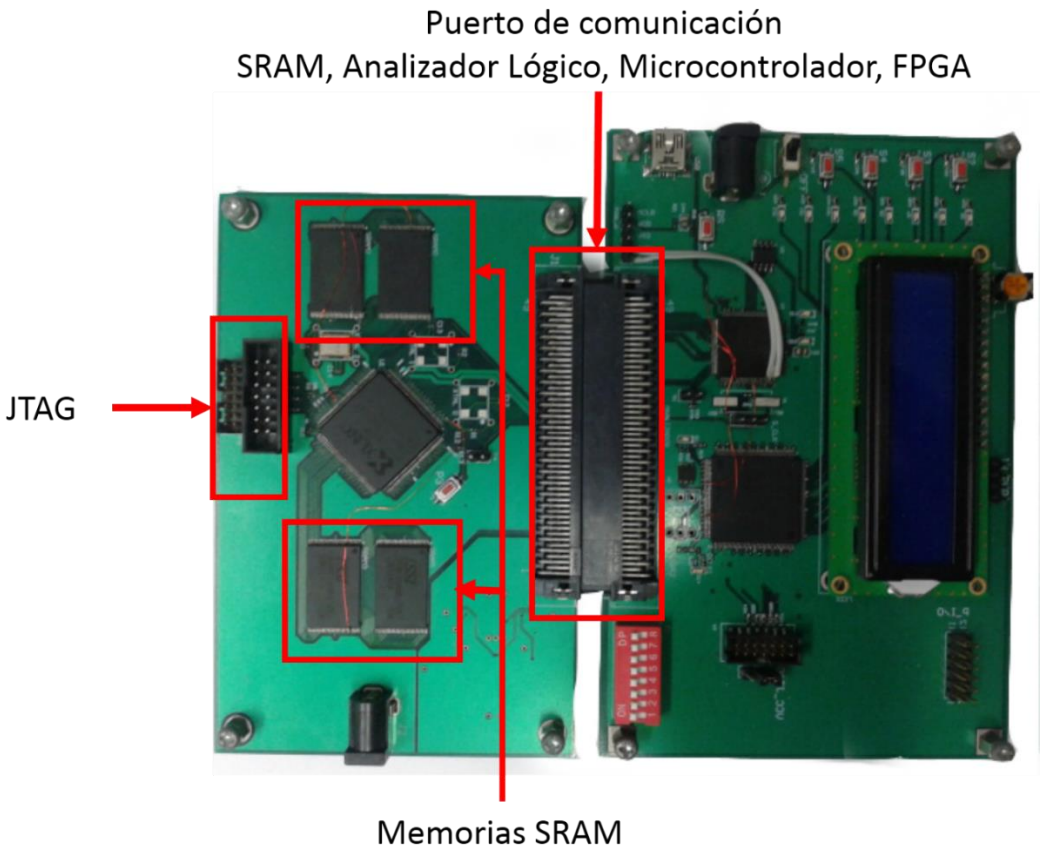


Figura 6.44 Tarjeta de CPLD y Memorias SRAM acopladas al FPGA y microcontrolador

6.2.4.1.1 Direccionamiento en memoria de muestreo

El espacio de direccionamiento para la memoria de muestreo es de 8M y la plataforma de experimentación tiene implementados 4 chips de 2MB. La decodificación de estas unidades de memoria se hace desde un bloque lógico implementado provisionalmente en el CPLD. Este bloque optimiza la capacidad de direccionamiento en función del número de canales seleccionado por el usuario para el analizador lógico. En la Figura 6.45 puede apreciarse el diagrama general de verificación del bloque de SRAM.

Los chips que conforman la unidad de memoria se relacionan como BANK0, BANK1, BANK2 y BANK3. Ellos tienen un bus de datos de 16 líneas y una capacidad de direccionamiento de 1Mx16 (2MB). Además de sus señales típicas de control WR (escritura), CS (Selección de Chip), OE (Habilitación de Salida), estos elementos contienen dos señales para el acceso independiente a los bytes de cada Word. La activación de la señal LB permite acceder al *byte* menos significativo, mientras la línea UB lo hace con el *byte* más alto. Las partes no seleccionadas se colocan en alta impedancia.

El mapa de direccionamiento se diseña conforme a la Tabla 6.4. El bus BD, relacionado en la tabla, es una señal interna del CPLD que hace referencia al número de canales a analizar solicitado por el usuario y hace parte importante de la decodificación, pues junto con las líneas más altas del bus de direcciones, define la activación de las señales de control CS, LB y UB de cada chip de memoria. Es importante anotar que el nivel de activación de éstas es bajo.

Para el almacenamiento de muestras de 32 bit deben seleccionarse un par de chips de memoria por cada dirección, con excepción del área no implementada, en la que no se habilita ningún banco. Si el usuario

selecciona 32 canales para adquisición del analizador lógico, el espacio de direccionamiento total es de 2M. Cuando el usuario selecciona 16 canales, el espacio de direccionamiento es de 4M. Para 8 canales, se selecciona el *byte* más alto o el más bajo de cada *chip*, según sea el caso, y se obtienen 8M en total.

El proceso de verificación de este bloque fue asistido por un analizador lógico externo de 16 canales. En este proceso, se implementaron provisionalmente en el CPLD un multiplexor 8-1 de 16 bit, con palabras fijas de entrada; un contador de 3 bits, para control del multiplexor y una FSM (*Finite State Machine*) que administra la comunicación asíncrona con el MCU. Estos bloques lógicos usaron apenas un 16% de las macroceldas del CPLD y un 3% de sus registros; su reloj tendría una frecuencia máxima de 286MHz.

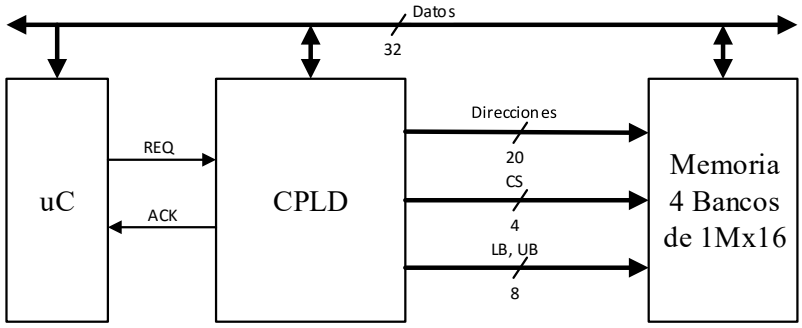


Figura 6.45 Arquitectura de prueba de banco de memoria SRAM

Tabla 6.4 Direccionamiento a memoria de muestreo

Número de Canales	BD[1:0]	Address[22:20]	[C0 LB0 UB0]	[C1 LB1 UB1]	[C2 LB2 UB2]	[C3 LB3 UB3]
32	00	000	000	000	1XX	1XX
	00	001	1XX	1XX	000	000
	00	01X	1XX	1XX	1XX	1XX
	00	1XX	1XX	1XX	1XX	1XX
16	01	000	000	1XX	1XX	1XX
	01	001	1XX	000	1XX	1XX
	01	010	1XX	1XX	000	1XX
	01	011	1XX	1XX	1XX	000
	01	1XX	1XX	1XX	1XX	1XX
8	10	000	001	1XX	1XX	1XX
	10	001	010	1XX	1XX	1XX
	10	010	1XX	001	1XX	1XX
	10	011	1XX	010	1XX	1XX
	10	100	1XX	1XX	001	1XX
	10	101	1XX	1XX	010	1XX
	10	110	1XX	1XX	1XX	001
	10	111	1XX	1XX	1XX	010

6.2.4.1.2 Comunicación microcontrolador – Analizador lógico

El microcontrolador administra la comunicación asíncrona con el analizador lógico, con base en el protocolo *Hanshake 4-phase* (Ver Figura 6.46). Esta comunicación se hace a través de instrucciones

enviadas desde el microcontrolador, con un formato de 8-bit que se transfieren paralelamente por cada ciclo de REQ/ACK. Algunas de estas instrucciones envían parámetros de configuración en la misma transferencia y otras lo hacen en los ciclos contiguos.

El microcontrolador envía los datos para configurar el modo de funcionamiento del analizador lógico después de reconocer la palabra 0xBB de sincronización con el servidor. Posteriormente, el microcontrolador envía los estímulos bajo el mismo protocolo. La trama de estímulos se envía *byte* a *byte* después de que el microcontrolador reconozca la palabra de 0xCC, enviada desde el Servidor. Esto se hace conforme a lo establecido en la Figura 4.23.

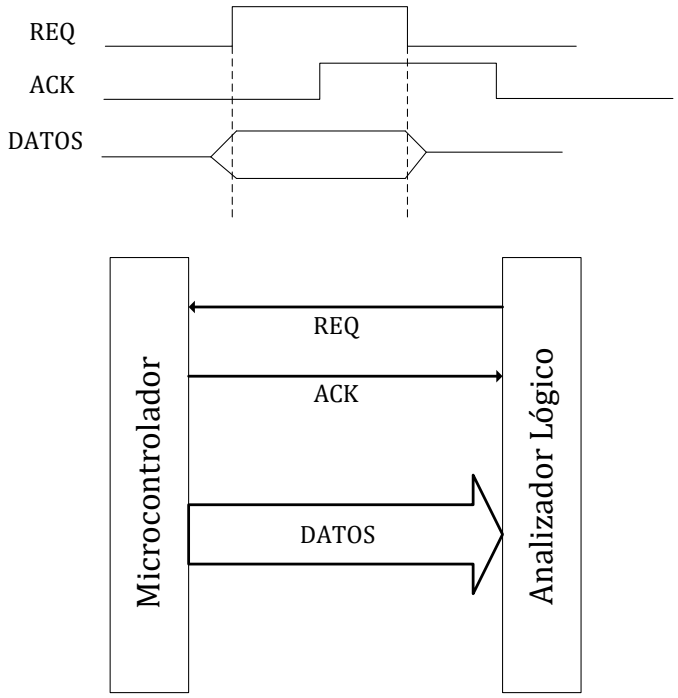


Figura 6.46 Comunicación con protocolo *4-phase handshake* entre microcontrolador y analizador lógico

29	URB	16.556936	BULK_OR_INTE...	OUT	1	\Device\USBPDO-5	0x8A29F1...	STATUS_SUCCESS	BB	1
30	URB	16.556937	BULK_OR_INTE...	OUT	1	\Device\libusb00001	0x8A29F1...	STATUS_SUCCESS	BB	1
31	URB	16.556960	BULK_OR_INTE...	IN	1	\Device\libusb00001	0x8A29F1...	STATUS_PENDING		0
32	URB	16.557081	BULK_OR_INTE...	IN	1	\Device\libusb00001	0x8A29F1...	STATUS_SUCCESS		0
33	URB	16.557146	BULK_OR_INTE...	OUT	1	\Device\USBPDO-5	0x8A29F1...	STATUS_SUCCESS	04 20 03 0A 00 00 FF FF	8
34	URB	16.557147	BULK_OR_INTE...	OUT	1	\Device\libusb00001	0x8A29F1...	STATUS_SUCCESS	04 20 03 0A 00 00 FF FF	8
35	URB	16.557169	BULK_OR_INTE...	IN	1	\Device\libusb00001	0x8A29F1...	STATUS_PENDING		0
36	URB	16.557333	BULK_OR_INTE...	IN	1	\Device\libusb00001	0x8A29F1...	STATUS_SUCCESS		0

Comando 0xBB	Tipo de Trigger	Numero de bits	F. De muestreo	Tipo de disparo	Pal. Disparo	Pal. Disparo	Pal. Disparo	Pal. Disparo	Tipo de Muestro
Comando 0xBB	Pre/Pos	16 bits	Fosc	Disparo por Nivel	0xFF	0xFF	0x00	0x00	Transicional

Figura 6.47 Trama de prueba para configuración de funciones en analizador lógico

6.2.4.1.3 Configuración de Funciones al Analizador Lógico

Cuando se reconoce la palabra 0xBB desde el servidor, la tarea de decodificación del RTOS crea una nueva tarea para la configuración y elimina la de configuración del FPGA, quedando como tarea de mayor prioridad en el sistema y tomando el control de la comunicación USB.

En la Figura 6.47 se muestra una trama enviada desde el servidor, para la que se ha seleccionado la siguiente configuración:

- Activacion de disparo= Pre/Post
- Numero de Bits= 32 bits
- Tipo de Disparo= Nivel
- Palabra de Disparo= 0x0000FFFF
- Frecuencia= Fosc

La relación entre estos parámetros y el valor asociado dentro de la trama es mostrada en la sección 3.4.8.2.1

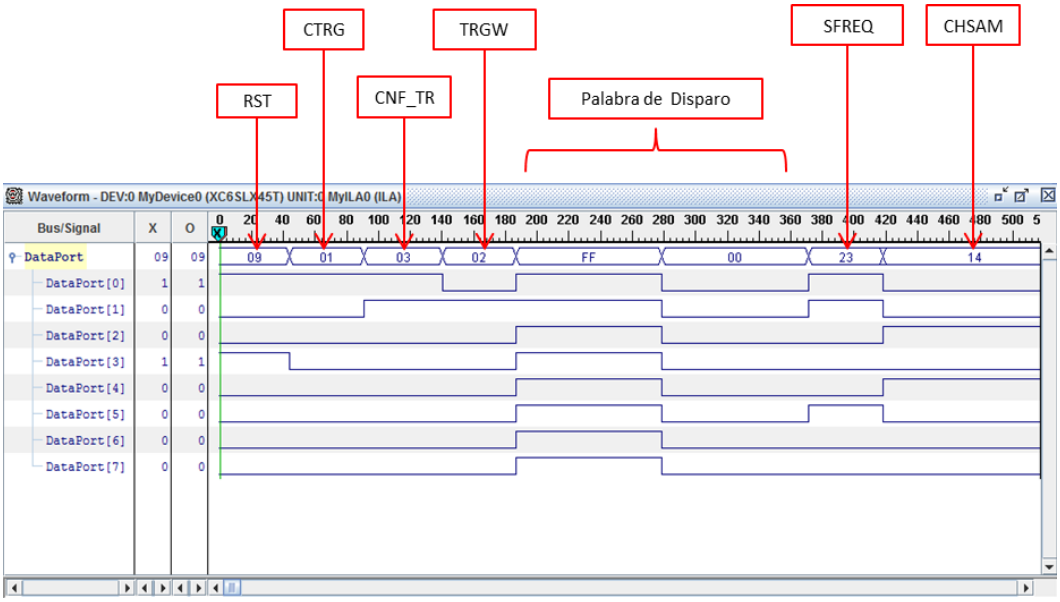


Figura 6.48 Captura de Trama de comandos y parámetros enviados por el microcontrolador para la configuración del analizador lógico con *ChipScope* de *Xilinx*

37	URB	15.274345	BULK_OR_INTE...	OUT	1	\Device\USBPDO-6	0xDE649...	STATUS_SUCCE...	CC	1
38	URB	15.274351	BULK_OR_INTE...	OUT	1	\Device\libusb00001	0xDE649...	STATUS_SUCCE...	CC	1
39	URB	15.274400	BULK_OR_INTE...	IN	1	\Device\libusb00001	0xDE649...	STATUS_PENDI...		0
40	URB	15.274479	BULK_OR_INTE...	IN	1	\Device\libusb00001	0xDE649...	STATUS_SUCCE...		0
41	URB	15.274527	BULK_OR_INTE...	OUT	1	\Device\USBPDO-6	0xDE649...	STATUS_SUCCE...	FF 55 44 33 22 11 AA 77 99 55	10
42	URB	15.274530	BULK_OR_INTE...	OUT	1	\Device\libusb00001	0xDE649...	STATUS_SUCCE...	FF 55 44 33 22 11 AA 77 99 55	10
43	URB	15.274565	BULK_OR_INTE...	IN	1	\Device\libusb00001	0xDE649...	STATUS_PENDI...		0
44	URB	15.274738	BULK_OR_INTE...	IN	1	\Device\libusb00001	0xDE649...	STATUS_SUCCE...		0

Figura 6.49 Trama USB con el comando y los Estímulos, tomado con *USBTrace*

Mediante la tarjeta de prueba con FPGA *Spartan-6* de la Figura 4.36, se implementa el *ChipScope* de Xilinx para capturar las transferencias del microcontrolador, cuando envía la trama de comandos y parámetros de

configuración del analizador lógico (Ver comandos en capítulo 5). La Figura 6.48 muestra las tramas identificadas.

La trama de estímulos se envía una vez el microcontrolador reconoce la palabra 0xCC como comando enviado desde el servidor. Para ilustrar esto, se ha establecido el servidor para que envíe 10 estímulos en una transacción. La actividad USB queda registrada en la Figura 6.49, mientras las tramas de comandos resultantes se muestran en la Figura 6.50.

Finalmente, para descargar las muestras desde la memoria, el microcontrolador espera el comando 0xDD que inicia el proceso. Una vez detectada esta palabra, se envía el comando RDSMEM al analizador lógico, la tarea creada para este propósito toma el control de la comunicación USB y elimina la tarea del proceso anterior. Este proceso de descarga de datos solicita al analizador lógico el direccionamiento de 4MBytes de datos almacenados en la memoria de muestreo SRAM, para su lectura desde el microcontrolador. Los datos recibidos son empaquetados para el envío hacia el servidor. En la Figura 6.51, se muestra la captura de la trama USB con la herramienta *USBTrace*, en la que se observa el envío del comando 0xDD desde el servidor y la respuesta con los datos de memoria.

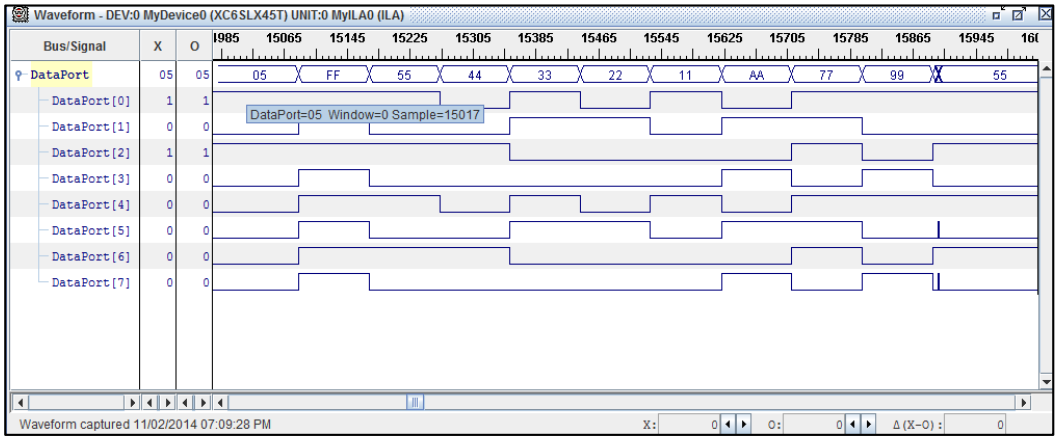


Figura 6.50 Trama de estímulos capturada con *ChipScope* de *Xilinx*

32	URB	12.950849	BULK_OR_INTE...	IN	1	\Device\libusb00001	0x893350...	STATUS_SUCCESS		0
33	URB	13.984988	BULK_OR_INTE...	OUT	1	\Device\USBPDO-6	0x86AB4...	STATUS_SUCCESS	DD	1
34	URB	13.984992	BULK_OR_INTE...	OUT	1	\Device\libusb00001	0x86AB4...	STATUS_SUCCESS	DD	1
35	URB	13.985033	BULK_OR_INTE...	IN	1	\Device\libusb00001	0x86AB4...	STATUS_PENDING		0
36	URB	13.985217	BULK_OR_INTE...	IN	1	\Device\libusb00001	0x86AB4...	STATUS_SUCCESS		0
37	URB	13.985317	BULK_OR_INTE...	OUT	81	\Device\USBPDO-6	0x86AB4...	STATUS_SUCCESS		0
38	URB	13.985319	BULK_OR_INTE...	OUT	81	\Device\libusb00001	0x86AB4...	STATUS_SUCCESS		0
39	URB	13.985348	BULK_OR_INTE...	IN	81	\Device\libusb00001	0x86AB4...	STATUS_PENDING		0
40	URB	13.985707	BULK_OR_INTE...	IN	81	\Device\libusb00001	0x86AB4...	STATUS_SUCCESS	BF 0D 25 70 A5 3C 73 ...	64
41	URB	13.985741	BULK_OR_INTE...	OUT	81	\Device\USBPDO-6	0x867282...	STATUS_SUCCESS		0
42	URB	13.985742	BULK_OR_INTE...	OUT	81	\Device\libusb00001	0x867282...	STATUS_SUCCESS		0
43	URB	13.985779	BULK_OR_INTE...	IN	81	\Device\libusb00001	0x867282...	STATUS_PENDING		0
44	URB	13.986205	BULK_OR_INTE...	IN	81	\Device\libusb00001	0x867282...	STATUS_SUCCESS	BF 0D 25 70 A5 3C 73 ...	64
45	URB	13.986251	BULK_OR_INTE...	OUT	81	\Device\USBPDO-6	0x867282...	STATUS_SUCCESS		0
46	URB	13.986257	BULK_OR_INTE...	OUT	81	\Device\libusb00001	0x867282...	STATUS_SUCCESS		0
47	URB	13.986300	BULK_OR_INTE...	IN	81	\Device\libusb00001	0x867282...	STATUS_PENDING		0
48	URB	13.986708	BULK_OR_INTE...	IN	81	\Device\libusb00001	0x867282...	STATUS_SUCCESS	BF 0D 25 70 A5 3C 73 ...	64
49	URB	13.986756	BULK_OR_INTE...	OUT	81	\Device\USBPDO-6	0x867282...	STATUS_SUCCESS		0
50	URB	13.986758	BULK_OR_INTE...	OUT	81	\Device\libusb00001	0x867282...	STATUS_SUCCESS		0
51	URB	13.986789	BULK_OR_INTE...	IN	81	\Device\libusb00001	0x867282...	STATUS_PENDING		0
52	URB	13.987207	BULK_OR_INTE...	IN	81	\Device\libusb00001	0x867282...	STATUS_SUCCESS	BF 0D 25 70 A5 3C 73 ...	64
53	URB	13.987226	BULK_OR_INTE...	OUT	81	\Device\USBPDO-6	0x867282...	STATUS_SUCCESS		0
54	URB	13.987227	BULK_OR_INTE...	OUT	81	\Device\libusb00001	0x867282...	STATUS_SUCCESS		0
55	URB	13.987254	BULK_OR_INTE...	IN	81	\Device\libusb00001	0x867282...	STATUS_PENDING		0
56	URB	13.987705	BULK_OR_INTE...	IN	81	\Device\libusb00001	0x867282...	STATUS_SUCCESS	BF 0D 25 70 A5 3C 73 ...	64
57	URB	13.987720	BULK_OR_INTE...	OUT	81	\Device\USBPDO-6	0x867282...	STATUS_SUCCESS		0
58	URB	13.987721	BULK_OR_INTE...	OUT	81	\Device\libusb00001	0x867282...	STATUS_SUCCESS		0
59	URB	13.987744	BULK_OR_INTE...	IN	81	\Device\libusb00001	0x867282...	STATUS_PENDING		0
60	URB	13.988082	BULK_OR_INTE...	IN	81	\Device\libusb00001	0x867282...	STATUS_SUCCESS	BF 0D 25 70 A5 3C 73 ...	64
61	URB	13.988100	BULK_OR_INTE...	OUT	81	\Device\USBPDO-6	0x867282...	STATUS_SUCCESS		0
62	URB	13.988101	BULK_OR_INTE...	OUT	81	\Device\libusb00001	0x867282...	STATUS_SUCCESS		0

Figura 6.51 Trama USB del envío de muestras al servidor, capturado con *USBTrace*

6.2.5 Evolución de los prototipos implementados

Es importante apreciar el grado de evolución que tuvo la plataforma *hardware* del sistema informático propuesto en este trabajo desde su concepción. La ausencia de modelos robustos de simulación que reúnan todas las instancias que implica un sistema de esta dimensión y sus múltiples interacciones, obliga el planteamiento de versiones intermedias que son depuradas y modificadas para que finalmente, encajen en un sistema más robusto que explota sus funciones. Con esta metodología, se obtienen módulos funcionales individuales que permiten la interconexión de otros sistemas, generalizando el propósito de los dispositivos que integran; por ejemplo, la Figura 6.52 muestra la interconexión de 6 módulos, de los cuales 3 corresponden a fuentes de alimentación con reguladores de conmutación. Estas fuentes ofrecen tensiones estables para sistemas digitales basados en niveles LVCMOS3V3, 2V5 y 1V2, con suministro de hasta 3A. Entre los otros 3 módulos están: una FPGA de la familia Spartan-6 de Xilinx con soporte para configuración en modos JTAG, *SelectMAP* y *Slave-Serial*, además de conectores de expansión para algunos de sus bancos E/S; un microcontrolador de 16-bit USB- *powered*, con sus puertos disponibles en conectores de expansión e interfaz para programación *In-system* y un módulo basado en CPLD *CoolRunnerII* de *Xilinx*, con disponibilidad de sus E/S e interfaz JTAG para su programación. Adicionalmente, fue implementado un banco de SRAM (no visible en la figura) con 4 chips R1LV0816ASB-5SI de 512kx16 cada uno.

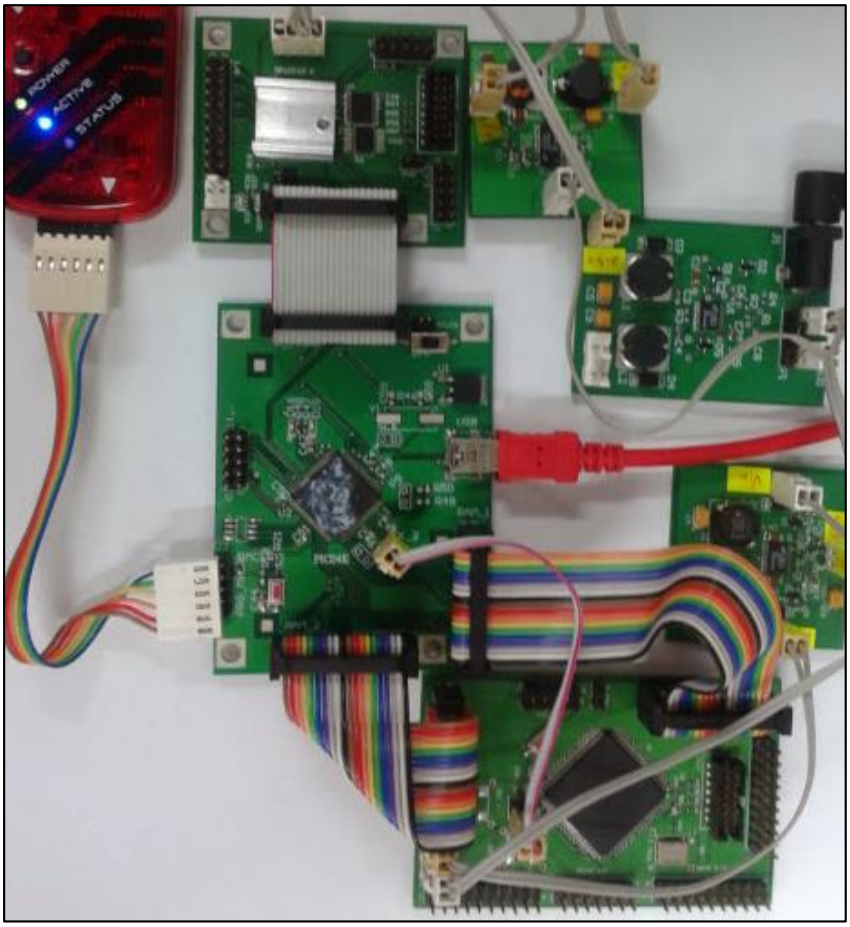


Figura 6.52 Interconexión de módulos funcionales de la plataforma hardware

Posterior a la implementación y verificación de los 6 módulos, con el primer proceso de integración la plataforma hardware evoluciona al sistema mostrado en la Figura 6.44, basado en dos módulos que se interconectan. Cada uno de estos módulos es configurable, pero sus propósitos se reducen debido a las conexiones mapeadas entre sus dispositivos.

En la Figura 6.53 se presenta la integración de la plataforma *hardware* en una única placa. Su propósito es específico para el sistema informático propuesto en este trabajo. No obstante, algunos dispositivos E/S han sido involucrados, además de un conector de expansión para la eventual conexión de *hardware* externo.

Finalmente, la Figura 6.54 presenta la implementación de una versión alternativa en dos módulos que pueden apilarse y que, en lugar de vincular un CPLD, vincula otro FPGA, que puede ser programado por JTAG o bien desde el microcontrolador. Esta versión del hardware brinda soporte especialmente para un incremento en los recursos de la plataforma al implementar configuración parcial del FPGA de usuario.

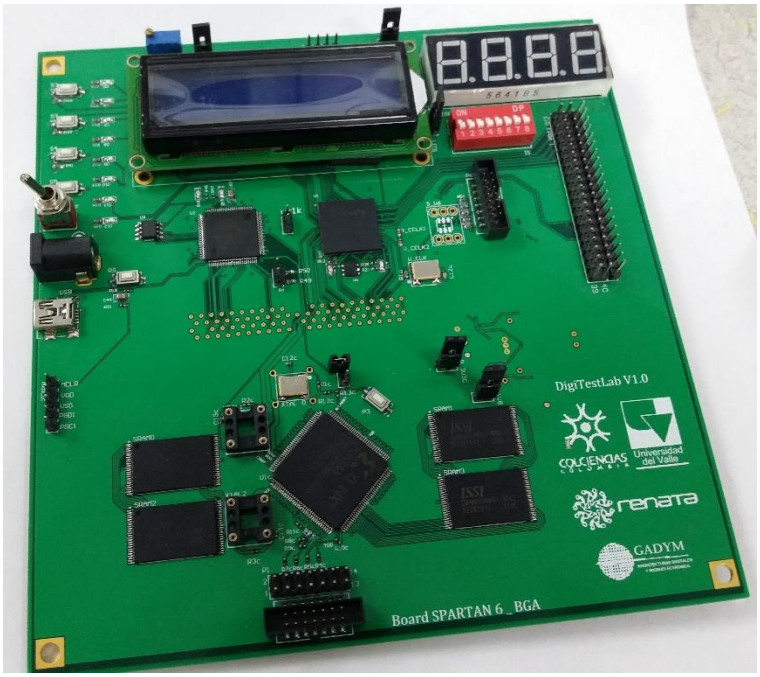


Figura 6.53 Plataforma hardware unificada

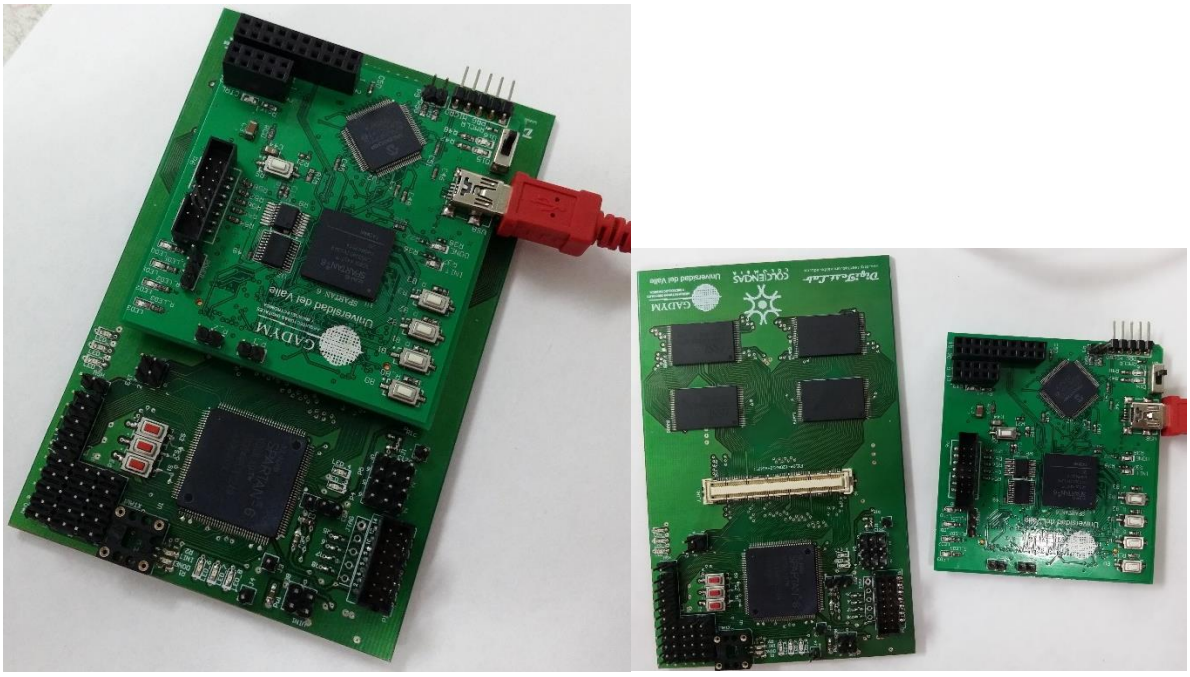


Figura 6.54 Plataforma hardware unificada en dos módulos

6.3 Resultados y Discusión de Implementación del Analizador Lógico

Cada uno de los módulos del analizador lógico, presentados en este capítulo, fueron descritos en *Verilog* e implementados mediante la herramienta ISE *Design Suite* sobre una FPGA Spartan-6 *xc6slx45t*, para facilitar su depuración bloque a bloque. Cabe destacar que esta FPGA se encuentra ensamblada en una de las tarjetas de desarrollo diseñadas para garantizar la depuración en *hardware* de estos módulos.

6.3.1 Síntesis de los módulos

Los esquemáticos obtenidos de la síntesis RTL para los principales bloques del subsistema de configuración se registran en la Figura 6.55, para la comunicación asíncrona, Figura 6.56 para la FSM de control, Figura 6.57 para el *Shell decoder*, Figura 6.58 para el banco de registros de configuración y Figura 6.59 para la cola de generación de estímulos. Estos gráficos pueden compararse con las descripciones arquitecturales presentadas en la sección 5.2. Los reportes de síntesis de sus implementaciones se muestran en la Tabla 6.5, en la que se puede apreciar que los bloques *FIFO Stimuli Generator* y *Shell decoder* ocupan más elementos lógicos (*BEL*) que otros en el subsistema de configuración. Algunos bloques lógicos para decodificación del banco de registros y soporte de las FSM no están incluidos en la tabla; sin embargo, estos bloques usan el 67.8% de los BEL, pero solamente el 38.7% de los *flip-flop*/*latches*. Estos porcentajes resultan de considerar el total usado por todo el subsistema y las unidades reportadas. Por otro lado, parametrizando la estructura FIFO con una profundidad de 256, ésta demandaría el 23.6% de los recursos del subsistema de configuración, lo que significaría casi una cuarta parte de su área.

El subsistema de configuración demanda una especial atención porque define todos los parámetros de operación del analizador lógico, en función de los propósitos del usuario. Así mismo, este subsistema es uno de los más complejos y el que más área demanda con respecto a los demás bloques.

Tabla 6.5 Recursos usados por el subsistema de configuración en el FPGA

Recursos	Comm. FSM	Control FSM	Shell_ decoder	Registers bank	FIFO Stimuli Generation	Total SubSys.
BELs	4	6	22	-	55	270
FlipFlops - Latches	3	3	-	56	57	194
Shift Registers	1	-	-	-	-	1
RAMS	-	-	-	-	4	16
Clock Buffers	1	1	-	1	2	2
I/O Buffers	6	9	32	79	22	110

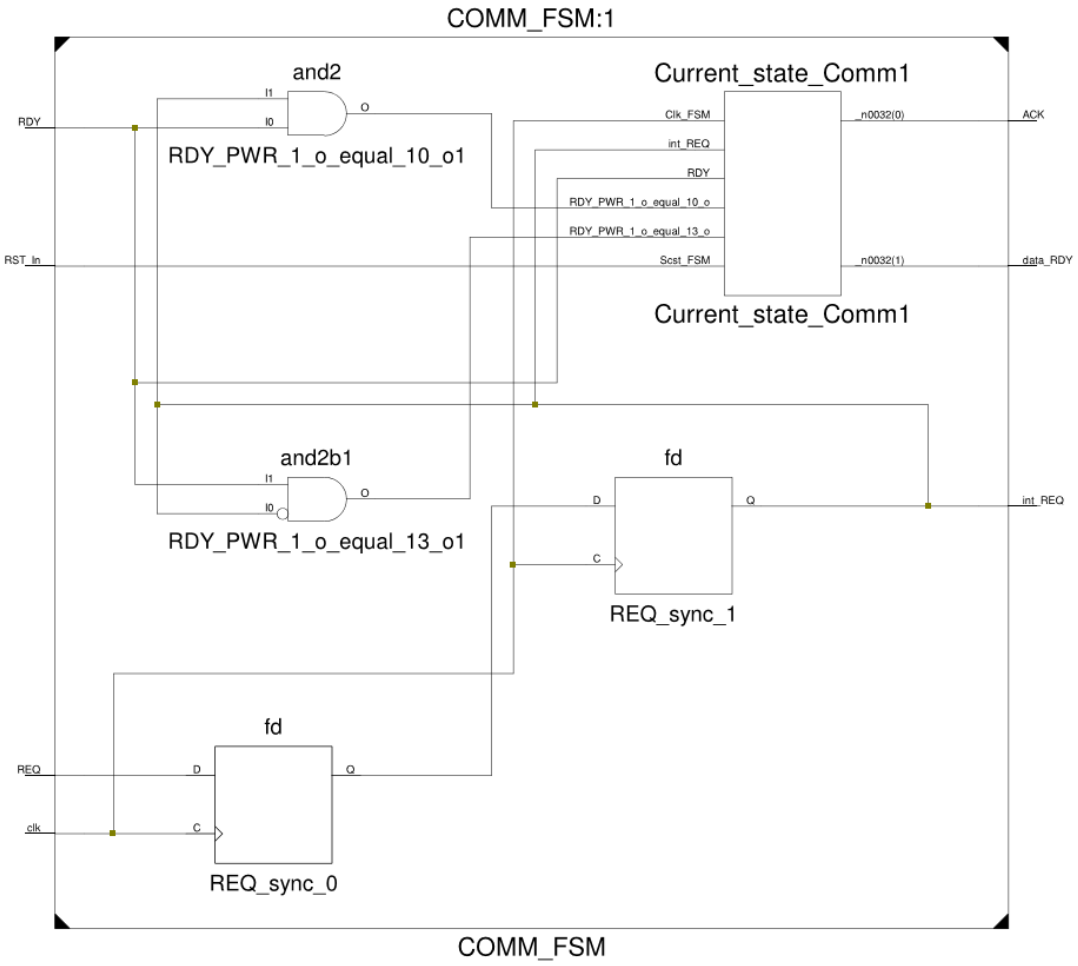


Figura 6.55 Síntesis RTL del bloque de comunicación asíncrona

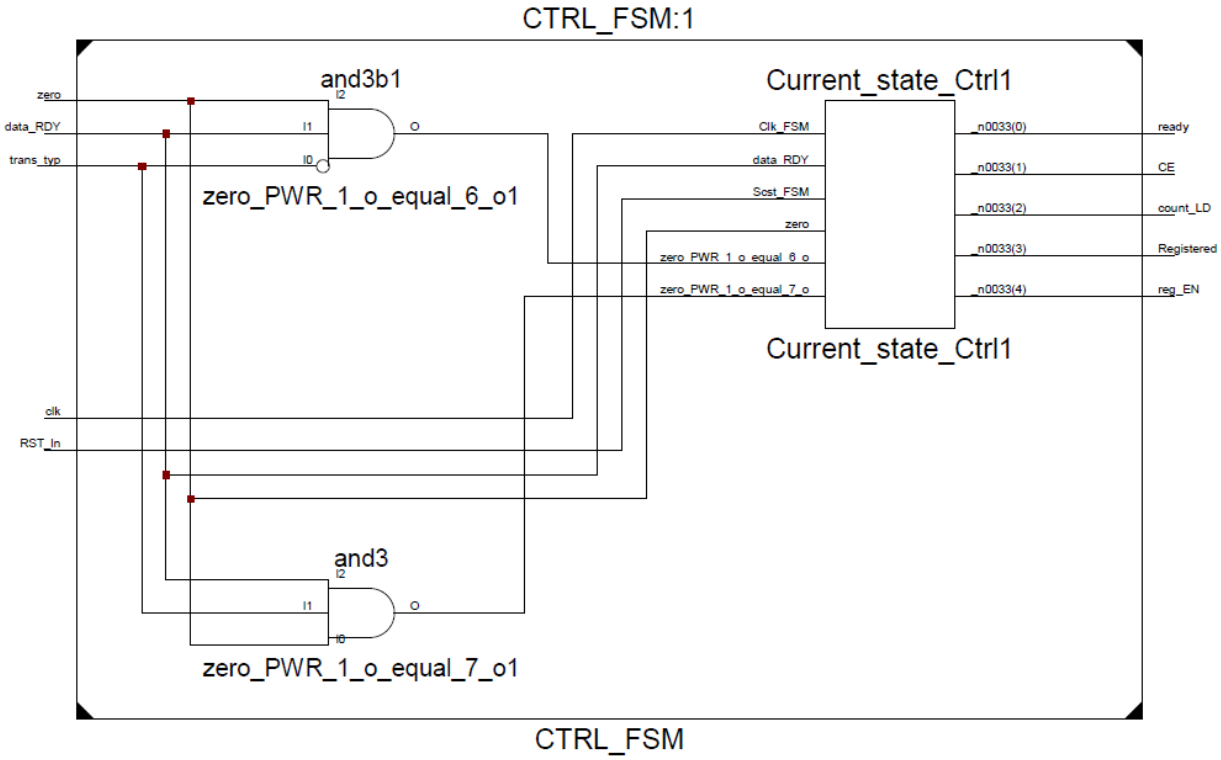


Figura 6.56 Síntesis RTL del FSM de control del subsistema de configuración

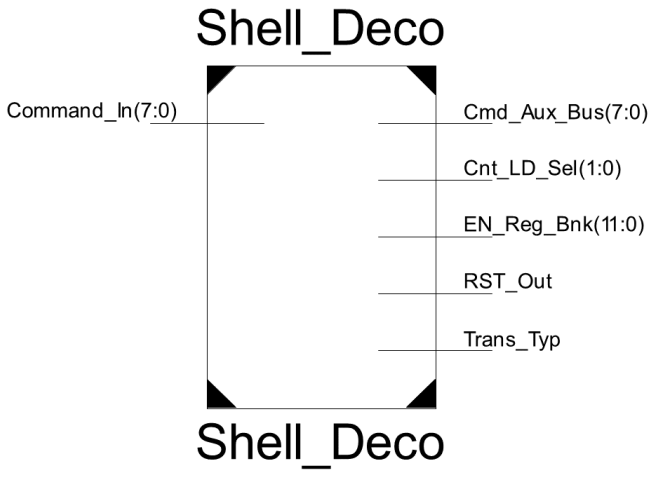


Figura 6.57 Módulo sintetizado del bloque de decodificación de comandos

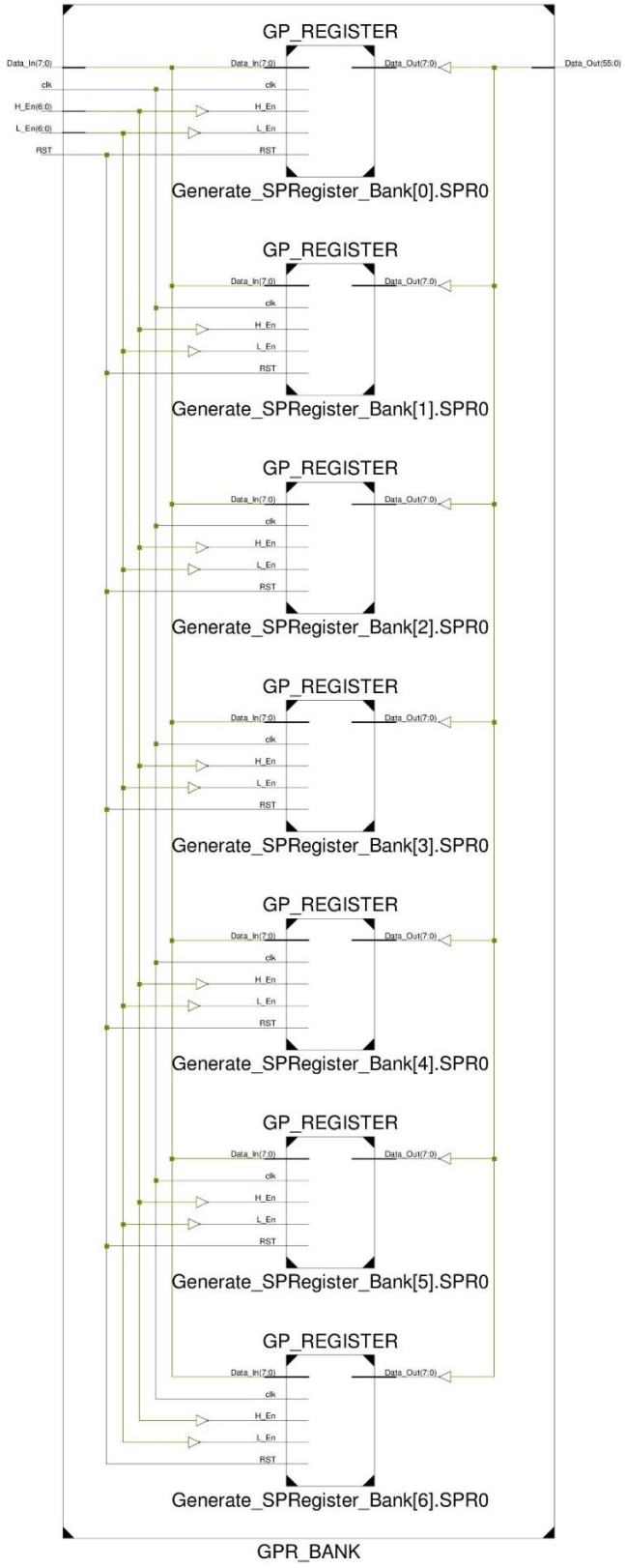


Figura 6.58 Síntesis RTL del bloque banco de registros de configuración

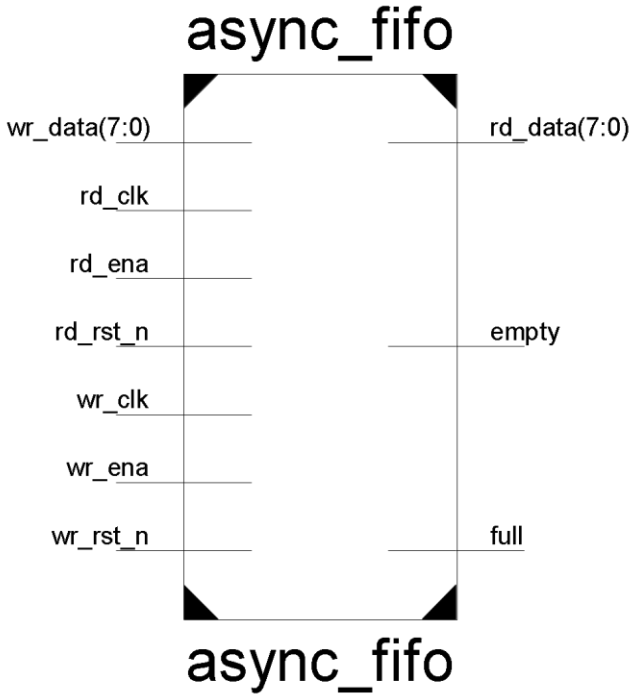


Figura 6.59 Módulo sintetizado del bloque de cola de generación de estímulos

Considerando los demás módulos del analizador lógico, el reporte de síntesis obtenido con la herramienta ISE para cada uno de ellos se resume en la Tabla 6.6, en la que se detallan los recursos demandados para su implementación en la FPGA. Los bloques de configuración *LA_COMM_CTRL*, de generación de direcciones *ADDRESS_GENERATOR* (Figura 6.61) y de muestreo de datos *DataSampl* (Figura 6.67) son los que más área demandan en la arquitectura del analizador lógico diseñada, usando respectivamente el 32.18%, 30.75% y 20.02% de los elementos lógicos, con respecto al total usado por el sistema completo *LA_MAIN*. En el mismo orden, estos bloques usan el 45.01%, 21.35% y 15.78% de los elementos de almacenamiento tipo *Flip-flop* o *Latch* reportados para el analizador. El porcentaje debido al subsistema de configuración es evidentemente alto por causa del banco de registros de configuración, la cola para almacenamiento de estímulos y las máquinas secuenciales que controlan la comunicación y lectura de comandos y parámetros.

Por otro lado, los subsistemas de direccionamiento *Addr_Decoder* (Figura 6.61), *DATA_BUS_MGM* (Figura 6.60) y detección de disparo por nivel *Lvl_Trigger_Det* (Figura 6.64), no involucran elementos de almacenamiento dada su naturaleza combinacional; mientras que, la máquina secuencial principal del analizador lógico usa apenas tres (3) *flip-flop* y un 0.95% de los elementos lógicos del sistema. Los bloques de sincronización de señales de control *Synchr_CtrlSign* (Figura 6.66) y de detección de disparo, *REQ2En_Pulse* y *TRIGG_EVENT_PULSE*, se constituyen como los de menor área con 0.72% y 0.36% de los elementos lógicos respectivamente. Cabe resaltar que estos dos últimos subsistemas de sincronización tienen la misma arquitectura de un detector de flanco de dos niveles (Figura 6.62), usando dos (2) *flip-flops*, dos (2) puertas *AND* y dos (2) inversores, lo que puede verificarse en el reporte de síntesis de la Tabla 6.6.

La síntesis del subsistema de detección de disparo por flanco y de la FSM principal del analizador lógico se observan en la Figura 6.63 y la Figura 6.65 respectivamente.

Tabla 6.6 Recursos demandados por los bloques del analizador lógico en su implementación sobre FPGA

Recursos	BELs	Slice LUTs	FlipFlops - Latches	LUT-Flip Flop pairs used	Shift Registers	RAMs	Clock Buffers	I/O Buffers
ADDRESS_GENERATOR	258	108	92	129			1	57
Addr_Decoder	14	14		14				19
DATA_BUS_MGM	32	32		32				66
Edge_Trigger_Det	10	9	16	18			1	24
Lvl_Trigger_Det	25	12		12				67
DataSampl	168	90	68	108			1	70
Synchr_CtrlSign	6	5	15	15			1	13
REQ2En_Pulse	3	2	2	4			1	4
TRIGG_EVENT_PULSE	3	2	2	4			1	4
LA_COMM_CTRL	270	232	194	293	1	16	2	110
MAIN_CTRL_FSM	8	8	3	11			1	11
LA_MAIN (Sistema completo)	839	561	431	681	1	16	3	127

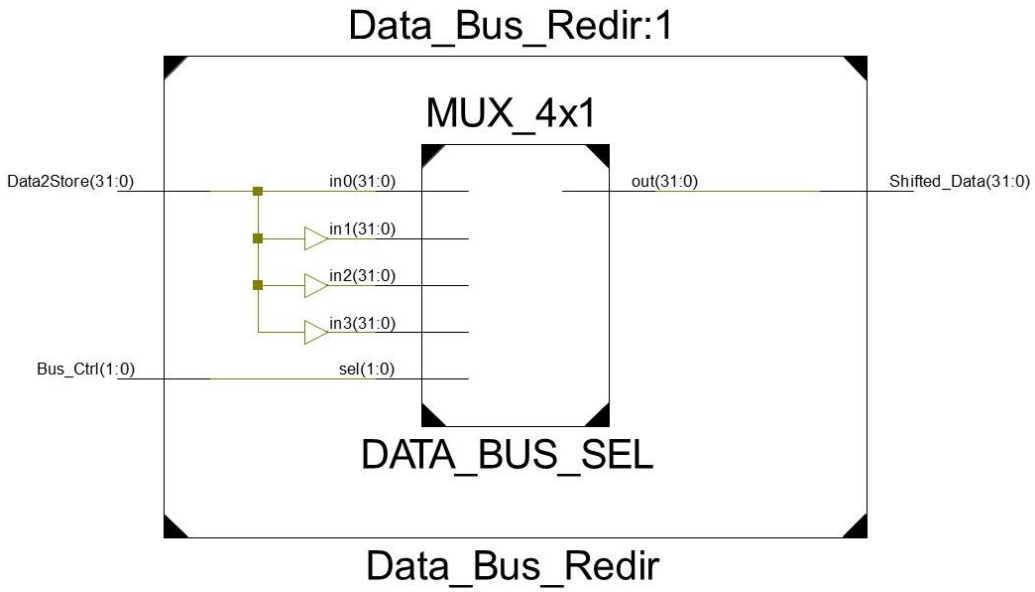


Figura 6.60 Módulo sintetizado del bloque de organización de datos *DATA_BUS_MGM*

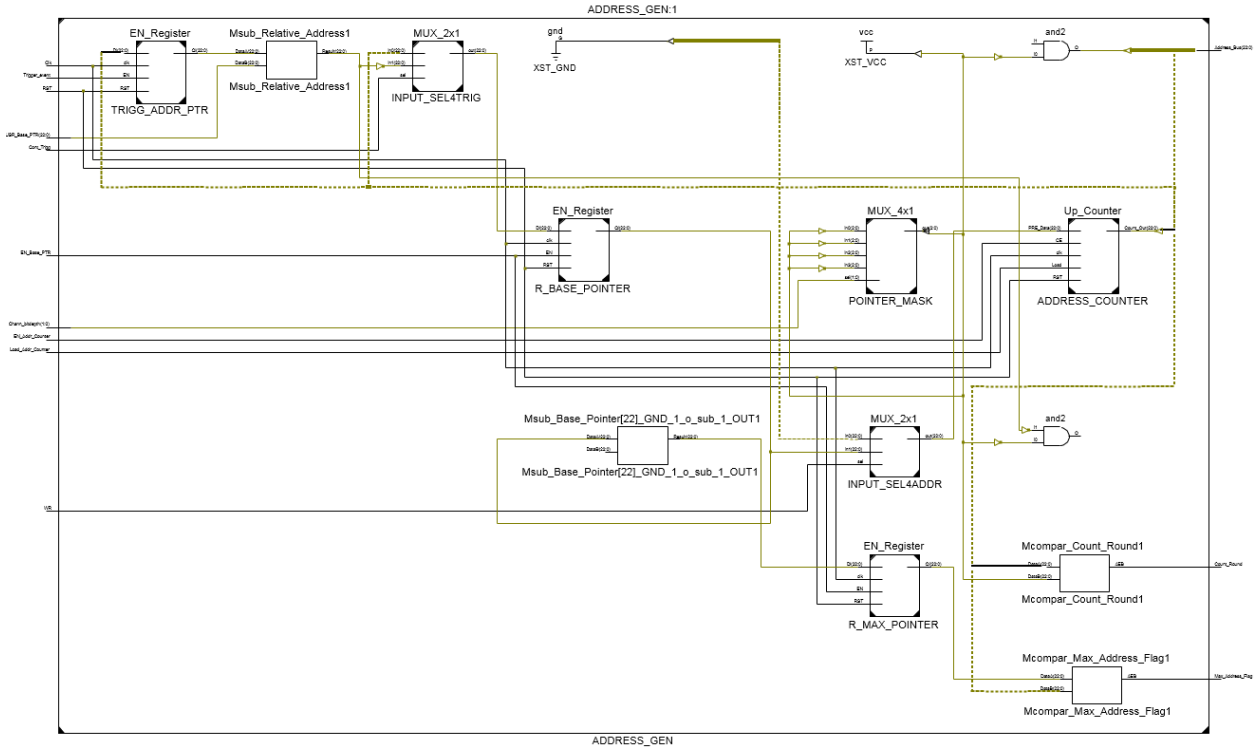


Figura 6.61 Módulo sintetizado del bloque de generación de direcciones *ADDRESS_GENERATOR*

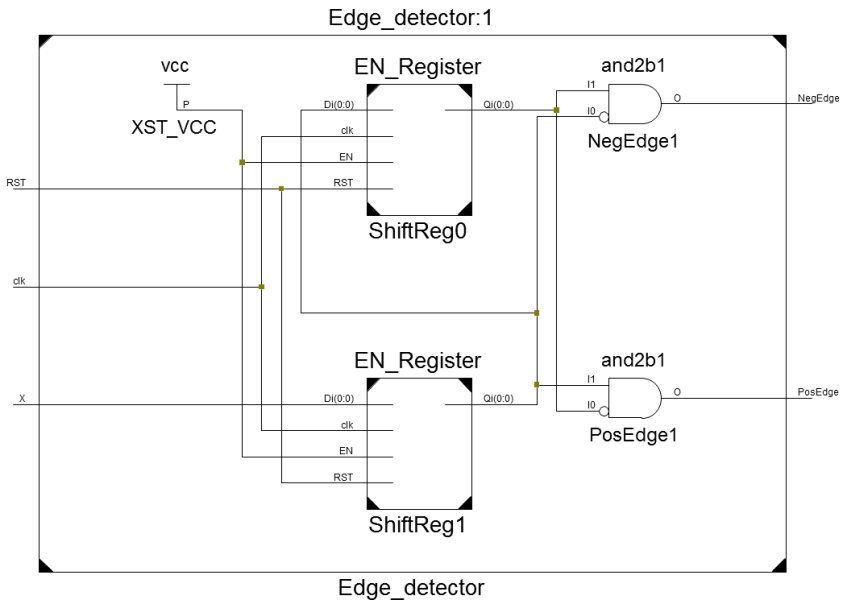


Figura 6.62 Módulo sintetizado de un detector de transiciones, para los subsistemas de sincronización *REQ2En_Pulse* y *TRIGG_EVENT_PULSE*

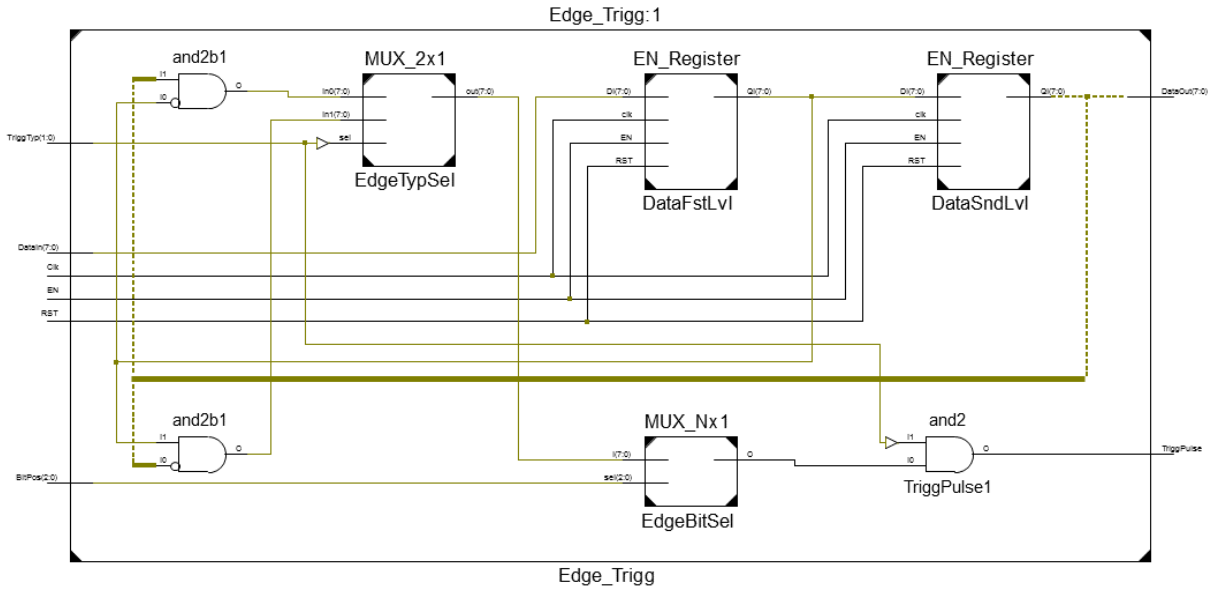


Figura 6.63 Módulo sintetizado del subsistema de detección de disparo por flanco *Edge_Trigger_Det*

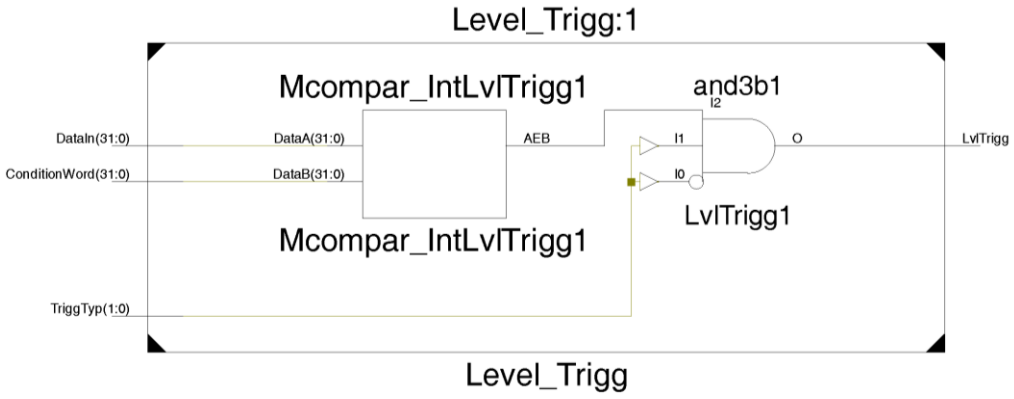


Figura 6.64 Módulo sintetizado del subsistema de detección de disparo por nivel *Lvl_Trigger_Det*

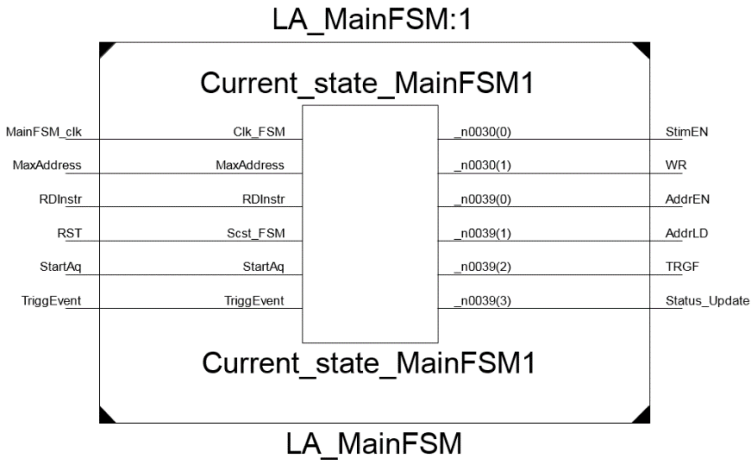


Figura 6.65 Módulo sintetizado de la máquina de estados principal del analizador lógico

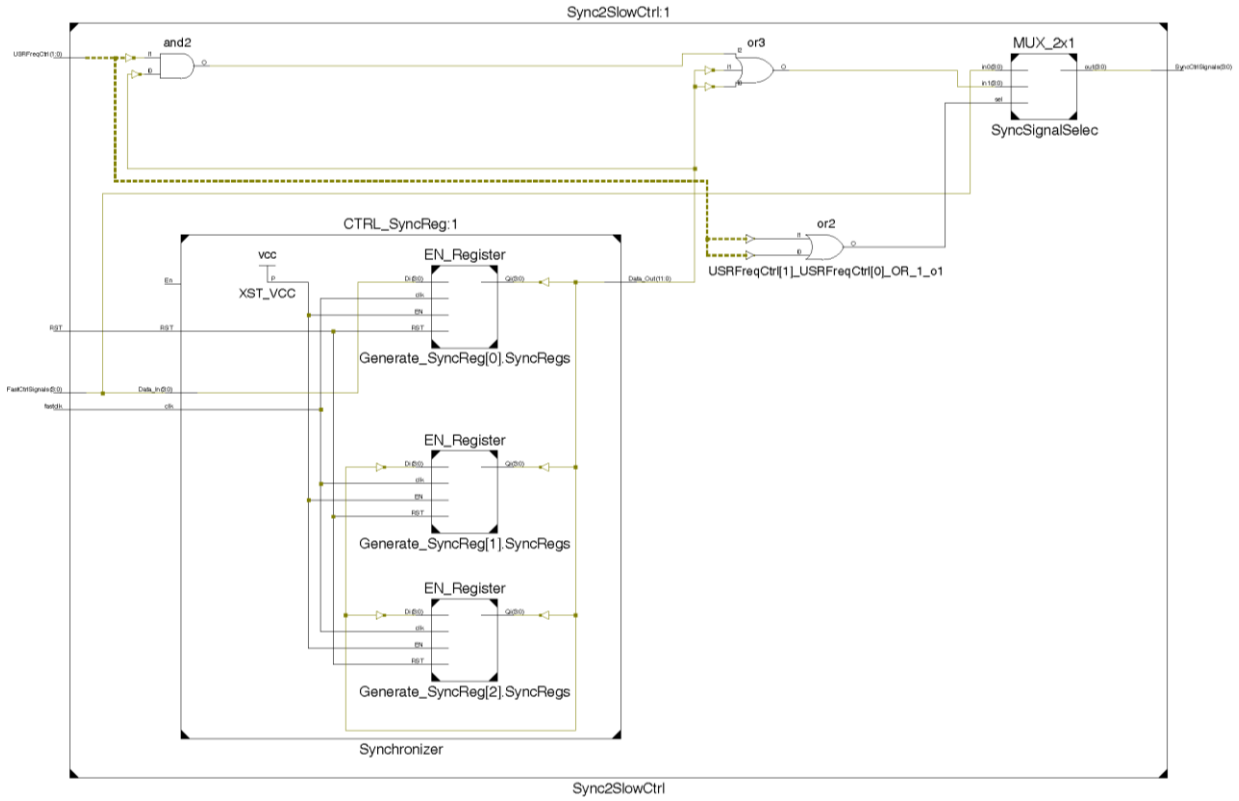


Figura 6.66 Módulo sintetizado del subsistema de sincronización de señales de control *Synchr_CtrlSign*

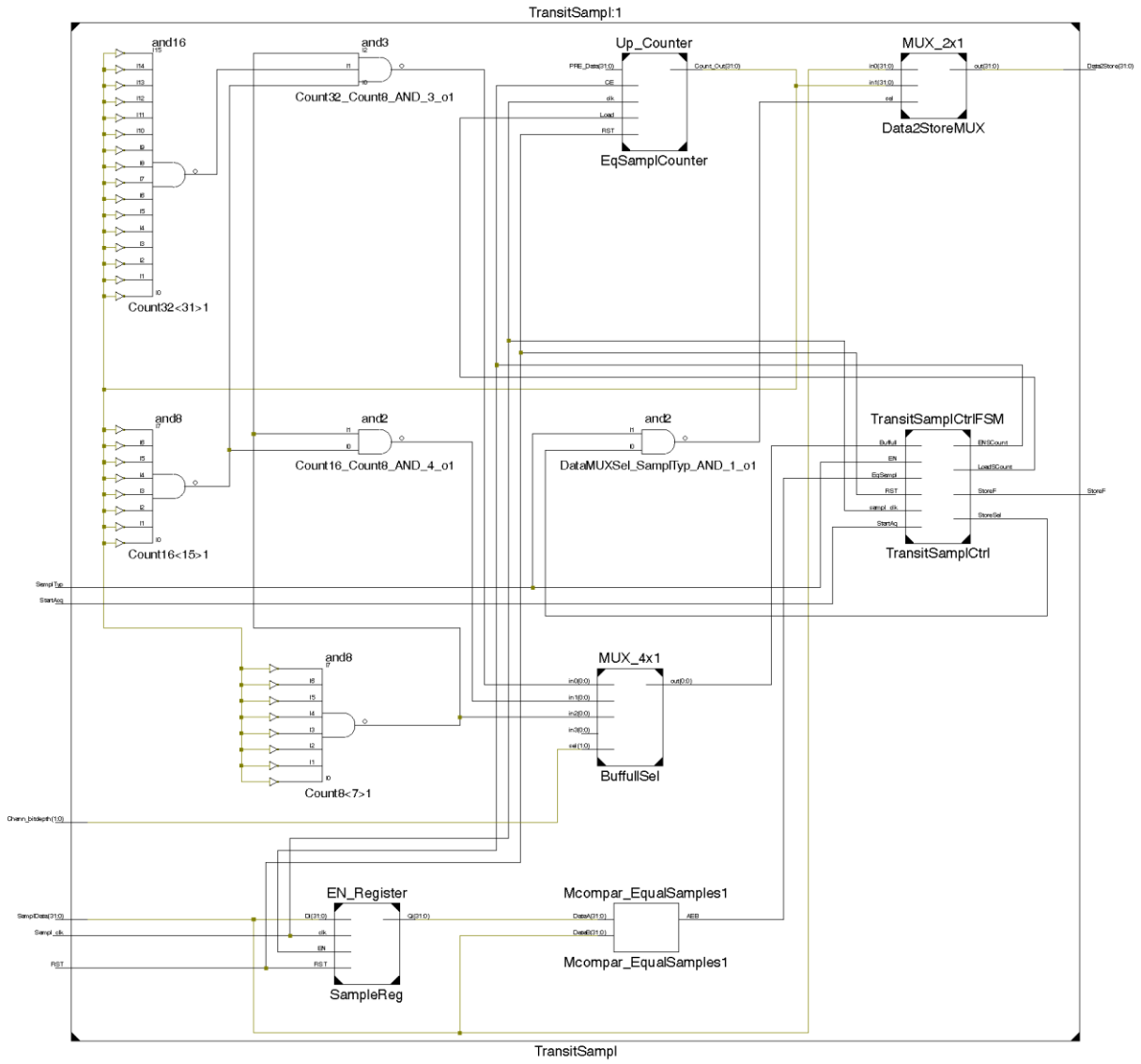


Figura 6.67 Módulo sintetizado del subsistema de muestreo de datos *DataSampl*

6.3.2 *Test* de comportamiento de los módulos

Para cada bloque se realizó una simulación de comportamiento a través *testbenches* escritos en *Verilog*. En cuanto al subsistema de configuración, la simulación de la FSM de comunicación asíncrona se muestra en la Figura 6.68, en la que se evidencia la transición de estados de acuerdo con el diagrama de la Figura 5.5. Esto puede verse analizando las señales del estado actual (*Current-state*) y próximo estado (*next_state*) cuando *REQ* y *RDY* son activadas o desactivadas. *REQ_sync* es la salida de un registro de desplazamiento que sincroniza la entrada *REQ*. La primera transacción en la Figura 6.68 muestra una ausencia de *ACK* debido a que *REQ* es demasiado corta y la comunicación de 4 cuatro fases (*4-phase*) no se ha completado. En este caso, se alcanza el estado *PROCESSING* en la máquina secuencial, pero retorna a *WAITING* sin activar *ACK*.

En la Figura 6.69 se muestra una transacción con comandos tipo 0 y tipo 1. Las transiciones de la FSM del subsistema de control pueden notarse inspeccionando *Current_state* y *next_state*, en las que "000" es *WAITING4COMM*, "001" es *SIMPLE_COMM*, "010" es *LOAD_CTRL*, "100" es *DATA_COUNTING* y "110" es *REGISTERING*. La mayoría de las transiciones de estado en estas transacciones cambian solamente un bit de sus códigos de estado, con el fin de disminuir su actividad de conmutación y minimizar el consumo de potencia debido a esto.

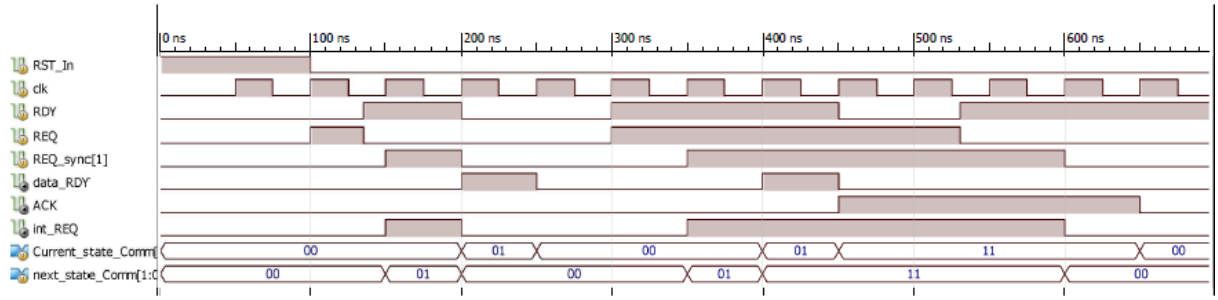


Figura 6.68 Simulación de comportamiento de la FSM de comunicación

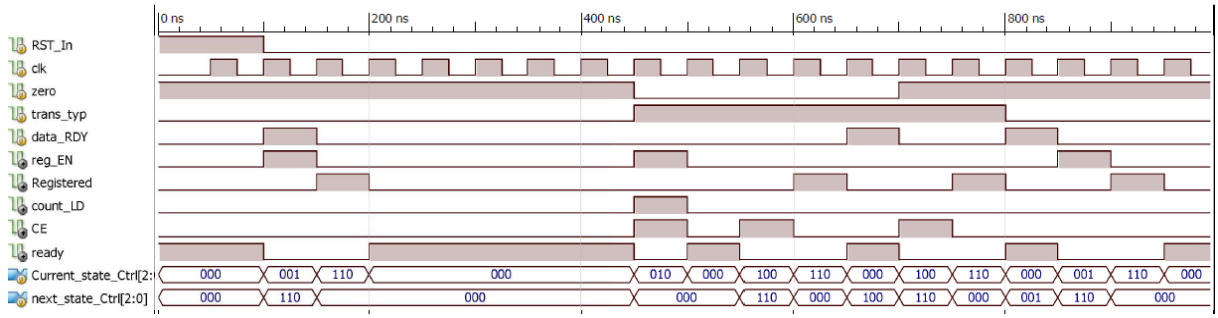


Figura 6.69 Simulación de comportamiento de la FSM del subsistema de control

El banco de registros de configuración fue simulado usando un barrido de control que habilita cada registro y carga desde sus entradas un conjunto de datos predefinidos. En la Figura 6.70, *H_En* habilita el *nibble* alto de uno de los registros y *L_En* lo hace con el *nibble* bajo. *Data_Out* muestra en un solo puerto de 56-bit el contenido de todo el banco de registros.

En la Figura 6.71 se muestra la simulación funcional del bloque *Shell decoder*, en la que *Trans_Typ* es el tipo de comando o transacción de acuerdo con lo recibido en *Command_In* (Ver Tabla 5.1); *Cnt_LD_Sel*

es asignada cuando se decodifica un comando tipo 1, permitiendo la precarga del contador de parámetros adicionales; *EN_Reg_Bnk* es un puerto de 12-bits que alimenta al registro *BNK_CTRL_REG* (Ver Tabla 5.2).

Los parámetros de temporización que aparecen en la Tabla 6.7 fueron obtenidos desde el proceso de síntesis con la herramienta ISE. La lógica de la estructura FIFO asíncrona introduce restricciones potenciales de temporización, considerando los otros bloques principales de este subsistema de configuración. Sin embargo, la lógica para decodificación del banco de registros y soporte de las FSM, que no fueron incluidas en la tabla, incrementan el máximo retardo debido al trayecto combinacional en todo el subsistema de configuración.

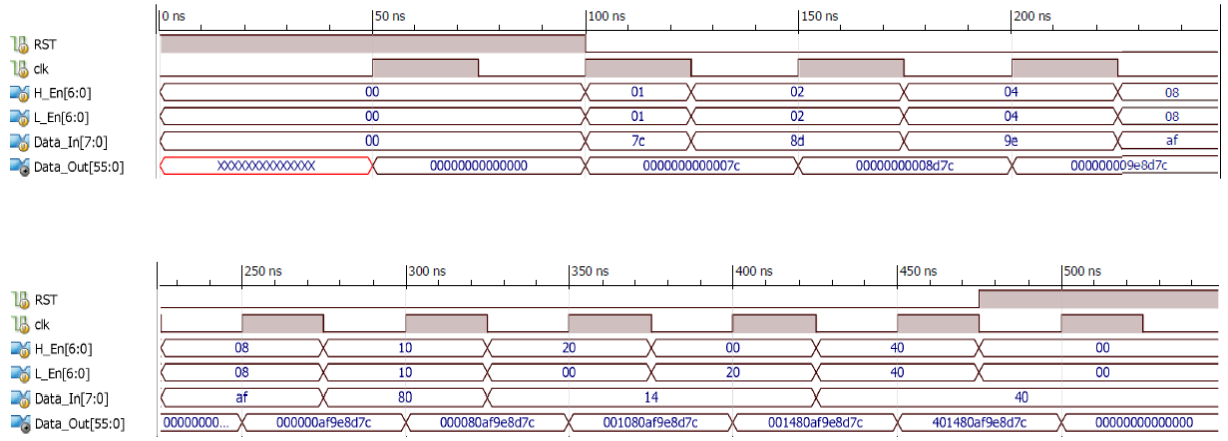


Figura 6.70 Simulación de comportamiento del banco de registros de configuración

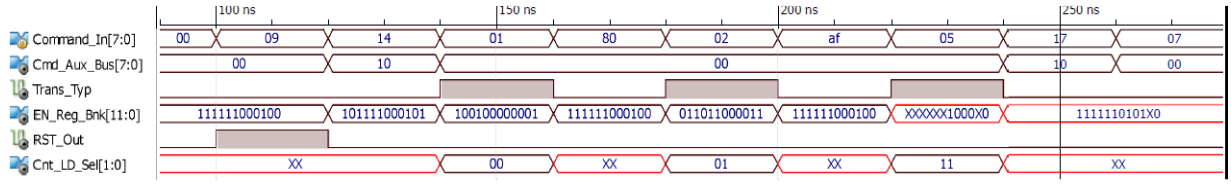


Figura 6.71 Simulación de comportamiento del bloque *Shell decoder*

Para la construcción del subsistema de configuración, todos los módulos fueron instanciados usando lenguaje *Verilog* y verificados funcionalmente usando un *testbench* que incluía cada comando y señales de *handshake* para la comunicación asíncrona. Algunos comandos modifican registros del banco de acuerdo con la Tabla 5.3, lo que puede observarse en simulación inspeccionando el puerto de 64-bits *RegisterBankData*, que contiene los siete registros de 8-bit en su parte menos significativa (56-bit).

La simulación de comportamiento de todo el subsistema de configuración y comunicación se puede observar en la Figura 6.72, en la que se consideran los comandos *RST*, *CHSAMP* y *CNF_TRG*. Cuando se reconoce el comando de *reset RST*, se genera una señal de restablecimiento para borrar los registros y la estructura FIFO de estímulos; así mismo, esta señal se considera como salida del subsistema de configuración para restablecer los otros módulos del analizador lógico. *CHSAMP* modifica el registro 0x05 (*AQ_FORMAT*) de acuerdo con la Figura 5.7. En la Figura 6.72, se considera el envío de este comando configurando una adquisición de 16 canales, lo que causa que el *nibble* más alto del registro seleccionado

se cargue con “0001”. En el mismo escenario, *CNF_TRG* envía 0x80 como parámetro adicional, para ser almacenado en el registro 0x04 (*CONF_TRIG*), lo que configuraría un muestreo transicional y disparo o adquisición continuo.

La simulación de comportamiento del comando *TRGW* está en la Figura 6.73, en la que se envía como palabra de condición de disparo 0xAF9C8D7C, que se almacena en los registros 0x03~0x00.

El envío de estímulos se realiza con *STM2SND* y su simulación de comportamiento considera 256 parámetros adicionales que se envían posteriormente al comando hasta llenar la estructura FIFO de estímulos (Ver Figura 6.74). Cuando se activa *STIM_RD_EN*, se leen los estímulos desde la cola (por el puerto *STIM_DATA*). Mediante las banderas *STIM_FIFO_FULL* y *STIM_FIFO_EMPTY* se comunica si la cola de estímulos está totalmente llena o vacía (Ver Figura 6.75 y Figura 6.76).

Tabla 6.7 Resumen de parámetros de temporización estimados para cada bloque

Parámetro	Comm. FSM	Control FSM	Shell_ decoder	Registers bank	FIFO Stimuli Generation	Total SubSys.
Minimum period (ns)	1.828	1.866	-	-	5.507	8.843
Maximum frequency (MHz)	547.046	535.906	-	-	181.587	113.084
Min. global offset in before (ns)	2.658	3.724	-	3.654	6.327	8.221
Max. output required time before clock (ns)	4.202	5.457	-	4.118	5.662	8.627
Max. combinational path delay (ns)	-	-	6.939	-	-	9.023

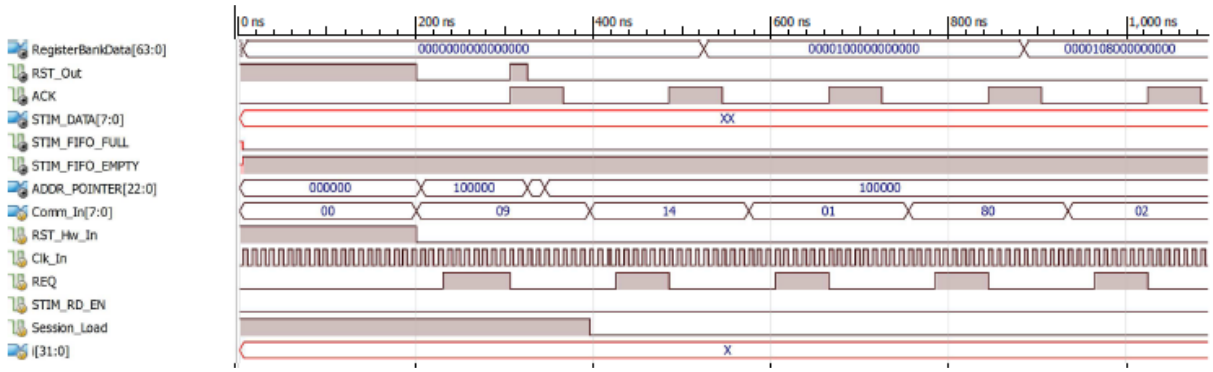


Figura 6.72 Simulación de comportamiento del subsistema de configuración – comandos: *RST*, *CHSAMP* y *CNF_TRG*

El puntero a memoria de muestreo (*ADDR_POINTER*) puede incrementarse o decrementarse en una posición debido a la ejecución de *PNTADJ*. En la Figura 6.77, este comando se envía como 0x17 para incremento y 0x07 para decremento. De acuerdo con la Tabla 5.4, el puntero a memoria de muestreo inicia conforme a la configuración dada por *CHSAMP*, por lo que es conveniente ejecutar este comando, antes de hacer el ajuste con *PNTADJ*.

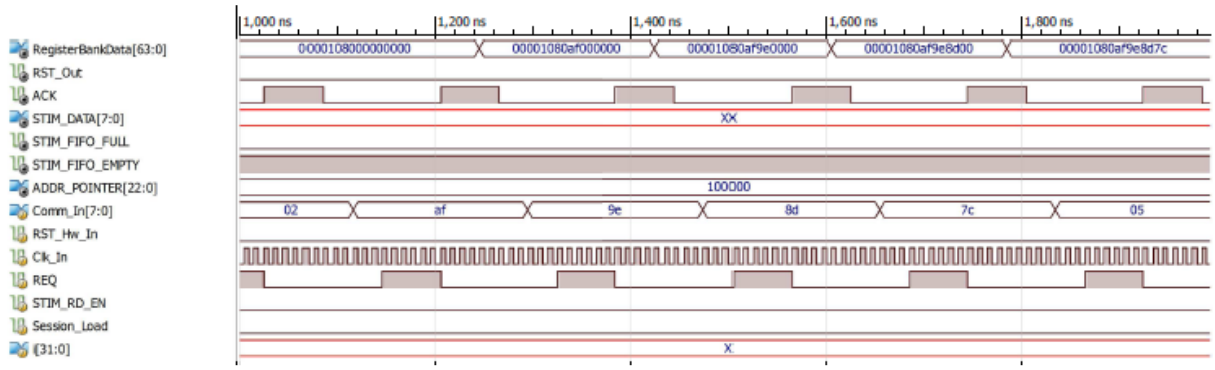


Figura 6.73 Simulación de comportamiento del subsistema de configuración – comando TRGW y parámetros adicionales

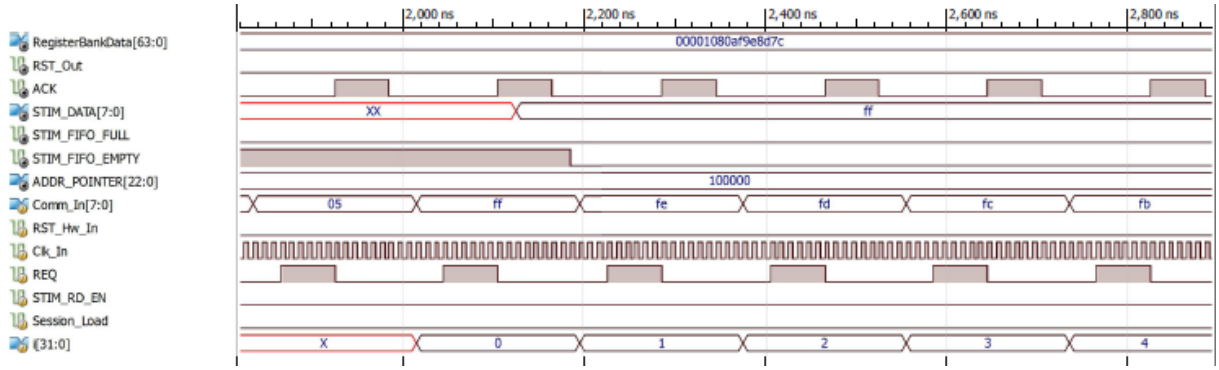


Figura 6.74 Simulación de comportamiento del subsistema de configuración – comando STM2SND y primeros parámetros adicionales

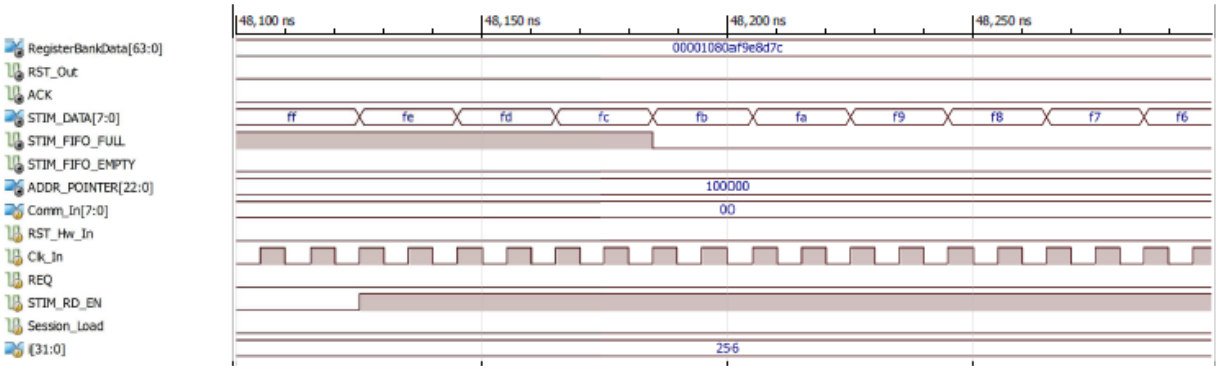


Figura 6.75 Simulación de comportamiento del subsistema de configuración – lectura de estímulos

Por otro lado, el subsistema de direccionamiento se destaca también por la forma particular en que se establecen los punteros para facilitar la lectura de datos de memoria en función de los parámetros de disparo

especificados por el usuario. Como se describió en la sección 5.3, los punteros *TRG_ADDR_PTR*, *R_BASE_PTR* y *R_MAX_PTR* se actualizan fundamentalmente durante la fase de escritura de muestras en memoria, cuando ocurre un evento de disparo, como puede apreciarse en la simulación de la Figura 6.78. El almacenamiento de las direcciones base y máxima sirve al bus de direcciones para apuntar a la dirección base cuando se inicia un proceso de lectura (ver Figura 6.79) y para que los subsistemas conectados reciban la alerta de alcance de la máxima dirección de memoria.

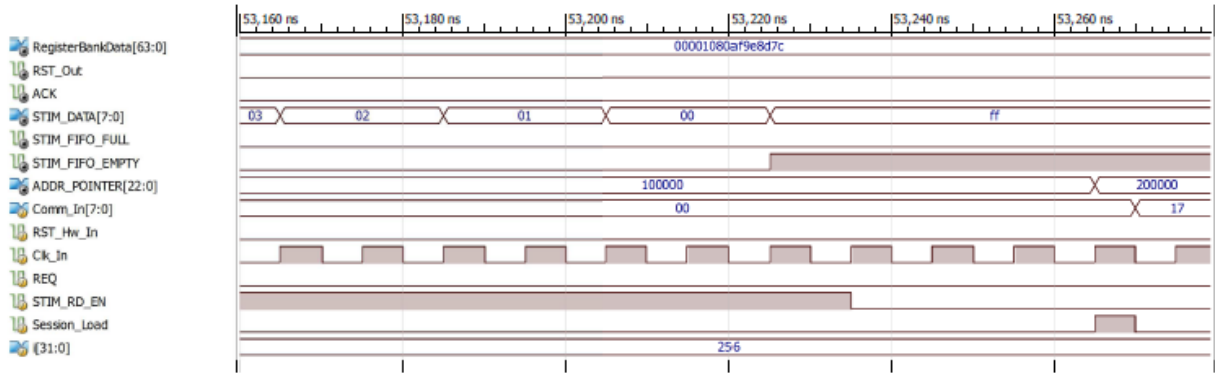


Figura 6.76 Simulación de comportamiento del subsistema de configuración – Finalizando lectura de estímulos

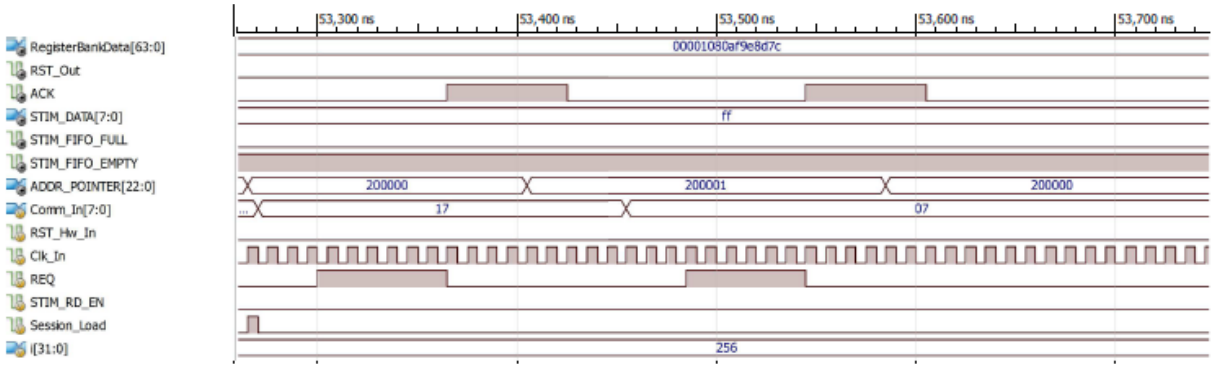


Figura 6.77 Simulación de comportamiento del subsistema de configuración – comando PNTADJ para ajuste de puntero hacia arriba y hacia abajo

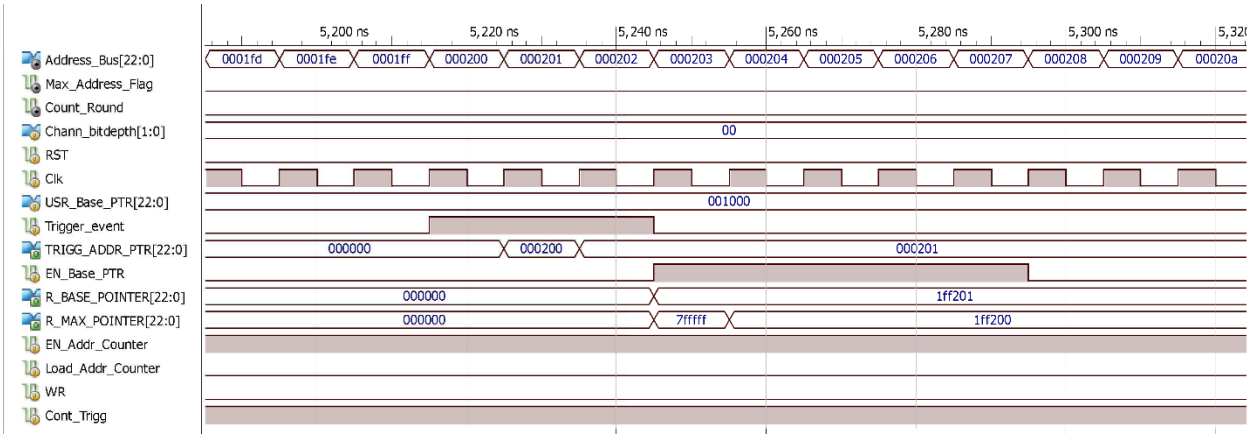


Figura 6.78 Simulación de comportamiento del subsistema de generación de direcciones – Carga de punteros

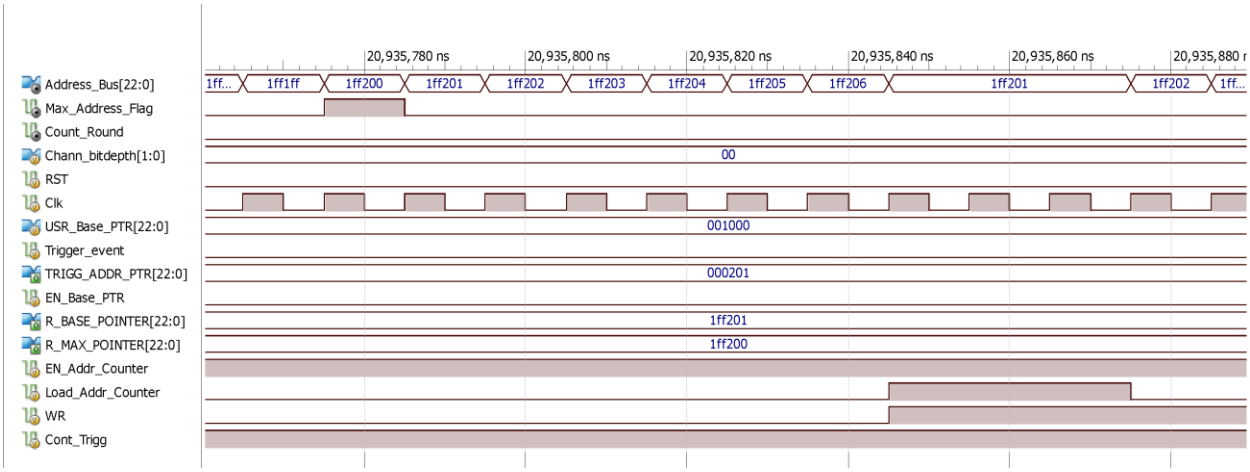


Figura 6.79 Simulación de comportamiento del subsistema de generación de direcciones- Establecimiento del bus de direcciones en la dirección base

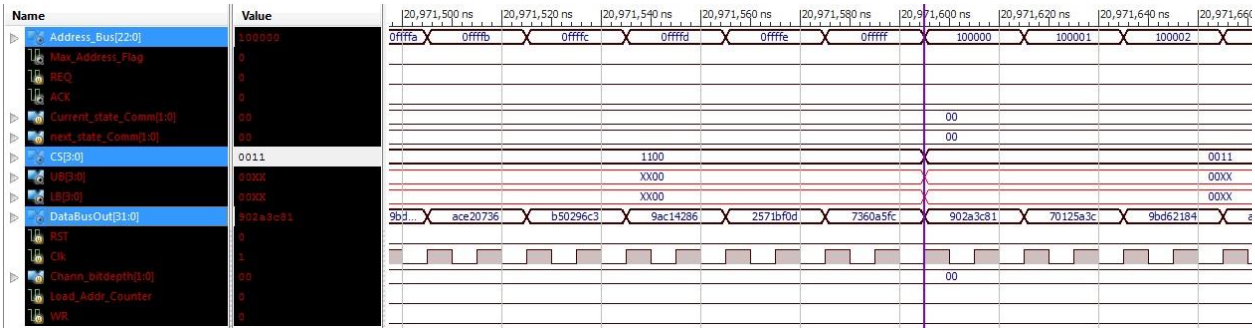


Figura 6.80 Simulación de comportamiento del decodificador de direcciones- Ciclo escritura, 32 canales

El decodificador de direcciones, descrito en la sección 5.3.1, hace posible la generación de las señales de control de habilitación para cada uno de los bancos de memoria. Conforme a los rangos de direccionamiento

dados en la Tabla 5.4 y la secuencia de activación de los bancos especificada en la Tabla 5.5, se definen las siguientes direcciones para cambio de banco o unidad física de memoria:

- 32 canales: 0x000000 y 0x100000
- 16 canales: 0x000000, 0x100000, 0x200000 y 0x300000
- 8 c*a*nales: 0x000000, 0x200000, 0x400000 y 0x600000

En estas direcciones, para cada caso debe haber una transición en el bus CS, que define la activación de chip de las unidades de memoria. Sin embargo, en el caso particular de selección de 8 canales, dentro de cada rango de direcciones en el que no cambia CS, se presenta una transición en los buses UB y LB (ver Tabla 5.5), además de la que se presenta durante los cambios de CS. En la Figura 6.80, se ilustra la transición de CS para la dirección correspondiente a una selección de 32 canales durante un proceso de escritura; la Figura 6.81 lo hace para una de las direcciones que corresponde a una selección de 16 canales y la Figura 6.82 presenta una transición de activación del *byte* bajo al *byte* alto mientras está seleccionado el chip N° 2 de la memoria de muestreo.

Los datos que aparecen en los puertos *O* y *DataBusOut* de las figuras mencionadas ilustran respectivamente la entrada y la salida del módulo de administración del bus de datos, que organiza los datos en el bus de salida para almacenamiento en memoria, conforme está descrito en la Tabla 5.5.

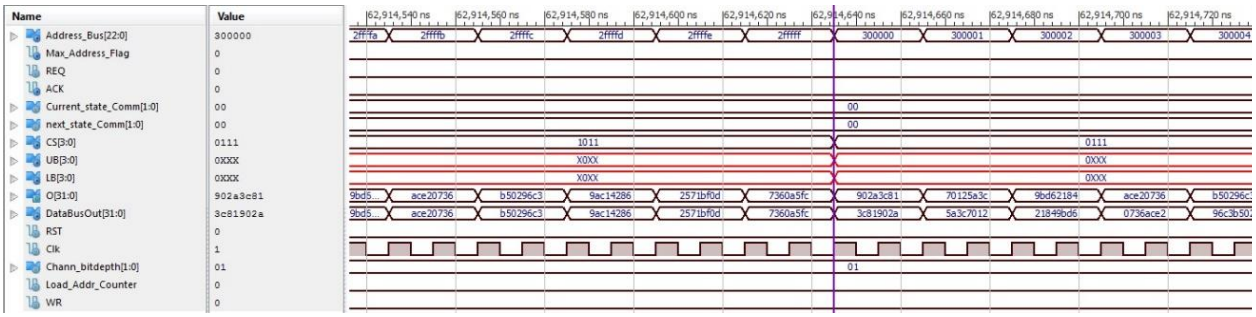


Figura 6.81 Simulación de comportamiento del decodificador de direcciones- Ciclo escritura, 16 canales

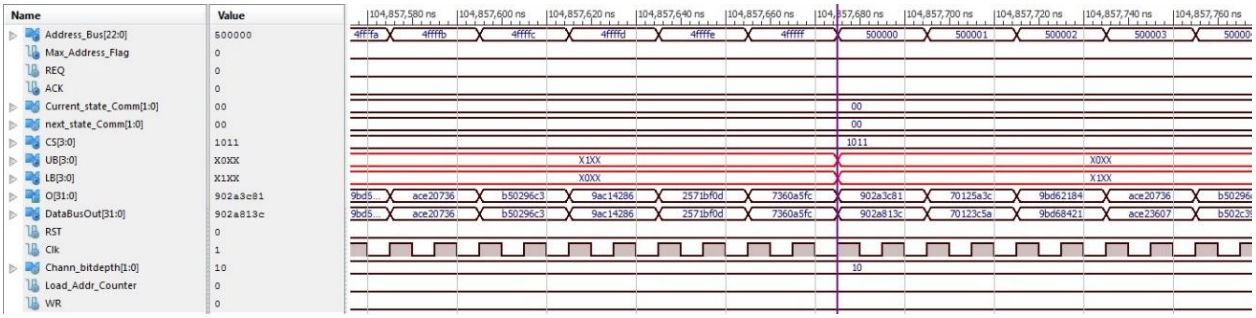


Figura 6.82 Simulación de comportamiento del decodificador de direcciones- Ciclo escritura, 8 canales

Para esta prueba, los datos se alimentan desde un multiplexor 8-1 de 32-bit, cuyo control se hace con un contador binario de 3-bits. Las entradas del MUX se fijan como aparece en la Tabla 6.8 y el módulo de administración del bus *DATA_BUS_MGM*, según su funcionamiento descrito en 5.3.1, puede entregar al bus de datos para almacenamiento la secuencia de la Tabla 6.8, la Tabla 6.9, la Tabla 6.10 o la Tabla 6.11, conforme se especifica en la Tabla 6.12. En esta última, se define esa salida del módulo en función del vector de datos aplicado, el número de canales de adquisición y el rango de direcciones que permite la selección de cada banco de memoria en particular.

Tabla 6.8 Vectores de datos de prueba - Secuencia 0

	HW				LW			
IN0	9	0	2	A	3	C	8	1
IN1	7	0	1	2	5	A	3	C
IN2	9	B	D	6	2	1	8	4
IN3	A	C	E	2	0	7	3	6
IN4	B	5	0	2	9	6	C	3
IN5	9	A	C	1	4	2	8	6
IN6	2	5	7	1	B	F	0	D
IN7	7	3	6	0	A	5	F	C

Tabla 6.9 Vectores de datos de prueba - Secuencia 1

HW				LW			
3	C	8	1	9	0	2	A
5	A	3	C	7	0	1	2
2	1	8	4	9	B	D	6
0	7	3	6	A	C	E	2
9	6	C	3	B	5	0	2
4	2	8	6	9	A	C	1
B	F	0	D	2	5	7	1
A	5	F	C	7	3	6	0

Tabla 6.10 Vectores de datos de prueba - Secuencia 2

HW				LW			
9	0	2	A	8	1	3	C
7	0	1	2	3	C	5	A
9	B	D	6	8	4	2	1
A	C	E	2	3	6	0	7
B	5	0	2	C	3	9	6
9	A	C	1	8	6	4	2
2	5	7	1	0	D	B	F
7	3	6	0	F	C	A	5

Tabla 6.11 Vectores de datos de prueba - Secuencia 3

HW				LW			
8	1	3	C	9	0	2	A
3	C	5	A	7	0	1	2
8	4	2	1	9	B	D	6
3	6	0	7	A	C	E	2
C	3	9	6	B	5	0	2
8	6	4	2	9	A	C	1
0	D	B	F	2	5	7	1
F	C	A	5	7	3	6	0

Tabla 6.12Vectores de salida en función de entradas, direcciones y tamaño de palabra de adquisición

Palabra Adquisición	**Rango direcciones**	**Secuencia de salida en bus**
32 canales	0x000000~0x0FFFFF	Secuencia 0
	0x100000~0x1FFFFF	Secuencia 0
16 canales	0x000000~0x0FFFFF	Secuencia 0

	0x100000~0x1FFFFF	Secuencia 1
	0x200000~0x2FFFFF	Secuencia 0
	0x300000~0x3FFFFF	Secuencia 1
8 canales	0x000000~0x0FFFFF	Secuencia 0
	0x100000~0x1FFFFF	Secuencia 2
	0x200000~0x2FFFFF	Secuencia 1
	0x300000~0x3FFFFF	Secuencia 3
	0x400000~0x4FFFFF	Secuencia 0
	0x500000~0x5FFFFF	Secuencia 2
	0x600000~0x6FFFFF	Secuencia 1
	0x700000~0x7FFFFF	Secuencia 3

Con respecto a otro de los módulos esenciales del analizador lógico, el subsistema de muestreo de datos, descrito en la sección 5.5, es transparente a los datos que se registran a su entrada cuando el usuario selecciona un muestreo normal, como se evidencia en la simulación de la Figura 6.70. En este caso, se asumió una selección de adquisición de 32 canales; con *SamplTyp*=0 se define un muestreo normal y la señal de advertencia *StoreF* no tiene efecto alguno, por lo que permanece sin cambiar. Obsérvese la relación entre la entrada de datos al módulo *SamplData* y los datos emitidos para almacenamiento *Data2Store*.

En el *testbench* también se considera la selección de un muestreo transicional, cuya simulación muestra, en la Figura 6.84, una adquisición de 32 canales, y en la Figura 6.85 una adquisición de 8 canales. En ambos casos, se presenta la persistencia de un dato en la entrada del subsistema durante más de un ciclo del reloj de muestreo para reflejar el correcto comportamiento del subsistema. Si se inspeccionan el puerto de salida de datos para almacenamiento, *Data2Store*, y la bandera *StoreF*, se puede observar el almacenamiento selectivo de cada dato de entrada y la cantidad de ciclos (aumentado en uno) durante los que persiste en el canal. Así mismo, la Figura 6.85 presenta un caso especial en el que se desborda el contador de muestras con respecto al tamaño de palabra definido por el número de canales seleccionado; en tal caso, se debe emitir para almacenamiento el valor máximo correspondiente y a continuación, el dato de entrada nuevamente, como si se tratara de un dato diferente.

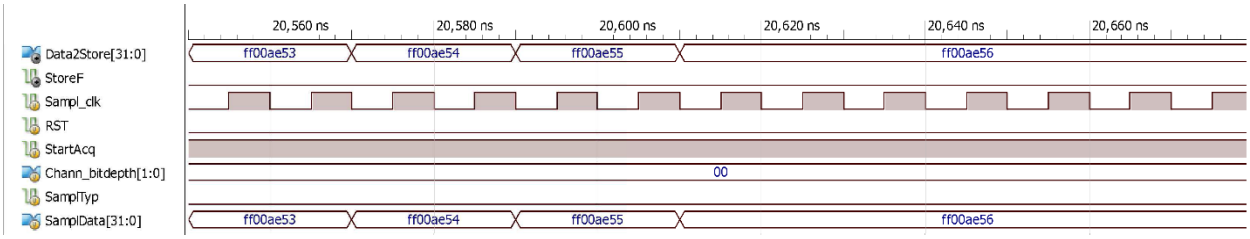


Figura 6.83 Simulación de comportamiento del subsistema de muestreo de datos– Muestreo Normal, 32 canales

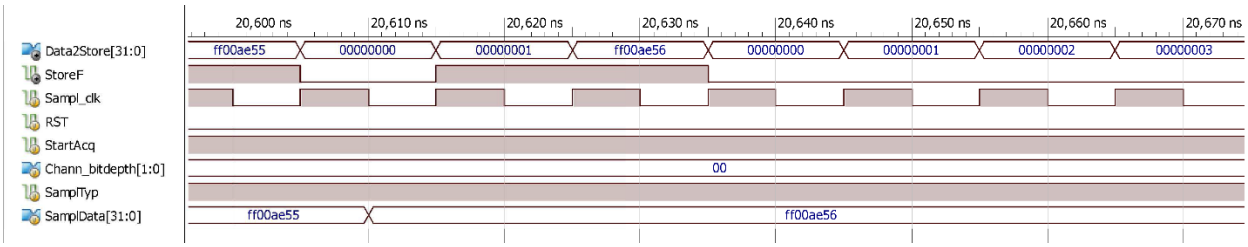


Figura 6.84 Simulación de comportamiento del subsistema de muestreo de datos– Muestreo Transicional, 32 canales

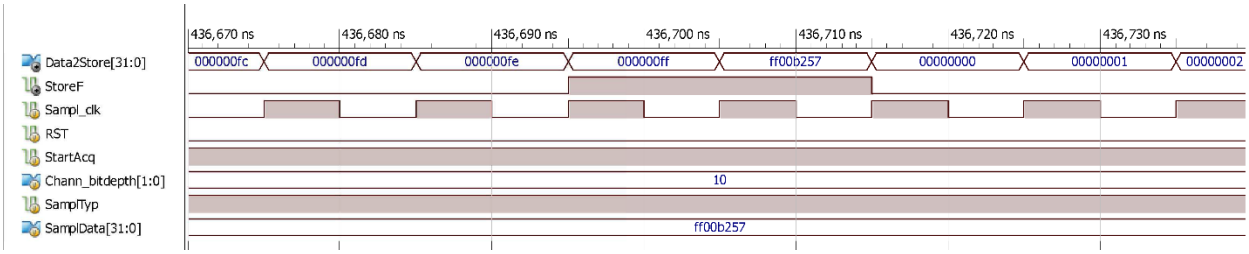


Figura 6.85 Simulación de comportamiento del subsistema de muestreo – Muestreo Transicional, 8 canales

Finalmente, para una verificación del sistema integral, se ajustan los comandos de configuración y operación del analizador lógico dentro de un *testbench*, que permite realizar a través de la interfaz del simulador ISIM de la herramienta EDA ISE, el seguimiento temporal a los registros y señales más relevantes de la arquitectura planteada. Para el caso particular que se muestra en la Figura 6.86, el analizador recibe y ejecuta los comandos *RST*, *CTRG*, *TRGW* y *SFREQ*, configurando una análisis disparado por nivel, con palabra de disparo 0xAF9E8D7C, muestreo transicional, frecuencia de muestreo a la mitad de la frecuencia del sistema, frecuencia de paso para aplicación de estímulos del mismo valor y 32 canales de adquisición. Es importante observar en esta simulación los valores con que se va actualizando el banco de registros, visto a través del puerto *RegisterBankData*, que corresponden con la configuración planteada por comandos, de acuerdo a los formatos definidos en la sección 5.2. Así mismo, en la ilustración pueden apreciarse la generación del *reset* y la inactividad de los buses de direcciones, datos y salida de estímulos durante este proceso.

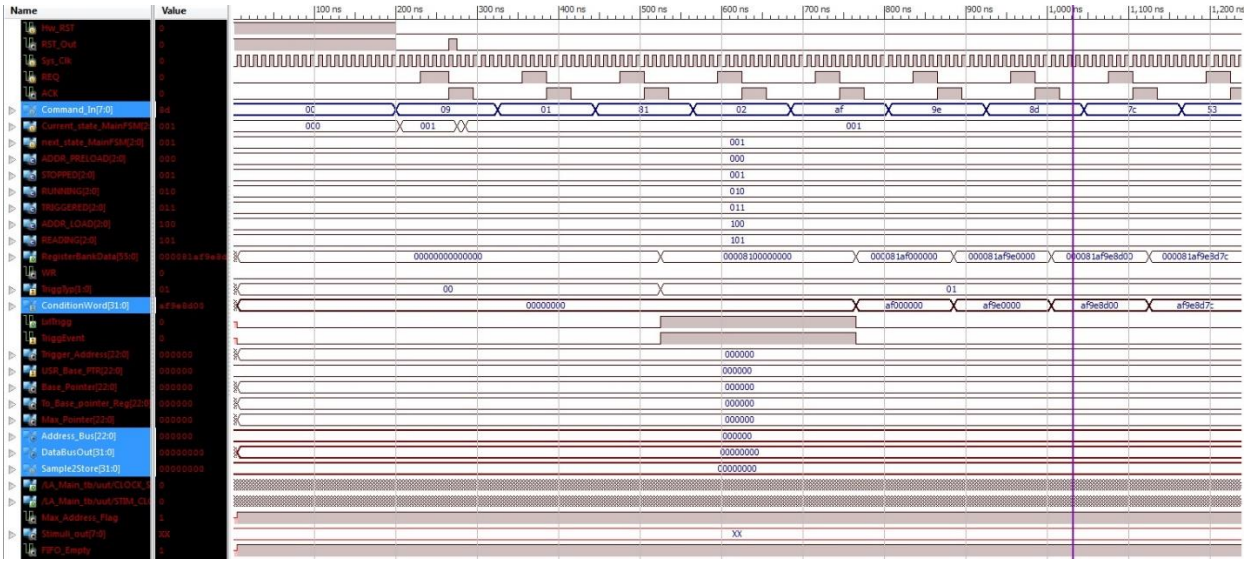


Figura 6.86 Comandos de configuración operando sobre el analizador lógico

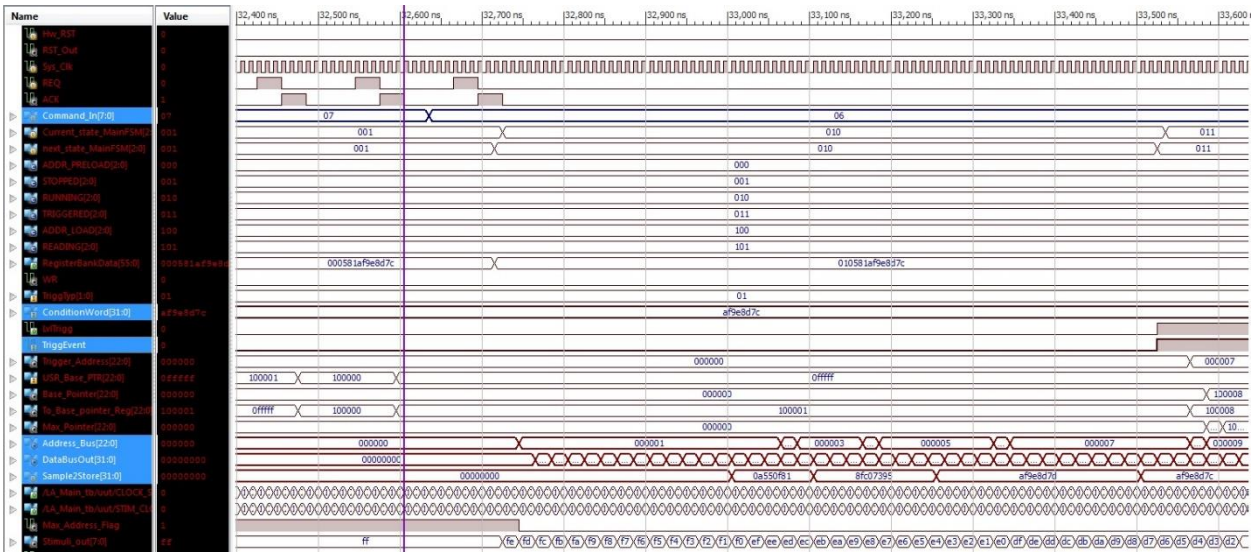


Figura 6.87 Comando de inicio de adquisición sobre el analizador lógico

En la Figura 6.87, se observa la actividad derivada de la ejecución del comando *STARTAQ*, que obliga a la FSM principal del analizador lógico a cambiar su estado a *RUNNING* (Ver sección 5.6) e iniciar la actividad de los buses. En la simulación pueden observarse la dinámica de los buses de direcciones (*Address_Bus*) y de datos para almacenamiento (*Sample2Store*), dado que se había configurado un muestreo transicional. Así mismo, es relevante el inicio de salida de estímulos, que se aplicarían sobre el sistema bajo prueba y provocan cambios en la salida del mismo. También hay que resaltar el cambio en el banco de registros, correspondiente al establecimiento del bit *STRTAQ* del registro *STATUS*, desde el inicio de la adquisición.

Los parámetros de temporización que aparecen en la Tabla 6.7 fueron obtenidos desde el proceso de síntesis con la herramienta ISE. La lógica del subsistema de configuración introduce el máximo retardo por camino combinacional de 8.813ns, debido especialmente a la lógica de decodificación que estructura este sistema

para el acceso al banco de registros y también a la componente combinacional de administración de la estructura FIFO asíncrona. Sin embargo, el 49% de este tiempo es debido al ruteo interno dentro del FPGA y el 51% restante corresponde a los retardos de propagación propios de la lógica utilizada. El módulo que presenta el menor retardo de propagación es el de sincronización de las señales de control, que puede operar a una frecuencia máxima de 755MHz, según estima la herramienta de síntesis. En la Tabla 6.14 se registran los parámetros de temporización para el sistema total, estimándose un máximo retardo por trayecto combinacional de 8.824ns. El 49.1% de este tiempo se debe a ruteo al interior del FPGA.

6.3.3 *Test* funcional de los módulos

La simulación de comportamiento para los módulos del subsistema de configuración requirió el acondicionamiento de señales de entrada que garantizaran su operación y permitiera verificar sus operaciones desde un *testbench*. Sin embargo, para hacerlo en un entorno práctico y bajo condiciones reales, se realizó un *test* a nivel físico mediante la implementación de una comunicación entre el subsistema propuesto y un microcontrolador dsPIC33E, en el que corría un *kernel FreeRTOS*. Sobre este controlador, se implementó un conjunto de tareas para obtener los parámetros especificados por el usuario desde una aplicación Web educativa y enviarlos al subsistema de configuración, de acuerdo con los procedimientos descritos en el capítulo 4. El microcontrolador opera hasta 70MHz, pero la frecuencia de reloj del subsistema propuesto se ajustó inicialmente a 100MHz y luego fue incrementada a 200MHz.

Se utilizó un analizador lógico externo 16801A de *Agilent* para capturar los patrones de las principales señales. Este instrumento presenta entre sus especificaciones: máxima tasa de muestreo de 500MHz, profundidad de memoria de muestreo entre 1M y 32M, un cable conector de 40 pines para 34 canales de adquisición, máxima frecuencia de reloj para estados de 250MHz, capacitancia de carga con sus conectores de 1.5pF. Estas especificaciones limitaron la verificación funcional modular del sistema a frecuencias superiores. A través de los puertos de este instrumento se permite la adquisición y análisis de las señales de control de la comunicación, las transacciones de datos y el contenido de los registros. Los siguientes comandos se enviaron al subsistema de configuración: *RST*, *CTRG* (configurando muestreo transicional y disparo por nivel – 0x83), TRGW (con el parámetro adicional 0x001451), *SFREQ* (Fosc) y *CHSAMP* (16-bit). Los principales patrones resultantes se muestran en Figura 6.88 y Figura 6.89, en las que se puede apreciar la señal *sync_REQ*, que corresponde al *REQ* sincronizado al reloj local. Esta señal habilita el registro de comando *REG_Command* para cargar el parámetro o comando entrante. *MUX_BANK_Ctrl* es una señal interna que selecciona cada registro del banco para ser capturado y visualizado por el analizador lógico externo en un puerto de 8-bit.

Solo hasta cuando un comando con parámetros adicionales se envíe (*CTRG*), el banco de registros mostraría algún tipo de actividad. El parámetro adicional de *CTRG* se escribe en el registro *CONF_TRIG* (0x04), como se muestra en la imagen aumentada de la Figura 6.90.

Los parámetros adicionales del comando *TRGW* se almacenan en los registros 0x03~0x00 (Ver la carga de cada uno de los parámetros al banco de registros desde la Figura 6.91 a la Figura 6.94). El *nibble* alto del registro *AQ_FORMAT* (0x05) se carga cuando *SFREQ* se ejecuta, mientras el *nibble* bajo lo hace mediante el comando *CHSAMP*. Estas asignaciones pueden observarse en la Figura 6.95 y Figura 6.96.

Tabla 6.13 Resumen de parámetros de temporización estimados para cada bloque del analizador lógico

Parámetro		Minimum period (ns)	Maximum frequency (MHz)	Min. global offset in before (ns)	Max. output required time before clock (ns)	Max. combinational path delay (ns)
ADDRESS_GENERATOR		3.380 Niveles de lógica: 25 59.6% lógica, 40.4% ruteo	295.836	6.268	4.435	6.788 Niveles de lógica: 4 72.5% lógica, 27.5% ruteo
Addr_Decoder		-	-	-	-	6.732 Niveles de lógica: 3 66.8% lógica, 33.2% ruteo
DATA_BUS_MGM		-	-	-	-	7.125 Niveles de lógica: 3 63.1% lógica, 36.9% ruteo
Edge_Trigger_Det		1.324 Niveles de lógica: 0 45.2% lógica, 54.8% ruteo	755.287	7.629	2.968	8.687 Niveles de lógica: 6 59.3% lógica, 40.7% ruteo
Lvl_Trigger_Det		-	-	-	-	7.678 Niveles de lógica: 4 61.8% lógica, 38.2% ruteo
DataSampl		4.986 Niveles de lógica: 4 32.3% lógica, 67.7% ruteo	200.562	6..422	5.724	6.849 Niveles de lógica: 3 65.6% lógica, 34.4% ruteo
Synchr_CtrlSign		1.324 Niveles de lógica: 0 45.2% lógica, 54.8% ruteo	755.287	5.509	2.941	6.446 Niveles de lógica: 3 69.7% lógica, 30.3% ruteo
REQ2En_Pulse		1.364 Niveles de lógica: 0 43.9% lógica, 56.1% ruteo	733.138	5.242	2.512	-
TRIGG_EVENT_PULSE		1.364 Niveles de lógica: 0 43.9% lógica, 56.1% ruteo	733.138	5.242	2.512	-
LA_COMM _CTRL	internal_clk	8.767 Niveles de lógica: 25 41.6% lógica, 58.4% ruteo	114.064	8.355	7.868	8.813 Niveles de lógica: 3 51.0% lógica, 49.0% ruteo
	stim_clk	6.908 Niveles de lógica: 4 23.1% lógica, 76.9% ruteo	144.756	7.235	7.157	
MAIN_CTRL_FSM		2.837 Niveles de lógica: 2 38.9% lógica, 61.1% ruteo	352.485	5.525	3.704	-

Tabla 6.14 Resumen de parámetros de temporización estimados para el analizador lógico

LA_MAIN	**Sys_Clk**	8.817 Niveles de lógica: 25 41.3% lógica, 58.7% ruteo	113.417	8.389	7.652	8.824
	USR_Clk	6.549 Niveles de lógica: 5 28.5% lógica, 71.5% ruteo	152.695	8.112	9.325	Niveles de lógica: 3
	STIM_Clk	7.022 Niveles de lógica: 5 26.1% lógica, 73.9% ruteo	142.414	7.285	6.371	50.9% lógica, 49.1% ruteo

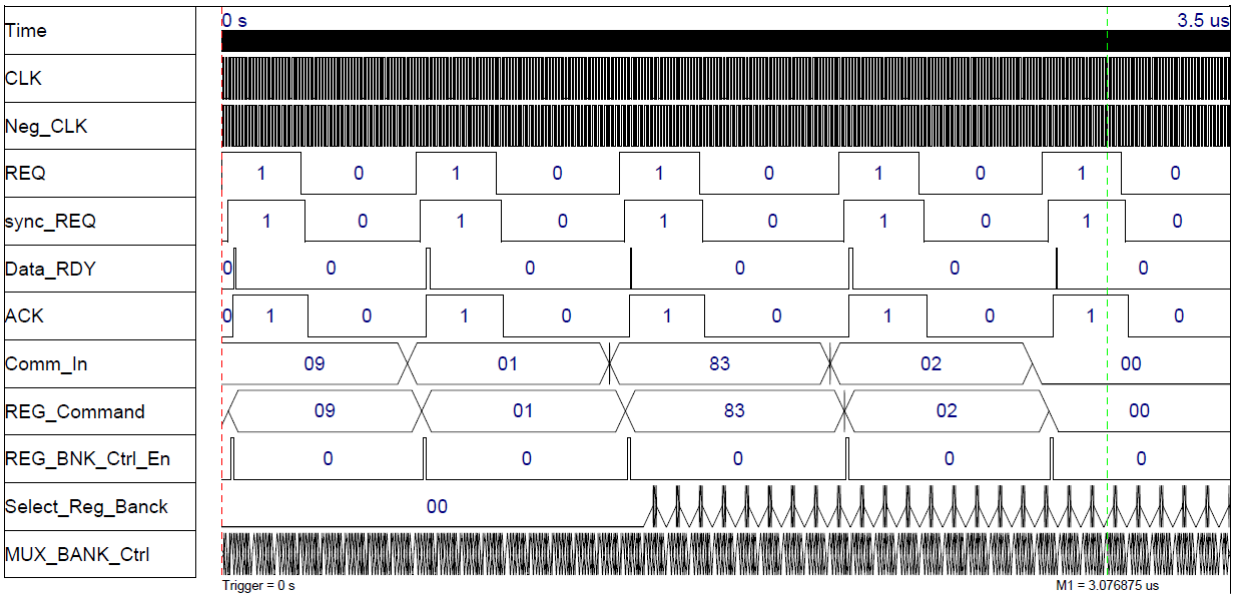


Figura 6.88 Transacciones RST, CTRG y TRGW capturadas mediante analizador lógico externo

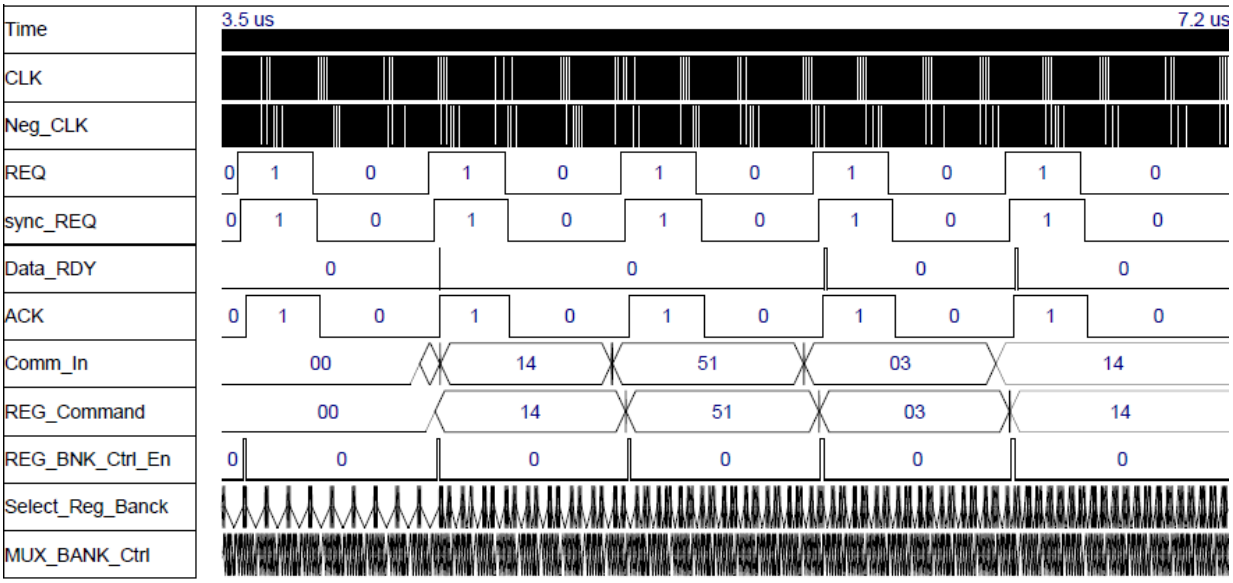


Figura 6.89 Parámetros de TRGW y transacciones SFREQ y CHSAMP capturadas mediante analizador lógico externo

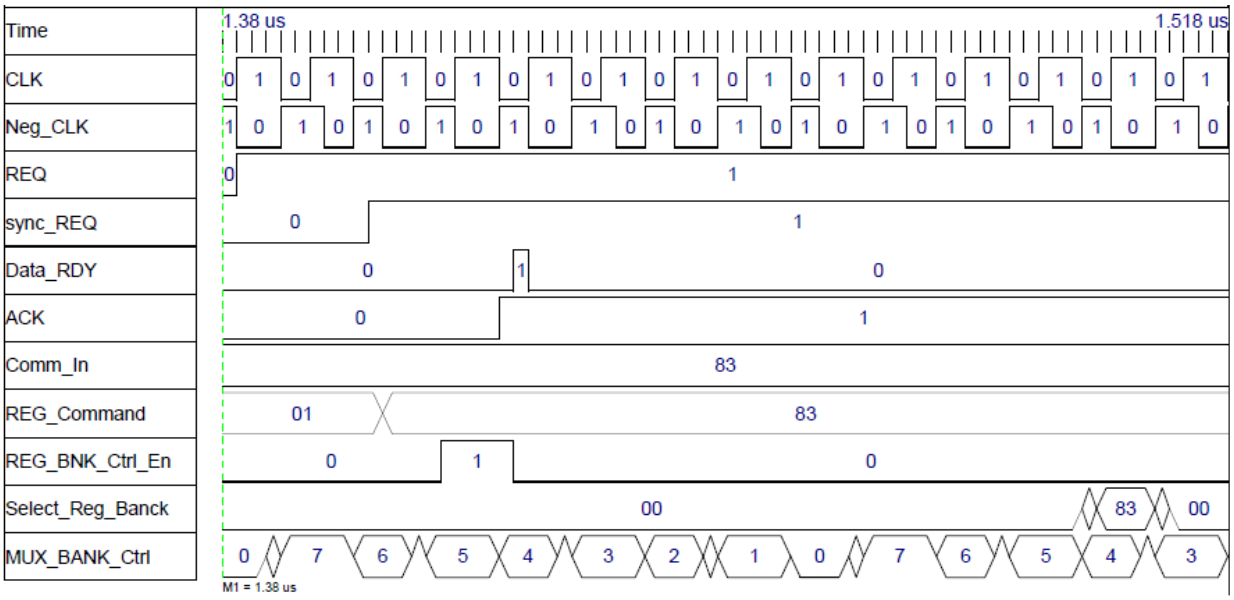


Figura 6.90 Imagen aumentada de una transacción CTRG y su parámetro adicional

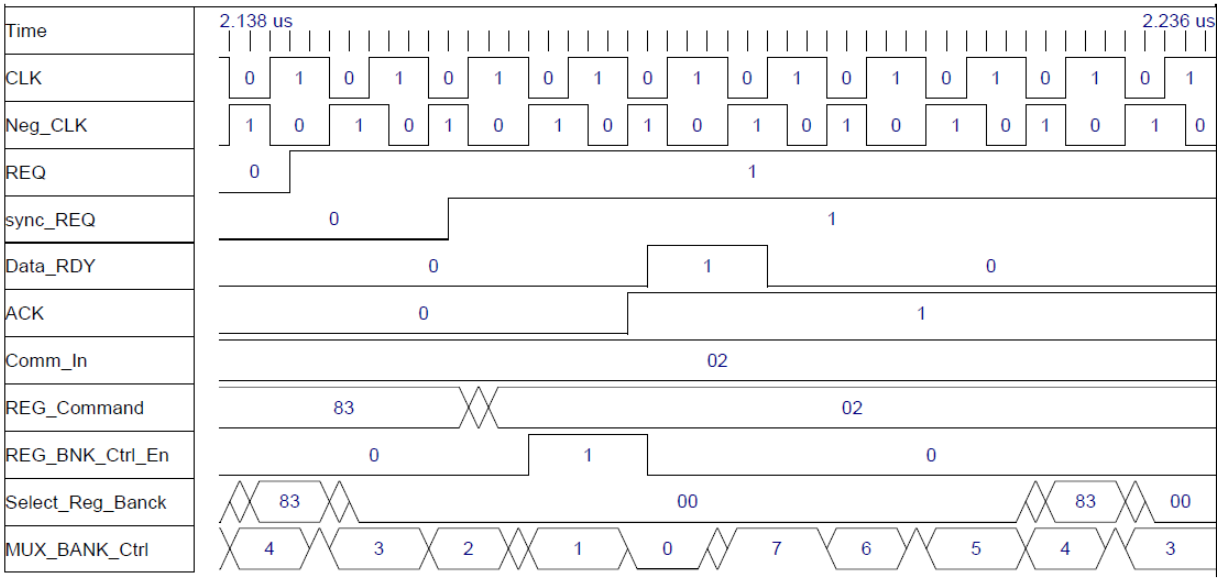


Figura 6.91 Imagen aumentada de una transacción TRGW

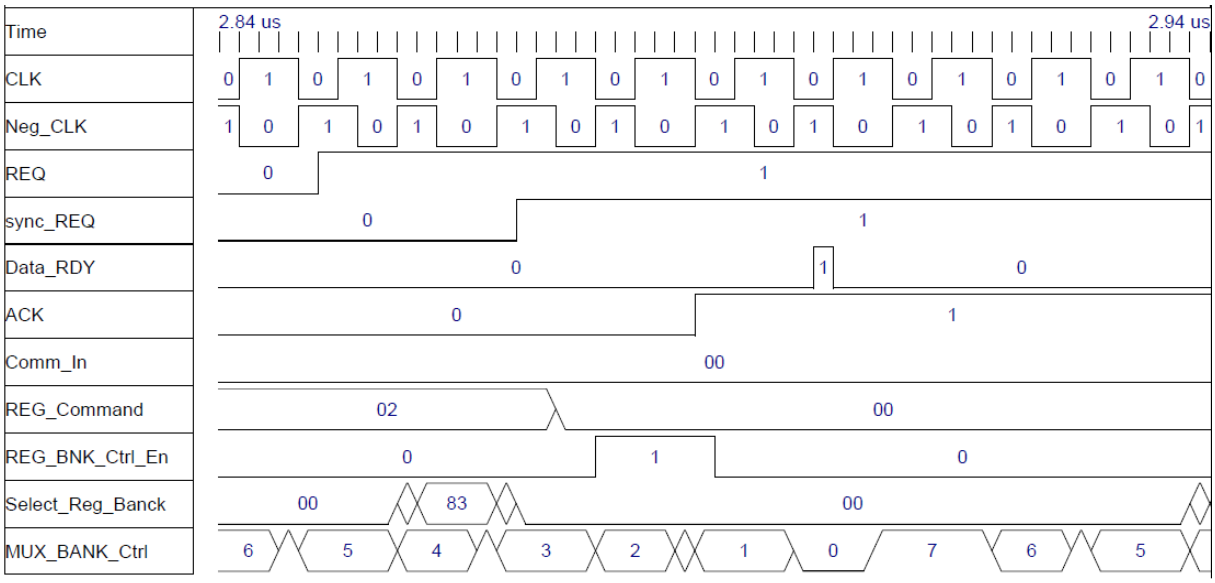


Figura 6.92 Imagen aumentada de una transacción TRGW y su parámetro adicional – byte 3

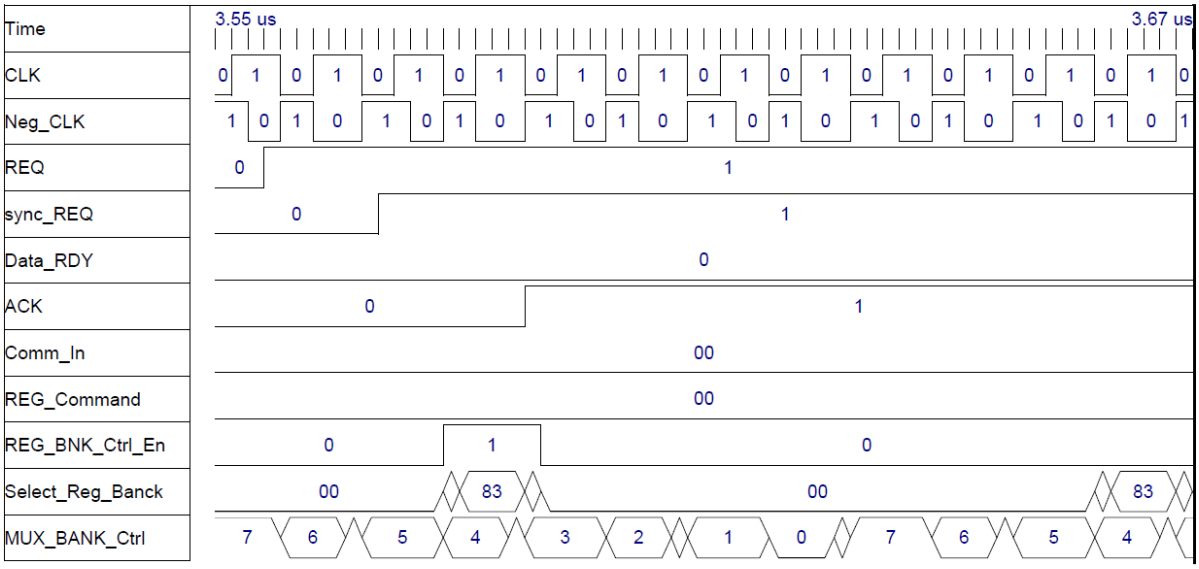


Figura 6.93 Imagen aumentada de una transacción TRGW y su parámetro adicional – byte 2

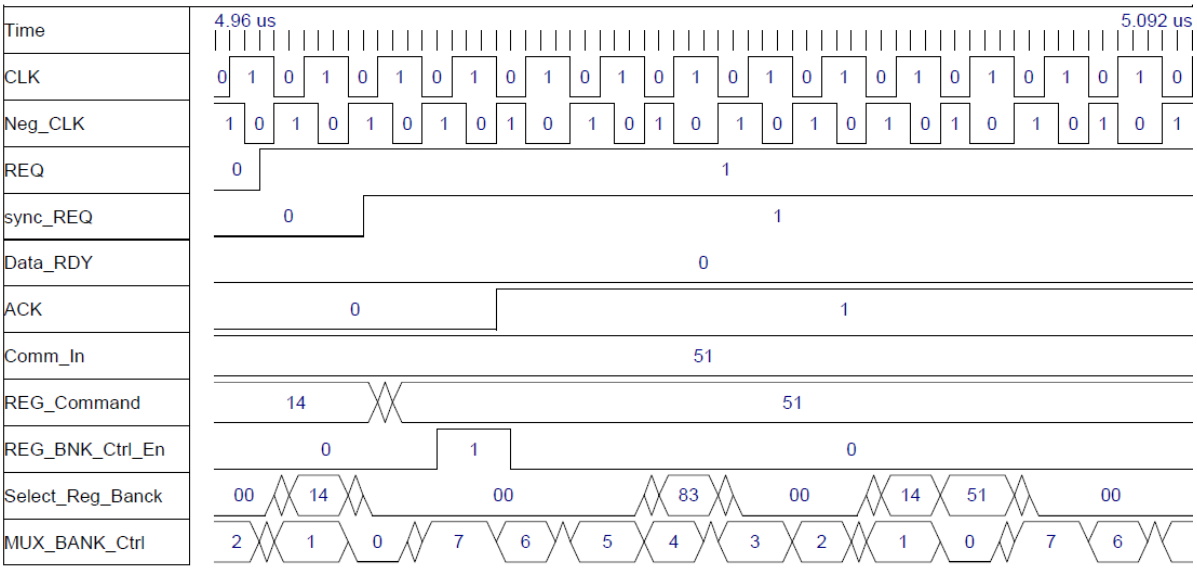


Figura 6.94 Imagen aumentada de una transacción TRGW y su parámetro adicional – byte 1, byte 0

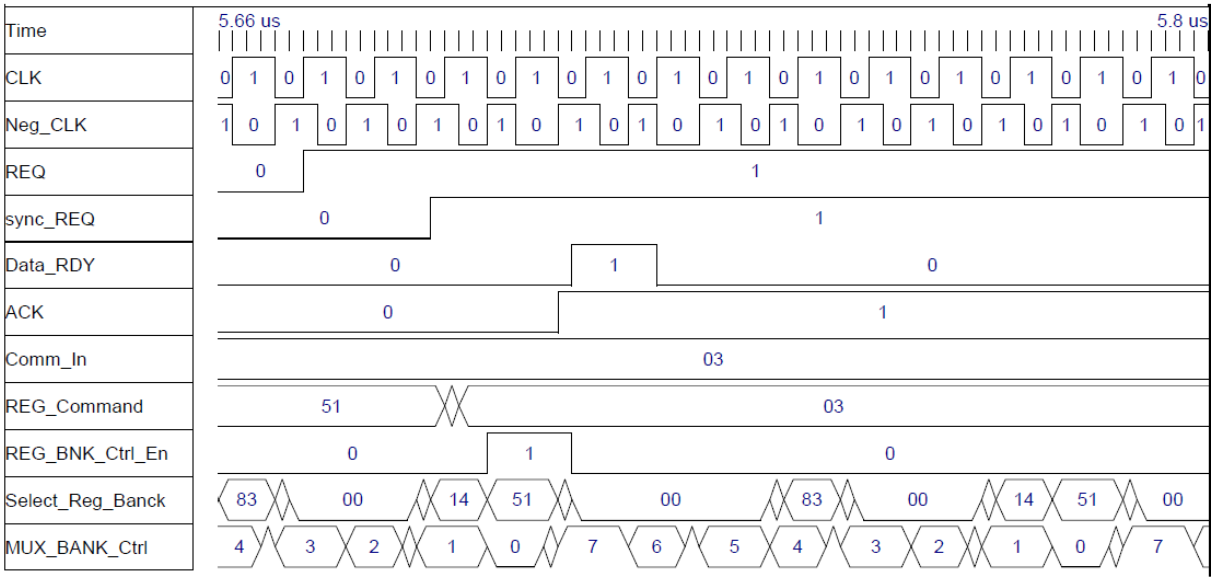


Figura 6.95 Imagen aumentada de una transacción SFREQ

El microcontrolador envía el comando *STM2SND* y una secuencia de estímulos para ser almacenados en la estructura *FIFO*. Esta transacción también fue capturada por el analizador lógico externo, obteniéndose los datos de la Figura 6.97 y la Figura 6.98, en las que se pueden observar también las banderas *FULL* y *EMPTY* que comunican el estado de la cola. Aunque la lectura de los estímulos se controla desde el sistema principal del analizador lógico, que define el momento en que éstos son aplicados sobre los canales del circuito bajo prueba, en esta fase de verificación se ordena la lectura de la cola de estímulos después que termina el

proceso de envío de los mismos (o escritura sobre la cola). Este proceso puede observarse en la imagen aumentada de la Figura 6.99, analizando la secuencia de salida y banderas de estado.

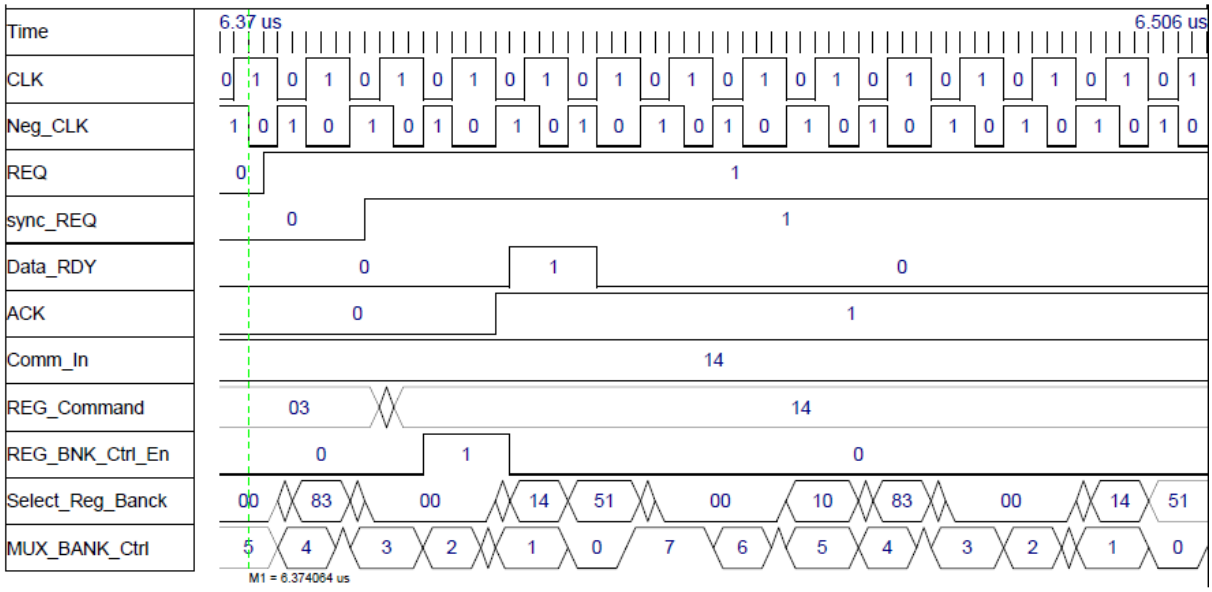


Figura 6.96 Imagen aumentada de una transacción CHSAMP

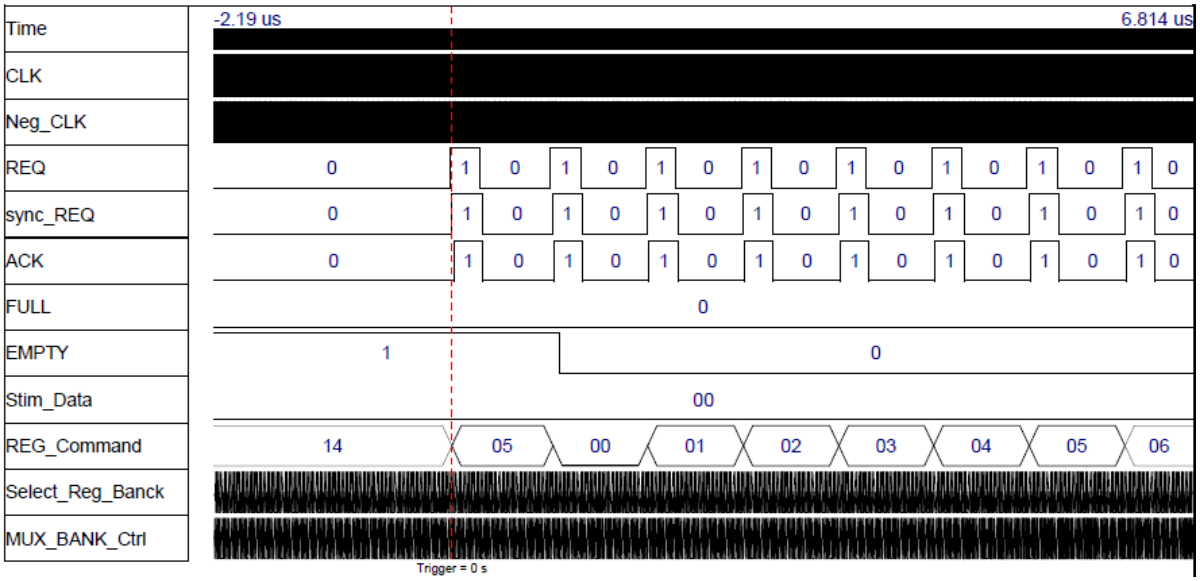


Figura 6.97 Transacción STM2SND capturada por analizador lógico externo

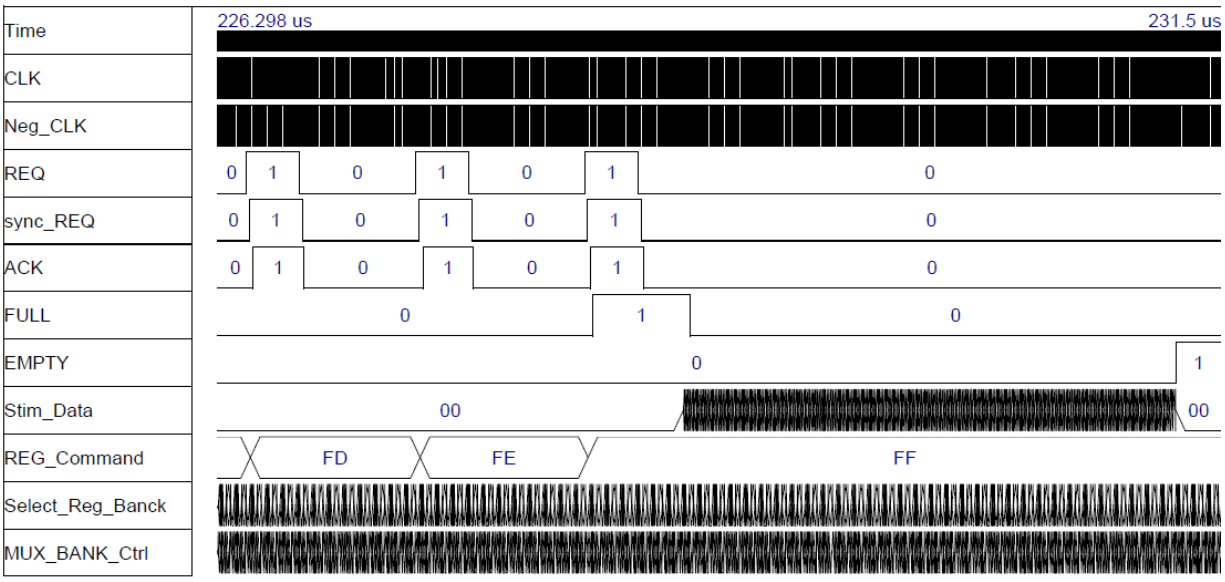


Figura 6.98 Fase final de una transacción STM2SND y lectura de estímulos desde estructura FIFO

El comando *PTRADJ* también fue enviado para verificar el desplazamiento del puntero a memoria de muestreo, desde la mitad del espacio de direccionamiento total para el número de canales ajustado. Recuérdese que el sentido del ajuste se define con base en el bit 4 del byte de comando, por lo que se espera un incremento de la dirección base al ser enviado 0x17 y un decremento de la misma si se envía 0x07. Esta transacción se muestra en la Figura 6.100 y Figura 6.101, en la que se ha configurado previamente una adquisición de 8 canales (Ver la dirección inicial del puntero en la Tabla 5.4)

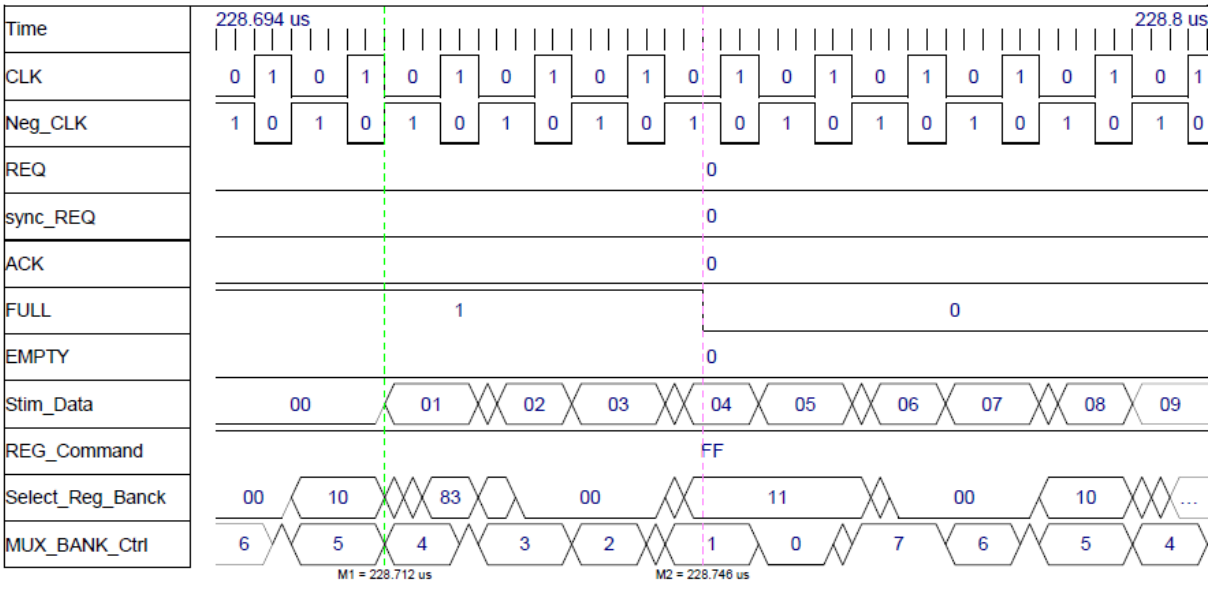


Figura 6.99 Imagen aumentada de lectura de estímulos

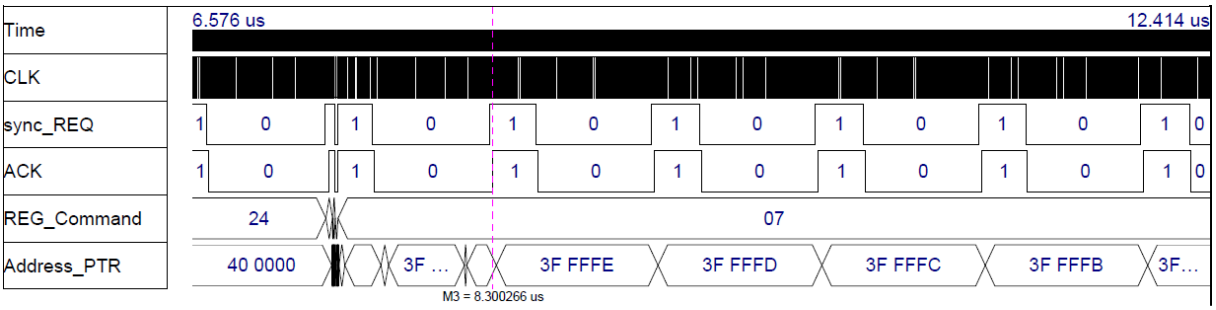


Figura 6.100 Transacción PTRADJ para decrementar el puntero a memoria de muestreo

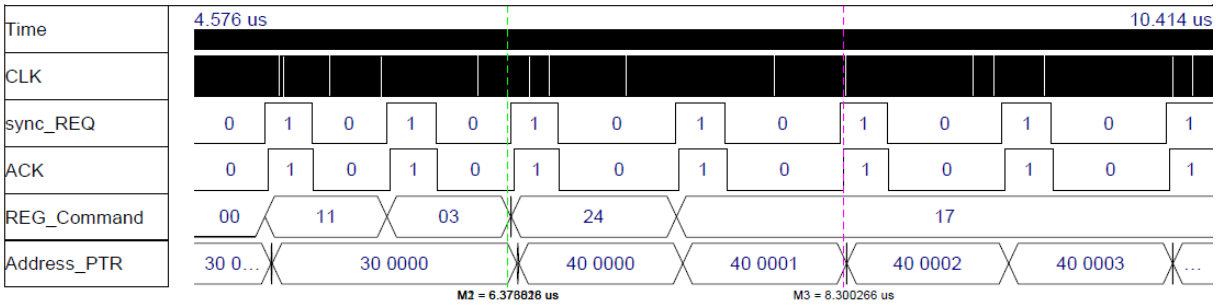


Figura 6.101 Transacción PTRADJ para incrementar el puntero a memoria de muestreo

Cuando el subsistema opera a 100MHz, la señal *ACK* se activa 20ns después de que *REQ* sincronizado se ha activado también. Cuando se envía el comando *PTRADJ*, el puntero a memoria de muestreo se establece después de 20ns del levantamiento de *ACK*. Esta señal, a su vez, se restablece a bajo 8ns después de que lo haga *sync_REQ*. Sin embargo, cuando el subsistema fue probado a 200MHz, todas estas medidas de tiempo fueron reducidas a la mitad. El margen de error de estas magnitudes es de 2ns, debido al período de muestreo del analizador lógico externo con el que se realizaron las capturas.

Por otro lado, se realizaron pruebas de generación aleatoria de eventos de disparo para la verificación funcional de los circuitos de disparo y direccionamiento, así como de la operación de la máquina de control principal del analizador lógico. En la Figura 6.102 se ilustra la actividad del registro de disparo desde la etapa de configuración hasta el instante en que ocurre un evento de disparo; así mismo, se observa la llegada de la orden de inicio de adquisición en el registro de comando. Adicionalmente, la Figura 6.103 registra la dinámica del bus de direcciones en una ventana de tiempo de ±28ns alrededor del suceso de disparo. Las señales de control al inicio, durante y final del disparo se muestran respectivamente en la Figura 6.104, Figura 6.105 y Figura 6.106.

La última fase de la adquisición, dada por el alcance del estado *READING* de la FSM principal se verifica mediante la herramienta *USBTrace* [123]. Este programa permite analizar las tramas de la comunicación USB, por lo que pueden identificarse los datos correspondientes a la adquisición del analizador lógico que son enviados mediante este protocolo al servidor. Los datos utilizados para las pruebas son los mismos vectores definidos en la Tabla 6.8.

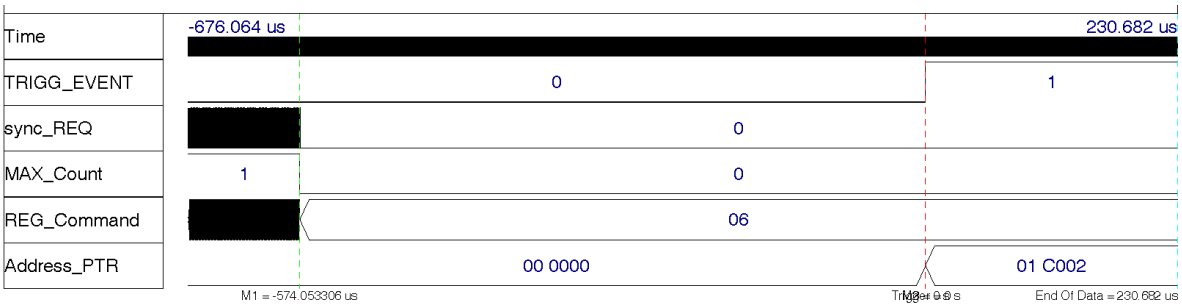


Figura 6.102 Analizador lógico en estado *RUNNING* y llegada de un evento de disparo

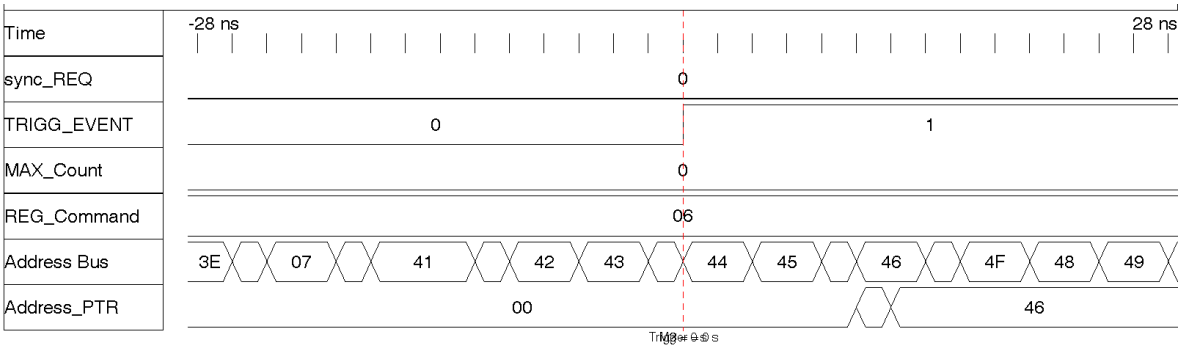


Figura 6.103 Estado del bus de direcciones y registro puntero de disparo

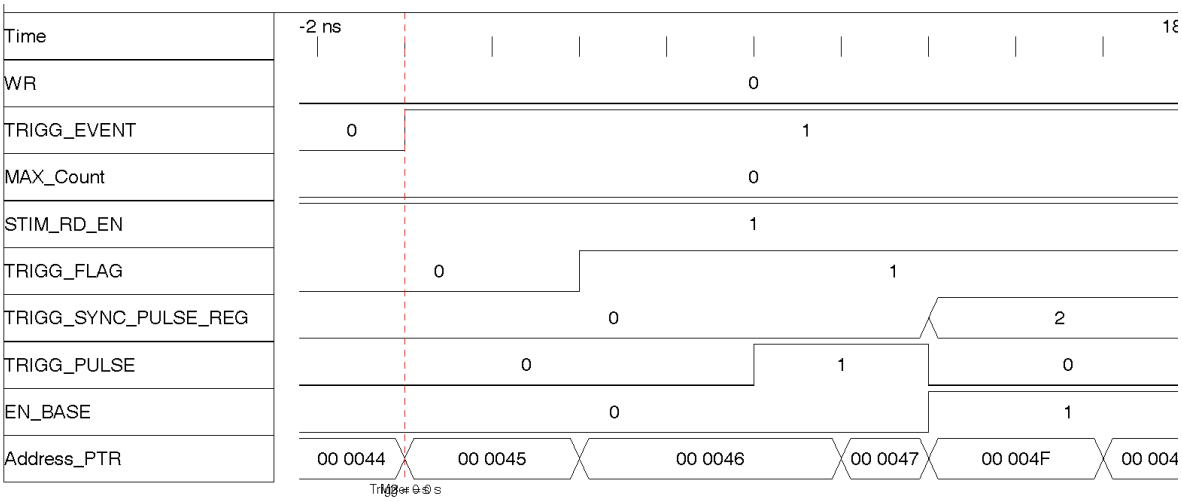


Figura 6.104 Dinámica de señales de control durante el inicio de un evento de disparo

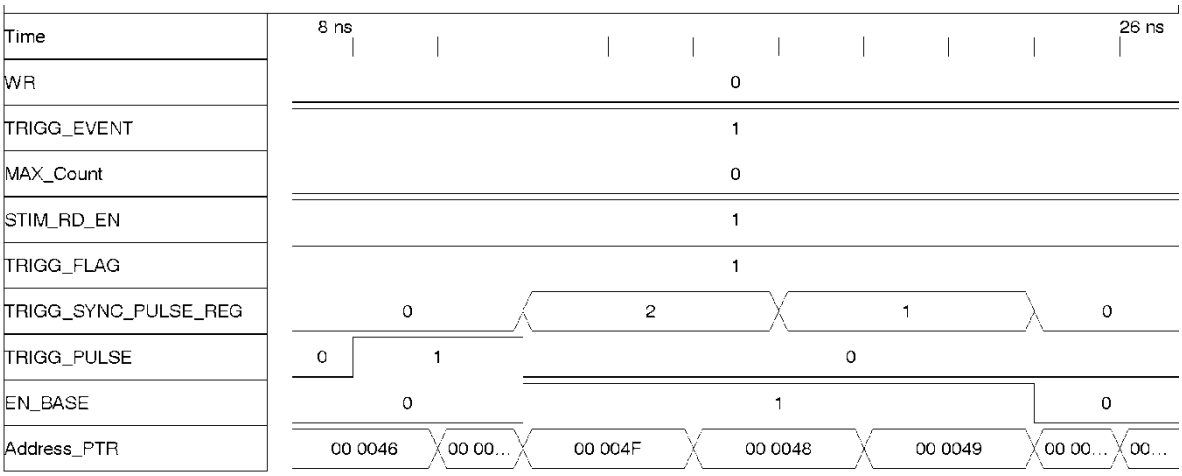


Figura 6.105 Dinámica de señales de control durante un evento de disparo

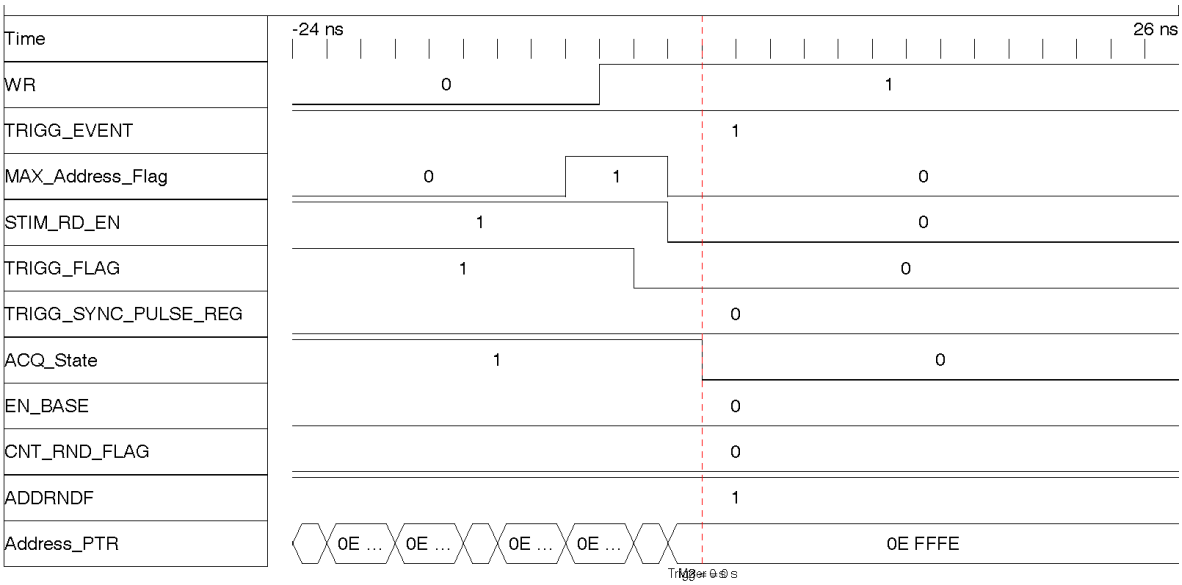


Figura 6.106 Dinámica de señales de control al finalizar el estado *TRIGGERED*

7. CONCLUSIONES Y TRABAJOS DERIVADOS

El interés de algunos sectores, particularmente el de la educación, en fortalecer e implementar estrategias didácticas para la enseñanza en modalidad a distancia o semipresencial en educación superior, ha promovido y mantenido la investigación en el tema de los laboratorios remotos, virtuales e híbridos en los últimos años y consecuentemente la proposición de modelos para su implementación en diferentes disciplinas.

El estado del arte muestra que, a pesar de los múltiples desarrollos en su campo, los laboratorios remotos representan una línea de investigación con un impacto importante en el sector educativo. Las múltiples disciplinas en este sector, la evolución y adaptación de las NTIC son la causa principal de la diversidad de modelos.

En ingeniería, la reflexión de la comunidad académica sobre su contexto educativo inmediato ha alcanzado un nivel trascendental. La iniciativa CDIO, la adopción y adaptación de metodologías de aprendizaje activo en los programas académicos de ingeniería revelan el ejercicio consciente por establecer modelos y estándares unificados que recuperarían el rol social de los ingenieros en formación como seres integrales.

Es imperativo establecer modelos de integración de las NTIC en las metodologías y didácticas de aula en ingeniería. Las modalidades semipresencial y distancia serían las beneficiarias primarias de tales modelos.

Los laboratorios remotos se introducen como herramientas esenciales, basadas en NTIC, sobre las didácticas de aprendizaje activo en ingeniería. Por esta razón, existe una fuerte tendencia por adoptar estándares, como el IEEE 1451 y el IEEE P1876, que permitan unificar aspectos arquitecturales de los laboratorios remotos e híbridos. Sin embargo, debe hacerse un cuidadoso seguimiento de la evolución de estas tendencias por su carácter técnico, lo que podría introducir una distancia importante con respecto al servicio educativo.

Entre las NTIC que merecen especial atención en el proceso de integración de los laboratorios remotos a las aulas de ingeniería se encuentran los servicios Web y el internet de las cosas (IoT – *Internet of Thngs*). El escenario planteado por estas tendencias da un lugar importante a los laboratorios dentro de las infraestructuras de red, en las que convergen herramientas al servicio educativo.

Por otro lado, el concepto de sistema informático refleja y explica la necesidad de contribuir al conocimiento también desde la innovación y el desarrollo tecnológico. La arquitectura general adoptada en este trabajo para el sistema informático corresponde a una de las más estables y usadas reportadas en la bibliografía, pero las principales contribuciones de este trabajo se ubican en el contexto educativo de una rama de la ingeniería electrónica, desde la aplicación Web de la componente software y la placa de experimentación de la componente hardware.

7.1 Desarrollo de la Componente Software

La aplicación Web propuesta en el capítulo 3 introduce aspectos novedosos en los laboratorios de acceso remoto para el aprendizaje de los sistemas digitales, ya que además de garantizar la ejecución de procesos

que involucran el *hardware*, está orientada también a facilitar la implementación de didácticas experienciales para el aprendizaje colaborativo y a distancia. La vinculación de herramientas de la Web 2.0 en un contexto educativo suministra a docentes y estudiantes un espacio interactivo para el aprendizaje, usando una plataforma híbrida concebida para tal fin e introduciendo este servicio educativo en las tendencias del internet de las cosas *IoT*.

La integración del CMS *Joomla* con el LMS *Moodle* en una aplicación Web orientada al apoyo de un curso de sistemas digitales resulta ideal para la adopción de estrategias alternativas de enseñanza/aprendizaje en este campo, como las propuestas en la iniciativa CDIO en el marco de la educación en ingeniería. Si bien representa una herramienta con servicios Web 2.0 integrados, permite que los docentes planifiquen e implementen estrategias de educación semipresencial, aprendizaje basado en proyectos y aprendizaje colaborativo. Cabe resaltar que los contenidos y estructura del curso implementado sobre la plataforma pueden ser fácilmente reestructurados por el administrador (o tutor) del mismo, dado el soporte de estas herramientas de gestión.

En el aprendizaje colaborativo se estimula el desarrollo de competencias de trabajo en equipo, para lo cual se pueden explotar las herramientas de *moodle* en un entorno virtual y se pueden integrar aplicaciones basadas en *java* para juegos de rol virtuales. Aunque, al activar la plataforma *hardware* de laboratorio remoto las pruebas y administración sobre dicho módulo son agendadas para un solo usuario por turno, los compañeros de curso pueden ver lo que sucede con el laboratorio en formato de *videostream*, y comunicarse bajo sala de chat, foros, redes sociales, o documentos compartidos en la nube. Por otro lado, las actividades que el docente incluye en la plataforma pueden estar orientadas al planteamiento de casos con documentos y videos de soporte para entrenamiento previo; éstas pueden estar ocultas o visualizarse según lo estime conveniente.

Por otra parte, el uso del protocolo USB para la comunicación entre el servidor de *hardware* y la tarjeta de experimentación flexibiliza la portabilidad del laboratorio, facilitando la configuración de conectividad a sistemas de cómputo, en los que se ha consolidado esta interfaz, esencialmente para almacenamiento masivo e interconexión de dispositivos.

Esta propuesta se desarrolló bajo el concepto de software libre, ya que herramientas como *Joomla*, *Moodle*, *wampServer* y *Tomcat* son de licencia *open source*; así mismo ocurre con los lenguajes PHP, *JavaScript* y *Java* y la biblioteca USB. Aunque la implementación del sistema se hizo sobre plataforma *Windows 7*, las herramientas utilizadas son multiplataforma e independientes de la arquitectura (32 o 64 bits).

7.2 Desarrollo de la Componente Hardware

En el aprendizaje en ingeniería electrónica y especialmente en sistemas digitales es importante fomentar el desarrollo de habilidades con procesos de experimentación; por esto, la herramienta propuesta ofrece a los estudiantes un acercamiento a una plataforma reconfigurable que incorpora un instrumento de validación. El acceso desde internet a una plataforma *hardware* robusta que brinda los servicios de un laboratorio para sistemas digitales, representa un aporte en alto grado a las didácticas del aprendizaje experiencial en ingeniería electrónica.

El trabajo presentado propone un prototipo *hardware* para un laboratorio remoto con un importante grado de innovación, debido a que integra un analizador lógico que permite validar las funciones dentro de un FPGA sin ocupar espacio dentro de su lógica configurable.

Se estableció la importancia de un método de diseño PCB siguiendo normas y documentos de los fabricantes para obtener un resultado exitoso en el desarrollo de la plataforma *hardware*. El proceso se dividido en cuatro fases (documentación y análisis de las tecnologías de diseño PCB disponibles, diseño del sistema, fabricación del circuito impreso y validación).

Durante el diseño y fabricación de los prototipos se ha evidenciado la dificultad que implica el proceso de depuración, ya que se presentan diferentes dificultades que se imposibilitan prever en fase de simulación. Algunas de estas dificultades son relativas a errores de fabricación y de sincronización que fueron resueltas ajustando resolución y geometría del trazado del PCB.

El uso de un *kernel* RTOS para el sistema de control de la tarjeta facilitó el diseño e implementación de los protocolos de comunicación con los demás sistemas presentes en la tarjeta de entrenamiento. Aunque la ejecución de las principales tareas muestra cierto determinismo en la secuencia de uso del recurso de procesamiento, el sistema es susceptible de actualizarse para migrar a un soporte de acceso multiusuario simultáneo. Esto implica que sobre la plataforma hardware propuesta, es posible implementar un *firmware*, que residiría sobre el MCU para ser actualizado en versiones posteriores. Estas versiones contemplarían la configuración dinámica del FPGA y adición de otras funciones E/S.

7.3 Analizador Lógico para Depuración Remota

Los analizadores lógicos comerciales tienen un alto nivel en prestaciones, lo que justifica su elevado coste. Actualmente, estos equipos se basan en arquitecturas PC con interfaces internas de propósito específico que comparten recursos del sistema; por esta razón, la mayoría son equipos de cómputo con una aplicación específica para el uso de estas interfaces hardware que establecen compatibilidad con sistemas operativos determinados.

De acuerdo con la revisión bibliográfica, las arquitecturas de analizadores lógicos embebidos tienen menos prestaciones que estos equipos de instrumentación comerciales, pero se desarrollan en función de la aplicación específica que les demanden. Esto explica las publicaciones de patentes en este campo, con arquitecturas ampliamente semejantes pero que introducen aspectos y/o funciones con propósitos particulares.

La arquitectura propuesta y descrita en este capítulo corresponde a un analizador lógico que se conecta en una misma tarjeta con un microcontrolador y un FPGA, con el objetivo de soportar los procesos de depuración y prueba de funciones lógicas implementadas por el usuario en este dispositivo configurable. Entre sus prestaciones se destacan el muestreo transicional, el disparo pre-post por nivel o flanco, la configuración de 32, 16 u 8 canales, memoria de muestreo de hasta 8MB, un subsistema para el almacenamiento y la aplicación de estímulos, así como un subsistema para la configuración y operación del analizador lógico con base en un conjunto de comandos. Esto último facilita la administración del instrumento embebido desde otro sistema digital.

7.4 Conclusiones Generales y Trabajos Derivados

En la educación en ingeniería electrónica es esencial el fomento del desarrollo de habilidades mediante didácticas experienciales en el marco de un aprendizaje activo. La revisión bibliográfica muestra esta tendencia, cuya discusión era propia de los profesionales en educación y pedagogía, pero que actualmente converge en un contexto que es competencia de los propios programas de ingeniería. La discusión y

adopción de didácticas para el aprendizaje activo y el aprendizaje basado en proyectos y/o problemas en el marco de la educación en ingeniería explican la profundidad de estas reflexiones, muchas de ellas formalizadas en lo que constituye actualmente la iniciativa CDIO.

Con este trabajo se ha logrado un aporte importante desde el desarrollo tecnológico a un campo particular de la educación en la ingeniería electrónica: los sistemas digitales. En este contexto, se proyectó un sistema informático que apoya la implementación de didácticas de aprendizaje activo en la educación semipresencial en esta disciplina, considerando herramientas de las TIC en sus componentes software y hardware, así como aspectos arquitecturales para el soporte de las etapas de programación y prueba de funciones lógicas desde un sitio remoto. A diferencia de otros trabajos, el sistema desarrollado permite a su administrador (generalmente el tutor) planificar actividades y políticas de acceso al recurso, establecer un contexto real de trabajo, configurar y valorar aspectos de trabajo en equipo y uso del recurso, en el marco de su estrategia de enseñanza. Así mismo, el usuario que tiene acceso a la plataforma hardware tiene toda el área del FPGA disponible para la implementación de su sistema y no consume recursos importantes de la misma para la realización de pruebas con el analizador lógico. Todos estos procesos son manejados desde una interfaz gráfica de usuario amigable e intuitiva para el usuario.

La interacción entre el usuario y la plataforma hardware remota del sistema informático fue desarrollada con base en los criterios y perfiles de los usuarios del mismo, así como el soporte de servicios Web 2.0 que se constituyen como un recurso fundamental de las TIC para soportar didácticas de aprendizaje colaborativo en modalidad semipresencial. Así, en esta interfaz de usuario se propuso la integración de *Joomla*, *Moodle* y la administración de la placa de experimentación, dados su propósito académico y la necesidad de organizar contenidos, actividades, estadísticas entre otros recursos relevantes para mejorar la relación entre los actores del proceso de aprendizaje y el seguimiento del mismo.

Desde un enfoque más técnico, la implementación de los gestores de aprendizaje y contenido en la componente software del sistema informático sugirió también la unificación de sus bases de datos, en las que reside información relevante de los usuarios, como por ejemplo las credenciales de acceso. La interfaz de usuario desarrollada implementa un mecanismo de autenticación básico de *login* y *passwod*. Cabe recordar que *Joomla* y *Moodle* definen por defecto el MD5 como algoritmo de cifrado para contraseñas; aunque esto se conservó en la implementación del sistema, es posible modificar las políticas de credenciales de acceso.

La placa de experimentación diseñada e implementada para el sistema informático conforma un sistema embebido con un FPGA, un microcontrolador, un analizador lógico, interfaces de E/S, expansión y comunicación USB, cuyo costo de fabricación fue del orden de $220USD. Es importante resaltar que este sería el costo por fabricación de prototipo, que disminuiría ostensiblemente si se contrata la fabricación masiva de tarjetas. Sin embargo, consultando los proveedores de más amplia penetración en el sector educativo en este tipo de aplicaciones, se puede observar que el costo de un laboratorio similar al propuesto, sumando todos los componentes que lo constituyen, es superior a $5000USD, pues tan solo un analizador lógico externo oscila entre los $1500USD y $8000USD, dependiendo de sus prestaciones.

Es claro que los analizadores lógicos comerciales tienen prestaciones ciertamente superiores a los embebidos y su conexión a una plataforma de acceso remoto, según lo evidencia la bibliografía, requiere el diseño y adaptación de interfaces. Para reducir costos y garantizar la totalidad del área del FPGA para el uso de proyectos de usuario, la placa de experimentación del sistema propuesto incorpora un analizador lógico desarrollado también como producto de esta investigación. Su arquitectura introduce aspectos importantes como una interfaz de comunicación asíncrona y configuración basada en comandos que puede operar a más de 200MHz y que permite el envío de señales estímulo para el sistema bajo prueba. Así mismo,

se logró la implementación, en el mismo subsistema de configuración, de una estructura para el almacenamiento de estas señales y su aplicación al SUT a una frecuencia configurable por el usuario e independiente de la de muestreo. Un banco de registros soporta la unidad principal de control, lo que puede permitir alteración de parámetros aún durante un proceso de adquisición de datos. La arquitectura propuesta implementa también muestreo transicional como alternativa configurable por usuario y longitud de registro a 32, 16 u 8 canales, lo que permite administrar más eficientemente el recurso de memoria de muestreo. Esta arquitectura fue sintetizada mediante la herramienta *ISE Design Suite* de *Xilinx*, que estima el consumo de potencia en 38mW.

Por otro lado, las pruebas de usabilidad del sistema fueron orientadas a docentes y estudiantes de los programas de ingeniería electrónica de la Universidad del Valle y la Universidad del Quindío, obteniéndose resultados satisfactorios en su pertinencia y operación. Por tanto, el impacto sobre este campo disciplinar de la ingeniería se resume en el soporte que brinda el sistema informático para el desarrollo de habilidades como las siguientes: desarrollar y evaluar nuevas tecnologías; definir conceptos, funciones y arquitectura de sistemas; desarrollar modelos e interfaces de sistemas; concebir el proceso de fabricación de equipos electrónicos; concebir pruebas y verificaciones; proponer procesos de integración hardware-software.

Así, es indispensable encaminar un trabajo continuo sobre los resultados de esta investigación de manera que sea medido el impacto de la implementación, en el sistema informático propuesto, de cursos básicos y avanzados del pregrado en ingeniería electrónica y afines. La razón fundamental de ello es que la medición del impacto sobre los procesos de enseñanza-aprendizaje no es inmediata y requiere de su aplicación y análisis por al menos tres períodos académicos. Así mismo, deben articularse mecanismos de autoevaluación que permitan la adaptación y actualización de las plataformas hardware y software del sistema.

La proyección e impacto del sistema informático propuesto se refleja también en los siguientes trabajos derivados de su desarrollo:

- *Módulo de entrenamiento basado en FPGA con analizador lógico embebido*. Trabajo de investigación avalado y financiado por la Universidad del Valle.
- *Sistema embebido para el entrenamiento de habilidades en el diseño electrónico digital usando TICs*. Trabajo de investigación avalado y financiado por el Ministerio de Educación Nacional, en el marco de su programa RENATA y su "convocatoria para conformar un banco de proyectos de investigación elegibles, en innovación educativa con uso de las Tecnologías de la Información y la Comunicación 2011-2012"
- *Desarrollo de un laboratorio de experimentación para la implementación de didácticas de aprendizaje activo en ingeniería electrónica con base en la iniciativa CDIO*. Trabajo de investigación en fase de aprobación presentado a la Universidad del Quindío para la vigencia 2015-2016.
- *Desarrollo de una aplicación cliente/servidor para la experimentación remota en sistemas digitales*. Tesis de pregrado, con distinción meritoria, desarrollada en la Universidad del Valle.
- *Desarrollo de un módulo de comunicación y administración para una tarjeta de entrenamiento basada en TICs*. Tesis de pregrado desarrollada en la Universidad del Valle.
- *Desarrollo de una aplicación para terminales móviles para la administración de una placa de experimentación remota de sistemas digitales*. Tesis de pregrado en desarrollo en la Universidad del Valle.

- *Diseño e implementación de un servidor Web embebido en un microcontrolador para gestionar el laboratorio remoto desarrollado por el grupo GADyM*. Tesis de pregrado en desarrollo en la Universidad del Valle.
- *Desarrollo de un sistema multiplataforma para el acceso paralelo remoto a un recurso programable por múltiples usuarios*. Tesis de maestría en desarrollo en la Universidad del Valle.

Los trabajos de investigación, las tesis de pregrado y de maestría que se encuentran actualmente en desarrollo revelan la vigencia del tema y aportes realizados en el contexto de esta tesis doctoral. El soporte para el acceso al sistema informático desde terminales móviles, la integración del servidor de recursos en una misma placa de experimentación, el soporte para el acceso simultáneo de múltiples usuarios hacia el FPGA mediante configuración dinámica, la incorporación de niveles de caché, controladores DRAM y técnicas para incrementar la frecuencia de muestreo en la arquitectura del analizador lógico, así como el diseño de un modelo de implementación de estrategias de aprendizaje activo con base en el recurso que define este espacio de trabajo de tipo laboratorio, son algunos de los trabajos más significativos para abordar en el corto plazo a partir de la proyección de esta investigación y los resultados obtenidos en su desarrollo.

Cada una de las interfaces de hardware y software fueron diseñadas con capacidad para proyectar y extender el sistema. Por lo tanto, algunos de los trabajos más relevantes para su realización consisten en explotar esas funcionalidades y validar las propuestas metodológicas que sugiere el uso del laboratorio en cursos plenos o completos. Para el soporte a nivel de educación post-gradual resulta necesaria la incorporación de interfaces especializadas, que pueden adaptarse sobre los buses de expansión implementados. Así mismo, la información correspondiente a la evolución temática, organización de contenidos y recursos tendrá que ser evaluada desde el uso adecuado de la componente software.

Finalmente, las siguientes publicaciones han surgido de este trabajo de investigación:

- *Integrating Learning and Web-Content Management Systems on Digital Systems Teaching*. Artículo en evento *4th IEEE Colombian Workshop on Circuits and Systems* y publicado en *IEEEXplore*.
- *Experimentación Remota para el Análisis de Dispositivos Ópticos y de Potencia*. Artículo en evento *Eleventh Latin American and Caribbean Conference for Engineering and Technology, LACCEI 2013*, realizado en *Cancún – México*.
- *Integración de un Hardware Configurable y una Aplicación basada en CMS-LMS para e-learning de Circuitos Digitales*. Artículo en evento *Décima Segunda Conferencia Iberoamericana en Sistemas, Cibernética e Informática, CISCI 2013*, realizado en Orlando – Florida.
- *Remote Laboratory for e-Learning of Digital Systems Design*. Artículo en revista *Asian Transactions On Fundamentals Of Electronics, Communication And Multimedia*, v.02 fasc.05 p.11 - ,2012.
- *Herramienta en Línea para la Programación y Depuración Remota de Funciones Lógicas Digitales*. Artículo en revista *Ingeniería Y Competitividad*, v.15 fasc.1 p.79 - 91 ,2013.
- *Hardware to Configure an Integrated Logic Analyzer with Educational Purposes.* Artículo sometido a revista *Information Technology and Control*.

8. REFERENCIAS

[1] M. Duque. “COMPETENCIAS , APRENDIZAJE ACTIVO E INDAGACIÓN : UN CASO PRÁCTICO,”, Revista Educación en Ingeniería.Vol. 1, núm. 2. pp. 7–18, 2006.

[2] C. Andrés, T. Suárez, E. María, and G. Agudelo, “Aprendizaje activo en cursos básicos de ingeniería : un ejemplo en la enseñanza de Dinámica Resumen,”. Uni-pluri/versidad, vol. 10, no. 2, 2010.

[3] H. DIETRICH, K. HENKE, and N. LUDWIG, “Remote Labs versus Virtual Labs for Teaching Digital System Design,” in *International Conference on Computer Systems and Technologies - CompSysTech*, 2005.

[4] CONGRESO DE COLOMBIA, “Ley N° 1341 del 30 de Julio de 2009.” .

[5] J. G. SCHUTTE, “Virtual Teaching in Higher Education: The New Intellectual Superhighway or Just Another Traffic Jam?”. OCLC WorldCat. California State University, Northridge, 1997.

[6] U.S. DEPARTMENT OF EDUCATION., “Evaluation of Evidence-Based Practices in Online Learning, Center for Technology in Learning,” 2009.

[7] G. CARDONA OSSA, “Educación virtual un paradigma para la democratización del conocimiento,” in *Modelo educativo virtual para educación básica y media y superior en Colombia*, Bogotá (Colombia), 2002.

[8] “LABORATORIO VIRTUAL CAE,” *Numérica Ltda. Bucaramanga*, 2003. [Online]. Available: http://www.numerica.com.co/cae-lab/es/informacion/default.php.

[9] “RESOURCE CENTER FOR ENGINEERING LABORATORIES ON THE WEB,” 1995. [Online]. Available: http://chem.engr.utc.edu/.

[10] “VIRTUAL LAB FOR CONTROL ENGINEERING EDUCATION ON THE WEB.” [Online]. Available: http://www.esr.ruhr-uni-bochum.de/esr/cs/vclab/vclab.html.

[11] P. Kämmerling, A. Ackens, H. Loevenich, and A. Borga, “FPGA Configuration by TCP/IP and Ethernet,” *Real-Time Conf. 2007 15th IEEE-NPSS*, 2007.

[12] F. Alam, “Technology-Enhanced Laboratory Experiments in Learning and Teaching,” in *Using Technology Tools to Innovate Assessment, Reporting, and Teaching Practices in Engineering Education*, Editorial Advisory Board. Australia, p. 292, 2014.

[13] I. I. Goodof, “The role of the laboratory in staff development.,” *J. Maine Med. Assoc.*, vol. 43, no. 2, pp. 51–52, 1952.

[14] CDIO, “The CDIO™ INITIATIVE.” [Online]. Available: http://www.cdio.org/.

[15] A. Maiti and B. Tripathy, “Remote Laboratories : Design of Experiments and Their Web Implementation,” *Educ. Technol. Soc.*, vol. 16, no. 3, pp. 220–233, 2013.

[16] C. Tr, H. Markus, and P. Bettina, “Environments for Remote Teaching in Embedded Systems Courses.”, Invited paper. Embedded Computing Systems Group, Vienna University of Technology,

[17] M. Prince, “Does Active Learning Work ? A Review of the Research,” *J. Eng. Educ.*, vol. 93, no. July, pp. 223–231, 2004.

[18] E. F. Crawley and W. A. Lucas, “The CDIO Syllabus v2 . 0 An Updated Statement of Goals for Engineering Education,” in *Proceedings of the 7th International CDIO Conference*, 2011.

[19] E. Calvo, U. Zuleta, Gangoiti, and J. M. López, “Laboratorios remotos y virtuales en enseñanzas técnicas y científicas,” Informe Proyecto Investigación, Escuela Técnica Superior de Ingeniería de Bilbao, 2009

[20] S. Dormido, “Control Learning: Present and Future,” ScienceDirect, Vol. 28, Issue 1, pp. 115-136, 2004.

[21] J. García-Zubia, P. Orduña, J. Irurzun, I. Angulo, and U. Hernández, “Integración del Laboratorio Remoto WebLab-Deusto en Moodle,” Universidad de Deusto, 2009

[22] S. Gröeb, M. Vetter, B. Eckert, and H. Jodl, “Remotely Controlled Laboratory (RCL): Experimenting from a Distance,” *En European Journal of Physics 2007, Universidad de Kaiserslautern*. .

[23] S. H. Chen, R. Chen, V. Ramakrishnan, S. Y. Hu, Y. Zhuang, C. C. Ko, and M. Chen, “Development of Remote Laboratory Experimentation through Internet,” National University of Singapore.

[24] C. S, V. Ramakrishnan, H. S, and Z. Y, “Development of Remote Laboratory Experimentation through internet,” National University of Singapore, 2001.

[25] E. L. López, *Plataformas Virtuales*. Diplomado Superior en Gestión de Proyectos. Universidad Israel, 2009.

[26] I. GUSTAVSSON, “Remote Laboratory Experiments in Electrical Engineering Education,” *Fourth IEEE Int. Caracas Conf. Devices, Circuits Syst.*, 2002.

[27] L. H. A, C. SALTZMANN, D. GILLET, and H. BOUZEKRI, “Information Technology Enhanched Learning in Distance and Conventional Education,” *IEEE Trans. Educ.*, vol. 42, 1999.

[28] A. Rosado, M. Bataller, J. Guerrero, and J. Vila, “UN LABORATORIO DE DISEÑO DIGITAL EN VHDL : APRENDIZAJE POR PROYECTOS.”. Universdad de Valencia, 2006.

[29] J. Alcaide, R. Ignacio, R. R. Jim, J. Carlos, and F. Jim, “Desarrollo de un entorno hardware para la ejecución de aplicaciones de propósito general sobre FPGA’s,” p. 151, 2009.

[30] F. Javier, A. Mauleón, S. Somovilla, and D. Torre, “Diseño y fabricación de una placa didáctica basada en FPGA,”Facultad Informática, Universidad Complutense de Madrid, 2006.

[31] M. Drutarovský, J. Šaliga, L. Michaeli, and I. Hroncová, “Remote Laboratory for Fpga Based Reconfigurable,” pp. 54–58, 2009.

[32] L. A. RODRIGUEZ E., “Desarrollo de un sistema de monitoreo y control electrónico remoto vía internet basado en lenguaje Java.,” Universidad del Valle, 2005.

[33] M. J. Callaghan, J. Harkin, T. M. McGinnity, and L. P. Maguire, “Client-server architecture for remote experimentation for embedded systems,” *Int. J. Online Eng.*, vol. 2, no. 4 Spec Iss, p. 6 pp. –, 2006.

[34] F. Slomka, M. Dorfel, R. Munzenberger, and R. Hofmann, “Hardware/software codesign and rapid prototyping of embedded systems,” *IEEE Des. Test Comput.*, 2000.

[35] ALTERA CORPORATION., *Quartus II Handbook*. 2009.

[36] Universidad de Deusto, Proyecto “WebLab-DEUSTO.”, 2005.

[37] OpenLabs Electronics Laboratory, User Manual “VISIR – Virtual Instrumental Systems in Reality.”, Electrical & Computer Engineering Department-DIEEC, .

[38] C. Saygin and K. Firat, “A Web-based programmable logic controller laboratory for manufacturing engineering education,” *Int. J. Adv. Manuf. Technol.*, vol. 24, 2004.

[39] D. KARADIMAS and K. y EFSTATHIOU, “Design, Implementation and Evaluation of a Remote Laboratory System for Electrical Engineering Courses,” *Proceedings of the Sixth International Conference on Advanced Learning Technologies (ICALT’06)*. IEEE Computer Society, 2006.

[40] R. Hashemian and J. Riddley, “FPGA e-Lab, a Technique to Remote Access a Laboratory to Design and Test,” *2007 IEEE Int. Conf. Microelectron. Syst. Educ.*, pp. 139–140, Jun. 2007.

[41] L. S. Indrusiak, M. Glesner, and R. Reis, “On Evolution of Remote Laboratory for Prototyping Digital Electronic System,” *IEEE Trans. Ind. Electron.*, vol. 54, 2007.

[42] N. FUJII and N. Koike, “Work in Progress - A New Time-sharing Remote Laboratory e-Learning System for Hardware Design and Experiment of Digital Circuits,” in *35th ASEE/IEEE Frontiers in Education Conference*, 2005.

[43] N. FUJII and N. Koike, “New Virtual Remote Laboratory Environments for Logic Circuit Design Realizing an efficient sharing of Test Equipments and Concurrent User Support,” in *Hosei University, Faculty of Computer and Information Sciences*, 2006

[44] D. I. R. U. Dugzduh, P. Zphophgdq, L. W. F. Xre, E. K. Zdhohophgdq, and J. Frp, “FPGA remote lab for hardware elearning courses,” *E-learning*, vol. 0, pp. 106–109, 2008.

[45] E. C. Ferreira, “Arquitetura para laboratório de acesso remoto com aplicação no ensino de engenharia eletrônica,” pp. 33–38, 2005.

[46] “Ferramentas Aplicadas no Desenvolvimento de Laboratório e/ou Presencial no Ensino de Engenharia Electrônica.,” Campinas, 2006.

[47] V. R. Moreira, F. A. C. M. Cardoso, and D. S. Arantes, “Plataforma reconfigurável para ensino e pesquisa em laboratório de sistemas digitais a distância.”. Anais do SBIE, 2008.

[48] R. J. Costa, G. R. Alves, M. Zenha-Rela, M. Zenha-Rela, R. Poley, and C. Wishart, “FPGA-based weblab infrastructures guidelines and a prototype implementation example,” *2009 3rd IEEE Int. Conf. E-Learning Ind. Electron.*, pp. 57–63, Nov. 2009.

[49] R. Costa, G. Alves, M. Zenha-rela, I. C. Laboris, and F. Cisuc, “Reconfigurable weblabs based on the IEEE1451 Std .,” pp. 1359–1366, 2010.

[50] M. Tawfik, C. Salzmann, D. Gillet, D. Lowe, H. Saliah-Hassane, E. Sancristobal, and M. Castro, “Laboratory as a service (LaaS): A novel paradigm for developing and implementing modular remote laboratories,” *Int. J. Online Eng.*, vol. 10, no. 4, pp. 13–21, 2014.

[51] S. Cueva, E. Pacheco, G. Rodríguez, and A. Santos, “Tecnologías de Información y Comunicación (TIC´s) en la Educación Superior.”

[52] G. Díaz, D. Gutiérrez, and E. Vargas, “TICs en la Educación.” 2012.

[53] A. López and E. Velásquez, “Una Mirada Crítica Al Papel De Las Tic En La Educación Superior En Colombia,” *Universidad del Magdalena, Universidad del Tolima*, 2012.

[54] F. Piedrahita, “EL PORQUÉ DE LAS TIC EN EDUCACIÓN.” [Online]. Available: http://www.eduteka.org/PorQueTIC.php.

[55] T. O’Reilly and J. Battelle, “What is Web 2.0?. Design Patterns and Business Models for the Next Generation of Software,” 2009. [Online]. Available: http://oreilly.com/web2/archive/what-is-web-20.html.

[56] M. Sabin and J. Leone, “IT Education 2.0,” in *Proceedings of the 10th ACM conference on SIG-information technology education*, 2009.

[57] Educastur, “Web 2.0 y educación.” [Online]. Available: http://blog.educastur.es/files/2007/06/web2_0v02.pdf.

[58] E. Tello Leal, C. M. SOsa Reyna, M. Lucio Castillo, and M. M. Flores Morelos, “Análisis de los servicios de la tecnología Web 2.0 aplicados a la educación,” *No Solo Usabilidad*, 2010.

[59] I. Peña and C. Corcoles, “Web 2.0 y difusión de la investigación: reseña del seminario,” *Revista de Internet, derecho y Política N°3*. [Online]. Available: http://www.uoc.edu/idp/3/dt/esp/pena_corcoles.pdf.

[60] E. Dans, “Educacion on-line. Plataformas educativas y el dilema de la apertura,” *Revista de Universidad y Sociedad del Conocimiento*. [Online]. Available: http://digithum.uoc.edu/ojs/index.php/rusc/article/view/26/21.

[61] C. Ullrich, K. Borau, H. Luo, X. Tan, L. Shen, and R. Shen, “Why Web 2.0 is Good for Learning and for Research: Principles and Prototypes,” in *Proceedings of the 17th International World Wide Web Conference*.

[62] P. Anderson, “What is Web 2.0: Ideas, technologies and implications for education. JISC Technology and Standards Watch,” *Technology & Standars Watch*, 2007. [Online]. Available: http://www.jisc.ac.uk/media/documents/techwatch/tsw0701b.pdf.

[63] N. Piedra, “Recursos y prácticas educativas abiertas utilizando herramientas y servicios basados en el software social,” in *II CONGRESO CREAD ANDES y II ENCUENTRO VIRTUAL EDUCA*.

[64] J. de Haro, “Redes Sociales en Educación.” [Online]. Available: http://danzanet.org/data/2011/10/21/35/file/1319411880redes_sociales_educacion.pdf.

[65] N. A. Vences, “Las redes sociales como herramienta educativa en el ámbito universitario,” *Rev. Electrónica ADA-Madrid*, vol. 3, no. 3, 2009.

[66] S. Visagie and C. de Villiers, “The consideration of Facebook as an academic tool by ICT lecturers across five countries,” in *Proceedings of the 40th annual conference of the Southern African Computer Lecturers’ Association (SACLA)*, 2010.

[67] Moodle.org, “Usos didácticos del Wiki.” [Online]. Available: http://docs.moodle.org/all/es/Usos_did%C3%A1cticos_del_Wiki. [Accessed: 11-Nov-2012].

[68] N. Saeed and Y. Yang, “Incorporating blogs, social bookmarks, and podcasts into unit teaching,” in *Proceedings Tenth Australasian Computing Education Conference (ACE 2008), Wollongong, NSW, Australia*, 2008.

[69] J. López, “Uso educativo de los blogs,” *Eduteka.org*. [Online]. Available: http://www.eduteka.org/BlogsEducacion.php. [Accessed: 15-Jan-2013].

[70] P. Anderson, “Entienda la Web 2.0 y sus Principales Servicios,” *Eduteka.org*. [Online]. Available: http://www.eduteka.org/Web20Intro.php. [Accessed: 15-Jan-2013].

[71] U. N. del Nordeste, “Informática. Conceptos Fundamentales.” Universidad Nacional del Nordeste, Corrientes- Argentina, Corrientes, Argentina.

[72] CNRTL, “Informatique - Etymologie,” *Centre National de Ressources Textuelles et Lexicales*, 2012. [Online]. Available: http://www.cnrtl.fr/etymologie/informatique. [Accessed: 25-Jan-2014].

[73] M. Fourman, “Informatics,” 2002.

[74] RAE, “Real Academia Española.” [Online]. Available: www.rae.es. [Accessed: 11-Nov-2014].

[75] O. Faust, R. Acharya U, B. H. C. Sputh, and L. C. Min, *Systems engineering principles for the design of biomedical signal processing systems*, vol. 102, no. 3. DAU/DSMC, 2011.

[76] J. D. Aguilar, R. A. Valdés, and D. A. Labaut, “Propuesta del sitio Web para la Gestión de Contenidos sobre Nanociencias y Nanotecnologías del Centro de Estudios Avanzados de Cuba,” *Ciencias la Inf.*, vol. 42, no. 2, pp. 61–71, 2011.

[77] J. M. Castro, “Gestión de Contenidos Web Basada en Software Libre Content,” *DYNA Ing. e Ind.*, vol. 83, no. 4, pp. 207–213, 2008.

[78] J. Tramullas, “Drupal: Fundamentos Técnicos,” 2010. [Online]. Available: http://isuu.com/tramullas/doc/drupalbibliotecas. [Accessed: 11-Nov-2012].

[79] Dycky, “20 Promising Open Source PHP Content Management Systems (CMS),” *Web Design Booth*, 2009. [Online]. Available: http://www.webdesignbooth.com/20-promising-open-source-php-content-management-systemscms/. [Accessed: 07-Jul-2012].

[80] Open Source Matters, “Joomla!,” *Joomla.org*. [Online]. Available: www.joomla.org. [Accessed: 07-Jul-2012].

[81] Astrolabio, “Joomla: razones para usarlo en su sitio web,” *Astrolabio Agencia Digital*, 2008. [Online]. Available: http://www.astrolabio.com.co/diseno-web/13-joomla-razones-para-usarlo-en-su-sitio-web.html. [Accessed: 07-Jul-2012].

[82] A. Villamizar, “Ventajas de la Web 2.0.” Universidad Autónoma de Bucaramanga, 2009.

[83] E. Velázquez, “eLearning: conociendo un Learning Management System (LMS),” *Pymes y Autónomos*, 2009. [Online]. Available: http://www.pymesyautonomos.com/tecnologia/e-learning-conociendo-un-learning-management-system-lms. [Accessed: 07-Jul-2012].

[84] Moodle.org, “The Moodle Project,” *Moodle.org*. [Online]. Available: https://moodle.org/. [Accessed: 07-Jul-2012].

[85] J. Gutiérrez, “Applet. Applets firmados. Encriptacion.” Universitat de València, 2005.

[86] Netcraft, “NETCRAFT,” *April 2014 Web Server Survey*, 2014. [Online]. Available: http://news.netcraft.com/archives/2014/04/02/april-2014-web-server-survey.html. [Accessed: 05-May-2014].

[87] H. Graf, “Manual Joomla! 2.5 Guía para principiantes,” *Joomla.org*. [Online]. Available: http://ayudajoomla.com/manual-joomla-2.5.html. [Accessed: 11-Nov-2012].

[88] B. Marcano, “El Facebook como herramienta 2.0 para el aprendizaje,” 2010. [Online]. Available: http://beatrizmarcano.blogspot.com/2010/06/el-facebook-como-herramienta-20-para-el.html. [Accessed: 11-Nov-2012].

[89] Microchip, “Generic Driver Demo,” *Microchip Technology Inc.* [Online]. Available: www.microchip.com. [Accessed: 11-Nov-2012].

[90] TRAC, “Biblioteca ‘libusb,’” *Integrated SCM & Project Management*, 2012. [Online]. Available: http://www.libusb.org/. [Accessed: 11-Nov-2012].

[91] Cypress, “Cypress EZ-USB FX2LP™,” *Cypress Perform*. [Online]. Available: www.cypress.com. [Accessed: 30-Nov-2012].

[92] Libusb, “Java libusb/ libusb-win32 wrapper,” *Computer Science Laboratory - Sourceforge.net*. [Online]. Available: http://libusbjava.sourceforge.net/wp/?page_id=4. [Accessed: 30-Nov-2012].

[93] Libusb, “libusb-1.0 API Reference,” *Sourceforge.net*. [Online]. Available: http://libusb.sourceforge.net/api-1.0/. [Accessed: 30-Nov-2012].

[94] Xilinx, “Spartan-6 Family Overview Summary of Spartan-6 FPGA Features,” pp. 1–11, 2011.

[95] R. Nieto, Á. Bernal, and A. Vera, “Sistema Embebido para el Entrenamiento de Habilidades en el Diseño Electrónico Digital Usando TICs,” Cali, 2012.

[96] Xilinx, “Partia Reconfiguration User Guide,” 2010.

[97] K. B. Sram, H. Pwm, and A. Analog, “16-bit Microcontrollers and Digital Signal Controllers (up to 512 KB Flash,” no. mm, 2012.

[98] W. Stalling, *Operating systems*, Pearson. Seventh Edition, 2011.

[99] Xilinx, “Spartan-6 FPGA PCB Design and Pin Planning Guide,” pp. 1–72, 2010.

[100] L. T. Corporation, “LT3501 - Monolithic Dual Tracking 3A Step-Down Switching Regulator,” pp. 1–28.

[101] L. T. Corporation, “LTC3546 - Dual Synchronous, 3A/1A or 2A/2A Configurable Step-Down DC/DC Regulator,” pp. 1–28.

[102] Association Connecting Electronics Industries, “IPC-2222,” no. February. 1998.

[103] A. Vera, J. Quiñones, F. Enriquez, S. Arenas, and A. Bernal, “Remote Laboratory for e-Learning of Digital Systems Design.” 2012.

[104] Microchip, “Generic Driver Demo.” [Online]. Available: http://www.microchip.com.

[105] J. E. QUIÑONES HEREDIA, “DESARROLLO DE UNA APLICACIÓN CLIENTE/SERVIDOR PARA LA EXPERIMENTACIÓN REMOTA EN SISTEMAS DIGITALES,” 2013.

[106] J. E. Quiñones, E. F. Enriquez, D. F. Garcia, A. Vera, and A. Bernal, “Integración de un Hardware Configurable y una Aplicación basada en CMS-LMS para e-learning de,” *CISCI 2013*, 2013.

[107] Tektronix, *Logic Analyzer Fundamentals*. Tektronix, 2010.

[108] L. Frédéric, “The value of logic analyzers for real-time embedded software debug,” *Byte Paradigm*, Nivelles - Belgium, pp. 1–7, 2012.

[109] C. F. Coombs, *Electronic Instrument Handbook*, Third Edit. McGraw-Hill, 1999.

[110] Agilent Technologies Inc., “Feeling Comfortable with Logic Analyzers. Application Note 1337,” *Agilent Technologies, Inc.*, pp. 1– 23, 2006.

[111] K. Veenstra, K. Rangasayee, and A. L. Herrmann, “Enhanced Embedded Logic Analyzer,” US 6704889 B2, 2004.

[112] A. L. Herrmann and G. P. Nugent, “Embedded Logic Analyzer for a Programmable Logic Device,” US 6389558 B1, 2002.

[113] M. W. William D. Corti, Robert Kenny, Jr., Joseph O. Marsh, Steven C. Parker, Frank X. Scanzano, “On-chip logic analyzer,” US 6834360 B2, 2004.

[114] J. L. Creigh, “Programmable Embedded Logic Analyzer in an Integrated Circuit,” US 7350121 B2, 2008.

[115] J. A. Correa B. and A. A. Ramírez O., “Diseño e Implementación de un Analizador Lógico Multicanal Usando Hardware Reprogramable,” Universidad del Valle, 2004.

[116] J. Pájaro, C. Torres, R. Nieto, E. Duque, and Á. Bernal, “Laboratorio Basado en Internet para la Enseñanza del Prototipado Rápido de Sistemas Digitales,” in *X Workshop Iberchip IWS’2004*, 2004, p. 5c.

[117] R. Ramirez, M. J. Wiznerowicz, S. T. Baartmans, and J. G. Sandri, “Providing an On-Die logic analyzer (ODLA) having reduced communications,” US 2012/0131404 A1, 2012.

[118] S. M. Nowick and M. Singh, “High-performance asynchronous pipelines: An overview,” *IEEE Des. Test Comput.*, vol. 28, no. 5, pp. 8–22, 2011.

[119] Xilinx Inc., “Finite State Machines,” in *Lab Workbook*, no. Summer, 2013, pp. 1–5.

[120] M. McNamara, “IEEE Verilog Standardization Group,” 2012. [Online]. Available: http://www.verilog.com/IEEEVerilog.html. [Accessed: 01-Dec-2014].

[121] J. Hom, “The Usability Methods Toolbox Handbook.” University of Technology, Eindhoven, Nederlands, pp. 1–72, 1998.

[122] X. Ferré, “Usabilidad: Software pensado para los usuarios.” [Online]. Available: http://noticias.universia.es/ciencia-nn-tt/noticia/2006/11/14/594994/usabilidad-software-pensado-usuarios.html. [Accessed: 15-Sep-2012].

[123] SysNucleus, “USBTrace: USB Protocol Analyzer Software for Windows.” [Online]. Available: http://sysnucleus.com/. [Accessed: 01-Dec-2014].

[124] P. Salinas, “Unified Modeling Language,” *Universidad de Chile, Facultad de Ciencias Físicas y Matemáticas. Dep. de Ciencias de la computación*. [Online]. Available: http://users.dcc.uchile.cl/~psalinas/uml/modelo.html#relacion. [Accessed: 11-Dec-2012].

[125] M. Lucas, “Checklist para Evaluar la Usabilidad de un Site.” [Online]. Available: http://miguellucas.com/2011/01/cuestionario-usabilidad/. [Accessed: 11-Dec-2012].

ANEXO A Soporte de *Facebook* para Aplicaciones

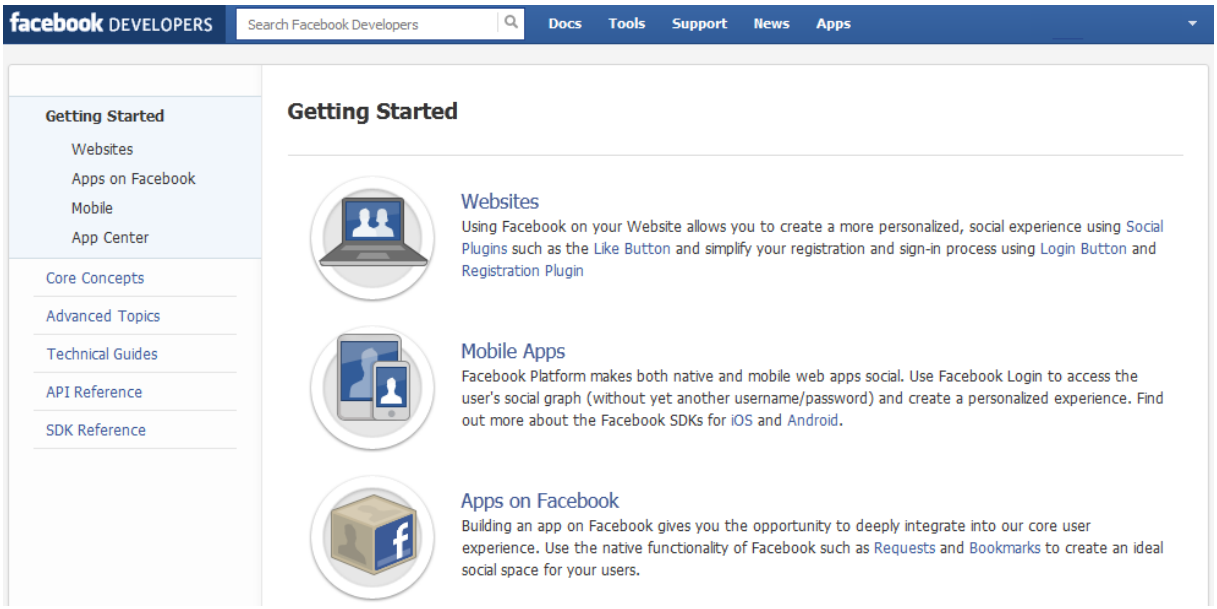
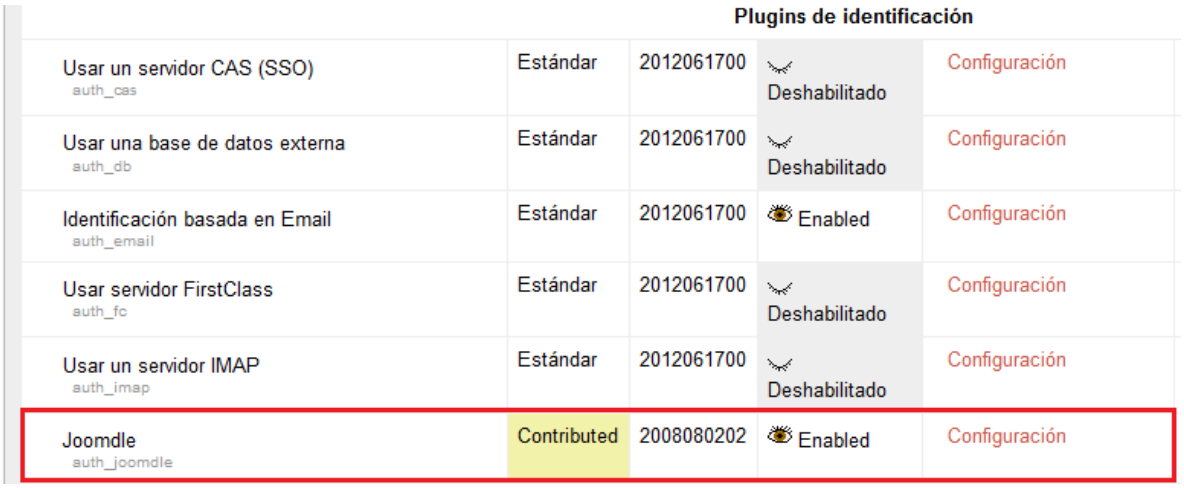


Figura A1 Plataformas para las que Facebook entrega soporte para aplicaciones.

ANEXO B Configuración del Complemento *Joomdle* desde *Joomla*

Figura B1 Configuración del complemento *Joomdle* desde *Joomla*.

ANEXO C Configuración del Complemento *Joomdle* desde *Moodle*



Plugins de identificación

Usar un servidor CAS (SSO) auth_cas	Estándar	2012061700	Deshabilitado	Configuración
Usar una base de datos externa auth_db	Estándar	2012061700	Deshabilitado	Configuración
Identificación basada en Email auth_email	Estándar	2012061700	Enabled	Configuración
Usar servidor FirstClass auth_fc	Estándar	2012061700	Deshabilitado	Configuración
Usar un servidor IMAP auth_imap	Estándar	2012061700	Deshabilitado	Configuración
Joomdle auth_joomdle	Contributed	2008080202	Enabled	Configuración

Figura C1 Habilitación y configuración del complemento *Joomdle* desde *Moodle*.

ANEXO D Integración LMS – CMS. Errores de Visualización

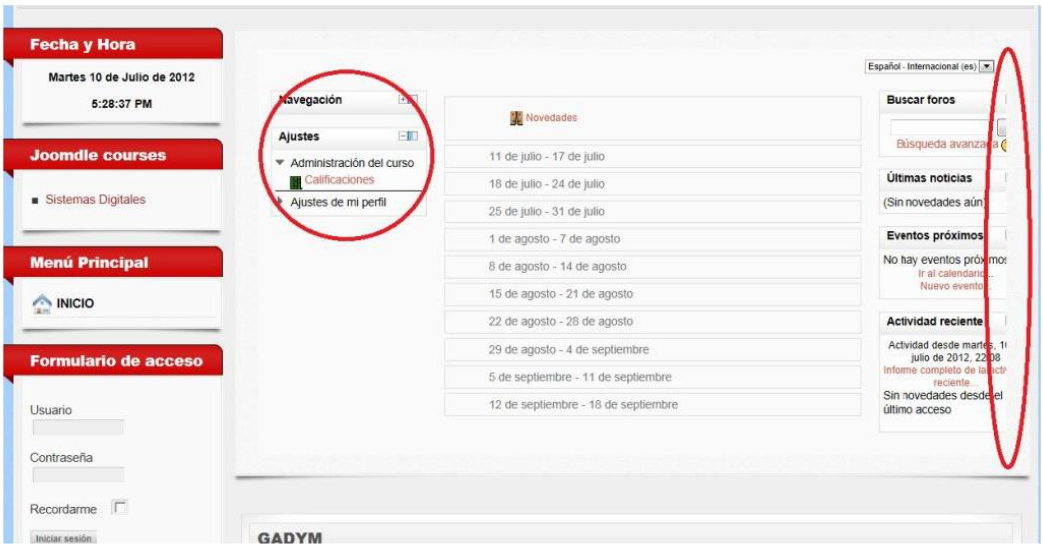


Figura D1 Errores de visualización presentes al realizar la integración entre *Joomla* y *Moodle*.

Figura D2 Errores de visualización corregidos, integración exitosa.

ANEXO E Diagramas de Clases

Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de contenido [124].

Un diagrama de clases está compuesto por los siguientes elementos:

- Clases: atributos, métodos y visibilidad.
- Relaciones: Herencia, Composición, Agregación, Asociación y uso.

A continuación se describen de forma general los elementos del diagrama de clases.

- ***Clase***. El elemento básico de la programación orientada a objetos en *Java* es la clase. Una clase define la forma y comportamiento de un objeto. Esta constituye la unidad básica que encapsula toda la información de un objeto (un objeto es una instancia de una clase). A través de éstas se puede modelar el entorno en estudio (una Casa, un Auto, una Cuenta Corriente, etc.).
  En UML, una clase es representada por un rectángulo que posee tres divisiones:

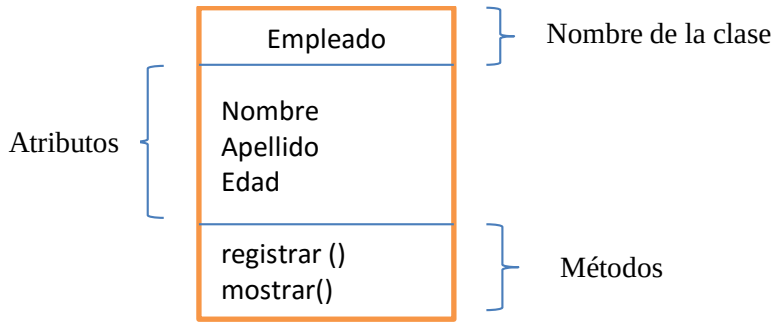


Figura E1 Elementos de una Clase

- ***Atributos***. Los atributos o características de una Clase definen el grado de comunicación y visibilidad de ellos con el entorno. Estos pueden ser de tres tipos:
  - ***public (+):*** Indica que el atributo será visible tanto dentro como fuera de la clase; es decir, es accesible desde todos lados.
  - ***private (-):*** Indica que el atributo sólo será accesible desde dentro de la clase (sólo sus métodos pueden acceder a él).
  - ***protected (#):*** Indica que el atributo no será accesible desde fuera de la clase, pero sí podrá ser accedido por métodos de la clase, además de las subclases que se deriven (ver herencia).
- ***Métodos***. Los métodos u operaciones de una clase son la forma como ésta interactúa con su entorno. Estos pueden tener características de *public*, *private* o *protected*.
- ***Relaciones entre Clases.*** Dos o más Clases pueden interrelacionarse, cada una con características y objetivos diferentes. En UML, la *cardinalidad* de las relaciones indica el grado y nivel de dependencia, se anotan en cada extremo de la relación y éstas pueden ser:
  - *uno o muchos*: 1..* (1..n).
  - *cero o muchos*: 0..* (0..n).
  - *número fijo*: m (m denota el número).

  Las relaciones entre Clases se establecen a continuación:
  - ***Herencia (Especialización/Generalización)***. Indica que una subclase hereda los métodos y atributos especificados por una Súper-Clase, por ende la Subclase además de poseer sus

propios métodos y atributos, poseerá las características y atributos visibles de la Super-Clase (*public* y *protected*), ejemplo:

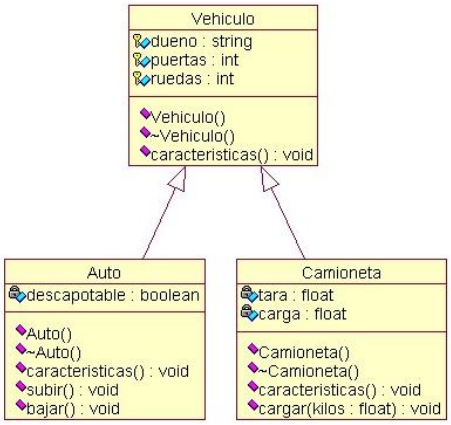


Figura E2 Relación de Herencia

En la figura se especifica que *Auto* y *Camión* heredan de *Vehículo*; es decir, *Auto* posee las características de *Vehículo* (*Precio*, *VelMax*, etc.); además, posee algo particular que es *Descapotable*. *Camión* también hereda las características de *Vehículo* (*Precio*, *VelMax*, etc.) pero posee como particularidad propia *Acoplado*, *Tara* y *Carga*.

Cabe destacar que fuera de este entorno, lo único "visible" es el método *Características* aplicable a instancias de *Vehículo*, *Auto* y *Camión*, pues tiene definición pública; en cambio, atributos como *Descapotable* no son visibles por ser privados.

- ***Asociación***. La relación entre clases conocida como Asociación, permite asociar objetos que colaboran entre sí. Cabe destacar que no es una relación fuerte; es decir, el tiempo de vida de un objeto no depende del otro. Ejemplo:

Figura E3 Relación de Asociación

Un *Cliente* puede tener asociadas muchas *órdenes de Compra*; en cambio, una *orden de compra* solo puede tener asociado un *Cliente*.

- ***Dependencia o Instanciación (Uso)***. Representa un tipo de relación muy particular, en la que una clase es instanciada (su instanciación es dependiente de otro objeto/clase) y se denota por una flecha punteada. El uso más particular de este tipo de relación es para denotar la dependencia que tiene una clase de otra; por ejemplo, una aplicación gráfica que instancia una *ventana* (la creación del Objeto *Ventana* está condicionado a la instanciación proveniente desde el objeto *Aplicación*):

Figura E4 Relación de Dependencia

Cabe destacar que el objeto creado (en este caso la *Ventana* gráfica) no se almacena dentro del objeto que lo crea (en este caso la *Aplicación*).

ANEXO F Diseño de Encuesta Online de Usabilidad

De acuerdo con la encuesta planteada en [125] para evaluar la usabilidad de un sitio Web, se aplican preguntas considerando los siguientes aspectos:

- **Legibilidad**
  - **¿Son el tamaño de la fuente e interlineado adecuados?** Para realizar esta valoración hay que estudiar si el tamaño efectivo de los caracteres en los monitores de resolución y tamaño más habituales es lo suficientemente grande como para poder realizar una lectura cómoda.
  - **¿Contrasta el color de la fuente suficientemente con el fondo?** La legibilidad aumenta cuando aumenta el contraste entre la fuente y el fondo. También suele ser mejor cuando jugamos con fondos claros y fuentes oscuras que al contrario.
- **Identidad**
  - **¿Ocupa el logo una posición visible y siempre la misma a lo largo de las diferentes secciones del sitio?** Los experimentos de usabilidad confirman que la expectativa del usuario promedio es encontrarse el logo en la parte superior izquierda. Por otro lado, el logo debe ser un elemento de estructura y encontrarse siempre en la misma posición.
  - **¿Cuenta con un título significativo y clarificador?** Es recomendable acompañar siempre el logo con un título que ayude al usuario a identificar cuál es el propósito general del *site* en que está aterrizando. Al igual que el logo, la expectativa es encontrarlo siempre en el mismo lugar.
  - **¿Es fácil ubicar el acceso a la información de contacto?** Una de las piezas de información de visita más frecuente es la de contacto. Por lo tanto, una interfaz usable debe mantener en una posición visible el acceso a dicha información.
  - **¿Es fácil ubicar el acceso a la información de acerca de?** De una manera similar a como sucede con la información de contacto, la información que presenta a la entidad y/o personas que se encuentran detrás de un *site*, debe estar en una posición prominente y visible.
- **Accesibilidad**
  - **¿Se podría usar el sitio sin imágenes?** Un sitio accesible debe hacer posible la navegación a lo largo de sus diferentes secciones aun cuando no fuera posible ver o cargar las imágenes. Para poder evaluar este elemento, puede intentar hacerse una navegación con la carga de imágenes deshabilitada en el navegador.
  - **¿Se podría usar el sitio con un navegador sin intérprete de JavaScript?** Situación similar a la anterior, solo que en este caso, la función no disponible es la de ejecutar código *Javascript*. Para evaluarlo, se puede deshabilitar expresamente el soporte *JavaScript* del navegador.
  - **¿Se puede usar el sitio sin soporte a Flash?** Muchos navegadores no cuentan con soporte a *Flash*, por lo que un sitio que requiera necesariamente esta funcionalidad para funcionar cuenta con un importante problema de usabilidad. Para estudiar esta situación en *Firefox*, *Herramientas > Complementos* y allí, deshabilitar la extensión *Shockwave Flash.*
  - **¿Cuenta con una página 404 con instrucciones para explicar y salvar la situación?** Ante páginas que no existen, muchos sitio devuelven el típico mensaje del navegador con información muy escueta y poco explicativa de la situación. Incorporar una página 404 que mantenga la estructura de una página típica, que explique la situación y que ofrezca soluciones alternativas (utilizar el buscador, navegar por el archivo, etc) permite mejorar las cotas de usabilidad.
- **Navegabilidad**

- **¿El botón de atrás siempre lleva a la página anterior?** Numerosos estudios de usabilidad demuestran que una de las funcionalidades más utilizadas en la navegación web es el botón de atrás del navegador. Y la expectativa de funcionamiento es que el navegador nos devuelva a la página que ha visitado justo antes que la actual. El caso es que en determinadas programaciones de página (generalmente asociadas a ciertos usos de *JavaScript*) el botón de atrás no lleva a la página anterior, suponiendo un importante problema de usabilidad.
- **¿El menú principal de navegación el claramente visible, ocupa siempre la misma posición y tiene siempre los elementos principales en el mismo lugar?** El menú principal es uno de los principales elementos de navegación con que cuenta un sitio web, por lo que es de vital importancia que este sea de fácil uso y su comportamiento sea siempre el mismo.
- **¿Son los nombres de las secciones del menú lo suficientemente significativos?** Para que los menús sean navegables es necesario que los nombres de las secciones evoquen de manera directa para su público objetivo el tipo de contenido que encontrarán detrás de ellas.
- **¿El árbol de navegación del menú tiene proporciones razonables para su legibilidad?** O dicho de otro modo, ¿la relación entre el ancho y el alto de las diferentes secciones del menú guarda las proporciones adecuadas para poder ser consumidos de manera fácil de un vistazo?
- **¿Son los enlaces fácilmente identificables?** Los enlaces son otro de los elementos clave en la usabilidad de un sitio. Estos han de ser fácilmente reconocibles, diferentes del texto base y deben utilizar una representación que sea constante a lo largo de todo el sitio para poder tener un nivel de uso razonable.

- **Contenido/Formato**
  - **¿Los títulos de sus páginas son claros y descriptivos?** Se trata de una variable fundamental para facilitar el uso/consumo de los contenidos y facilitar la ubicación en el sitio.
  - **¿Los mensajes fundamentales están accesibles sin hacer *scroll*?** Es recomendable que la oferta de los contenidos o propuestas de acción fundamentales estén disponibles a primera vista sin necesidad de hacer *scroll* para los monitores de resolución y tamaño más habituales.
  - **¿Es fácilmente escaneable?** Puesto que uno de los patrones clave de lectura en web es el escaneo, las páginas usables deben estar construidas y formateadas de modo que se facilite su escaneo.
  - **¿Es la representación de los elementos como listas, citas, títulos, *etc* constante a lo largo de todo el sitio?** Es fundamental para la usabilidad que el mismo tipo de elementos gocen de un aspecto gráfico constante a lo largo de todo el sitio.
  - **¿Son los formularios fácilmente reconocibles?** Para facilitar su uso, es necesario que los formularios parezcan formularios. Hay ocasiones en que determinadas creatividades encaminadas a conseguir una mejor integración gráfica dificultan el reconocimiento de elementos como cajas de texto, botones etc.
  - **¿La solicitud de citas es fácil de entender?** Es necesario que la solicitud de citas se fácil de entender para los usuarios para incrementar la usabilidad.
  - **¿Todos los procesos del laboratorio están claramente especificados?** Debido a los diferentes pasos por los que debe pasar un estudiante, se requiere que cada uno se encuentre claramente especificado.
- **Socialización**

- **¿Cuenta con elementos que facilitan la socialización entre usuarios?** Uno de los usos de frecuencia creciente en la web social consiste en la compartición en redes sociales de los contenidos más interesantes. Facilitar esta función permite elevar las cotas de usabilidad.

- **Llamada a la acción**

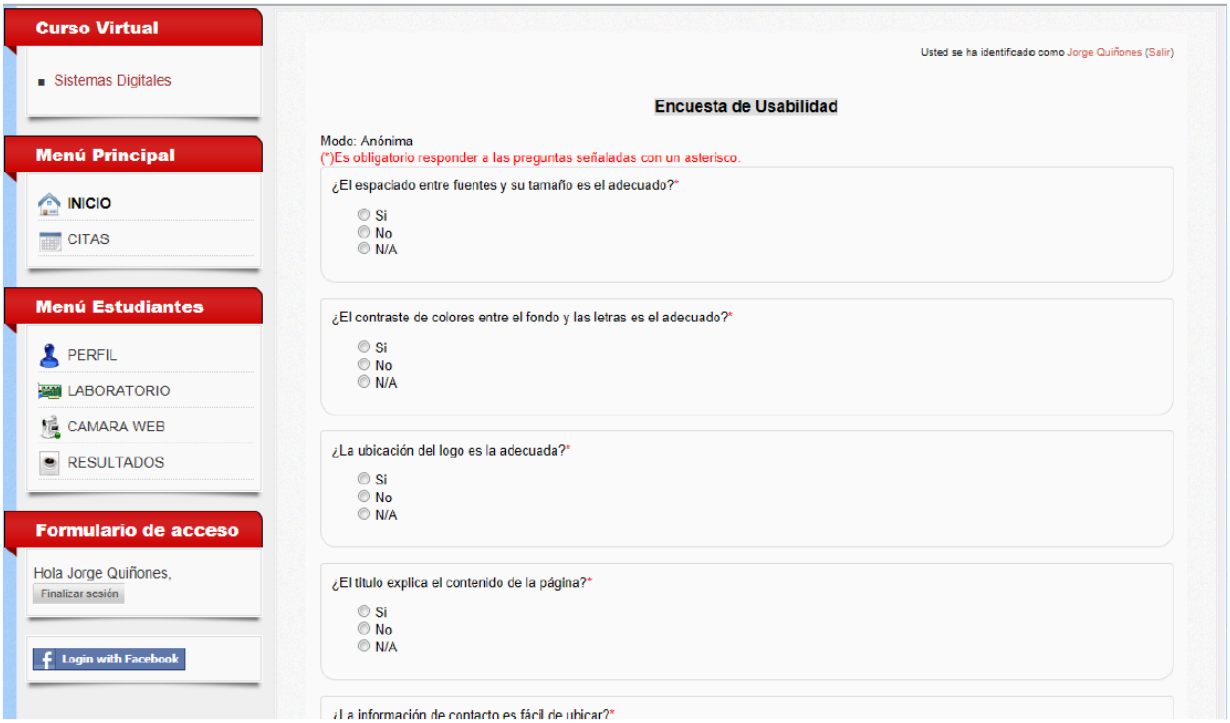


Figura F1 Encuesta de usabilidad en línea