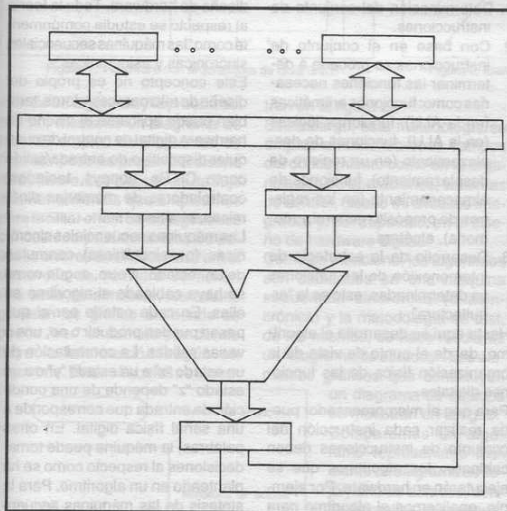


Conceptos fundamentales en el diseño de hardware digital



□ Luis Alberto Gómez
Ingeniero Electricista
Profesor Auxiliar
Departamento de Electricidad
Universidad del Valle

RESUMEN

Existen algunos conceptos generales importantes en el diseño del hardware digital. Estos son: concepción y empleo de algoritmos, verificación del algoritmo mediante un diagrama de flujo, secuencialidad y paralelismo en la realización de eventos digitales y estudio de los retardos de tiempo involucrados en cada evento que se rea-

liza. En este artículo se hace un análisis sobre cada uno de estos aspectos aplicados al diseño de hardware digital y se analizan ejemplos en los que se resumen los conceptos.

ABSTRACT

There is some important general concepts concerning with the digital hardware design. They are: conception and using of algorithms, testing the algorithms by the use of a flow chart, the sequences employed to make on every digital event and the parallelism concepts, finally the analysis of the time delay that performs every digital event. This paper shall make an analysis on every issue concerning to the digital hardware design. We will go into examples as the concepts are exposed.

1. INTRODUCCION

Los ingenieros que trabajan en el área de los sistemas digitales a menudo se encuentran con la necesidad de diseñar un hardware digital, el cual realiza una función específica dentro de un sistema mayor. Muchos de ellos no asistieron a un curso de diseño de hardware lo que les hace difícil allegar una solución fiable para el problema planteado. En este artículo se hace énfasis en los aspectos más

importantes para el diseño y puesta en funcionamiento de un hardware digital, asumiendo que el lector ha tomado los cursos básicos en lógica digital.

El primer concepto que se analiza es el uso de algoritmos para trazar una estrategia de solución del problema planteado, luego se estudia un método gráfico de verificación de la validez del algoritmo. A estos conceptos básicos se añaden los de la secuencialidad, paralelismo y el estudio de los retardos de tiempo asociados a cada función lógica. Los conceptos expuestos se aplican a ejemplos para hacer claridad sobre los mismos.

2. CONCEPCION Y EMPLEO DE ALGORITMOS

Estamos acostumbrados a escuchar la palabra "algoritmo" y de inmediato asociarla con la realización de software; nada más errado. El concepto de algoritmo está asociado al hombre mismo en todos los aspectos de su vida, casi cualquier evento que queramos realizar implica la concepción anticipada de un algoritmo. Por ejemplo, cada mañana antes de ir a trabajar, muchos de ustedes piensan en cómo organizar el día de forma que éste sea bien aprovechado. Se piensa en detalles como: ¿a dónde debo ir primero en la mañana?, ¿qué ruta de bus debo tomar para ir a dicho lugar?, ¿será conveniente dejar el evento para la tarde?, ¿qué haré por la noche? etcétera. De igual forma, una persona que se muda de casa pensará en la forma más conveniente de acomodar sus pertenencias para aprovechar, de la mejor manera, el espacio de su nueva casa.

Como se ve el concepto de organizar eventos o cosas (algoritmo) no es propio del desarrollo de software, es un concepto más universal. Para el diseño de hardware se debe tener en cuenta este concepto desde dos puntos de vista: primero la organización física de las funciones de hardware, de forma tal que

cumplan con el objetivo del sistema y por último la realización en hardware de los eventos que deben realizar las funciones que constituyen el sistema. Aunque suene extraño, un algoritmo concebido para diseño de un hardware puede ser cableado y construido con circuitería digital.

Por ejemplo, suponga que se necesita diseñar un microprocesador de propósito general, éste es el problema planteado al diseñador. Desde el punto de vista de la organización de las funciones necesarias se podría seguir el siguiente algoritmo:

1. Determinación del conjunto de instrucciones.
2. Con base en el conjunto de instrucciones se procede a determinar las funciones necesarias como: funciones aritméticas (en la ALU), funciones lógicas (en la ALU), funciones de desplazamiento (en un registro de desplazamiento), funciones de almacenamiento (en los registros de propósito general y memoria), etcétera
3. Desarrollo de la estrategia de interconexión de las funciones ya determinadas, esto es la "arquitectura".

Hasta aquí se desarrolla el algoritmo, desde el punto de vista de la organización física de las funciones digitales.

Para que el microprocesador pueda realizar cada instrucción del conjunto de instrucciones deben cablearse los algoritmos que se ejecutarán en hardware. Por ejemplo, analicemos el algoritmo para realizar la instrucción suma de dos valores. Los valores fueron almacenados con anterioridad en los registros llamados A y B, además la ALU cuenta con dos registros temporales llamados **temp1** y **temp2** y un registro de salida llamado **temp3** (ver figura 1); suponemos que el resultado debe almacenarse de nuevo en el registro A. El algoritmo que debe cablearse para la realización de la instrucción suma podría ser:

1. Leer el contenido del registro A y escribirlo sobre el registro **temp1**.
2. Leer el contenido del registro B y escribirlo sobre el registro **temp2**.
3. Ordenar a la ALU la realización de la suma con los operandos almacenados en **temp1** y **temp2**. Almacenar el resultado en **temp3**.
4. Leer el contenido del registro **temp3** y escribirlo sobre el registro A.

El concepto de "algoritmo cableado" es uno de los más importantes en el diseño de hardware. Toda la teoría al respecto se estudia comúnmente como "las máquinas secuenciales sincrónicas y asincrónicas".

Este concepto no es propio del diseño de microprocesadores, también puede aplicarse al diseño de hardware digital de control de cualquier dispositivo de entrada/salida como CRT's, floppys, teclados, controladores de memorias dinámicas, etcétera.

Las máquinas secuenciales sincrónicas (o asincrónicas) conmutan de un "estado" a otro, según como se haya cableado el algoritmo en ellas. En cada estado por el que pasan pueden producir o no, una o varias salidas. La conmutación de un estado "x" a un estado "y" o a un estado "z" depende de una condición de entrada que corresponde a una señal física digital. En otras palabras, la máquina puede tomar decisiones al respecto como se ha planteado en un algoritmo. Para la síntesis de las máquinas secuenciales se debe tener con anterioridad el algoritmo a cablear, éste se representa normalmente mediante un diagrama llamado "diagrama de estados".

3. VERIFICACION DE LOS ALGORITMOS DE HARDWARE MEDIANTE LOS DIAGRAMAS DE ESTADO

Cuando se diseña un software se asocia a este proceso la concep-

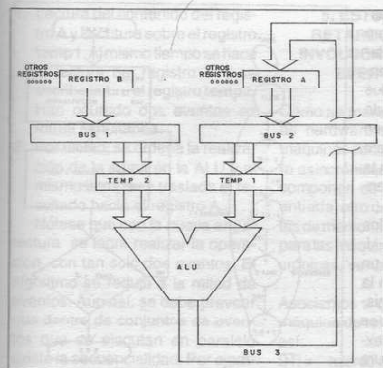


Figura 1. Hardware con arquitectura de BUS único

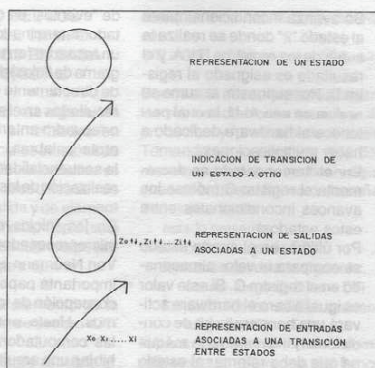


Figura 3. Elementos gráficos de un diagrama de estados

ción de un algoritmo y la prueba del mismo, mediante un diagrama de flujo. El empleo de diagramas de flujo tampoco es propio del diseño de software. Los diagramas de flujo son herramientas que empleamos para facilitar el test de los algoritmos concebidos y así minimizar los posibles errores en el sistema. Para el desarrollo de software se tienen una serie de símbolos, los cuales generan un gráfico en donde se puede "ver" el camino por donde fluye la información según un algoritmo. Estos símbolos se mues-

tran en la figura 2. Sin embargo, esta metodología se emplea en muchos otros campos del conocimiento como en la organización de operaciones, sistemas de control, diseño de circuitos integrados y, por supuesto, en el diseño de hardware digital.

Como se mencionó, los algoritmos son cableados en una máquina secuencial sincrónica (o asincrónica) y la metodología de test, de los mismos, se hace mediante un "diagrama de estados". Los elementos gráficos que constituyen un diagrama de estados se muestran en la fig. 3. Comparemos un algoritmo representado mediante un diagrama de flujo y uno de estados, donde se realiza una multiplicación por el método de sumas sucesivas. Por supuesto, el primer algoritmo será parte de un programa (software) y el segundo será parte de un circuito digital (hardware). Ver la figura 4.

El diagrama de flujo representa el siguiente algoritmo:

1. La señal de "start" colocada al inicio del

diagrama indica el punto de partida para el análisis. En el siguiente recuadro se establecen las variables:

result = 0 (almacena el resultado)
cont = multiplicador (contador)
var = multiplicando (variable)

Se avanza al siguiente recuadro donde se realizan las sumas sucesivas así:

result = result + var
cont = cont + 1

Se procede a comparar el valor del contador en un rombo de decisión. Si "cont" es diferente de cero se retorna a sumar de nuevo, de lo contrario se termina.

El diagrama de estados representa el siguiente algoritmo:

1. Se inicia con un estado de espera (wait) donde se prueba la señal digital llamada mult, si mult=0 la máquina secuencial permanece en el estado de espera, de lo contrario (si mult=1) la máquina avanza al siguiente estado a partir del cual se inicia la multiplicación en hardware. Los valores a multiplicar han sido almacenados en registros con anterioridad.
2. En el estado "1", se lee el valor almacenado en el registro B y se escribe en el registro C.

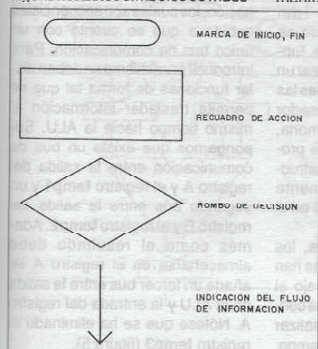


Figura 2. Elementos gráficos de un diagrama de flujo

3. Se avanza incondicionalmente al estado "2" donde se realiza la suma de los registros D y A y el resultado es asignado al registro D. Por supuesto, la suma se realiza en una ALU, la cual pertenece al hardware dedicado a hacer multiplicaciones.

4. En el tercer estado se decrementa el registro C (nótese los avances incondicionales entre estos estados).

5. Por último, en el cuarto estado se compara el valor almacenado en el registro C. Si este valor es igual a cero el hardware activará una bandera (o bit de condición=1) indicando a la máquina que debe retornar al estado de espera, pues ya ha concluido su función, de lo contrario la bandera permanece desactivada indicando a la máquina que debe regresar al estado 2. El resultado quedará almacenado en el registro D.

En cualquiera de los dos casos puede verificarse fácilmente el funcionamiento del algoritmo de multiplicación, utilizando los diagramas de flujo o estados respectivamente. Debe aclararse que cualquier software que se realice se ejecuta, en últimas, en un hardware de propósito general (un microprocesador), sin embargo los niveles de abstracción involucrados en un sistema complejo como un microcomputador, permiten aislar el software de alto nivel del hardware, haciendo más fácil el trabajo para el programador, pero permitiendo que todos (diseñadores de hardware y software) utilicen los mismos conceptos básicos de diseño.

4. SECUENCIALIDAD Y PARALELISMO EN LA REALIZACIÓN DE LOS EVENTOS

El concepto de secuencialidad está asociado directamente con los diagramas de flujo o de estados mencionados anteriormente. En ellos se ve claramente que la ejecución

de "eventos" en un estado determinado o en un recuadro (en un diagrama de flujos) depende directamente de los resultados en el estado o recuadro anterior, en otras palabras, existe la secuencialidad en la realización de los eventos.

En los inicios de los microprocesadores, J. Von Neumann jugó un importante papel en la concepción de los mismos. Hasta entonces las computadoras exhibían una arquitectura tal, que resultaba imposible integrar todas las funciones en un solo chip. Las ideas de "programa almacenado"

e "interconexión mediante un solo canal de comunicaciones entre funciones (conocido como BUS)" fueron propias de Von Neumann y otros pioneros de las computadoras y permitieron integrar por primera vez un procesador en un chip. Sin embargo estas arquitecturas llevaron a lo que se conoce como "el cuello de botella de Von Neumann". Una condición mediante la cual no se logra un incremento apreciable de la eficiencia del microprocesador, aún cuando se incremente su frecuencia de funcionamiento. El hecho de utilizar un único bus para comunicar todas las funciones de un microprocesador (como: ALU, registros, memoria, etc) obliga a que los eventos propios de la ejecución de una instrucción se hagan completamente secuenciales (como ya se vio en el ejemplo inicial, figura 1).

Para lograr mayor eficiencia, los diseñadores de computadoras han tomado otras alternativas bajo el concepto del paralelismo. Este concepto indica que es posible realizar más de un evento al mismo tiempo, en vez de realizar cada evento en forma secuencial. Sin duda esta

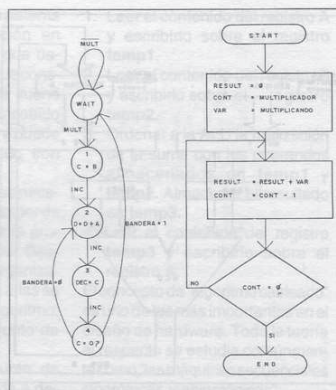


Figura 4 Representación de algoritmos que realizan la operación de multiplicación en software y hardware.

alternativa incrementa la eficiencia de un computador, el problema se concentra ahora en que se debe establecer qué eventos se realizan necesariamente en forma secuencial y cuales pueden ser realizados al mismo tiempo o paralelamente.

Analicemos estos conceptos con un ejemplo. Ya se ha visto que la realización de la suma, según el algoritmo del primer ejemplo, requiere de 4 eventos para su culminación, los cuales son secuenciales debido a que se cuenta con un único bus de comunicación. Para introducir paralelismo se deben aislar funciones de forma tal que se permita trasladar información al mismo tiempo hacia la ALU. Supongamos que existe un bus de comunicación entre la salida del registro A y el registro temp1 y un segundo bus entre la salida del registro B y el registro temp2. Además como el resultado debe almacenarse en el registro A se añade un tercer bus entre la salida de la ALU y la entrada del registro A. Nótese que se ha eliminado el registro temp3 (figura 5). Bajo estas condiciones el algoritmo puede reescribirse como:

1. Lectura del contenido del registro A y escritura sobre el registro temp1. Al mismo tiempo se hace la lectura del registro B y se escribe sobre el registro temp2. Han ocurrido dos eventos en forma simultánea.

2. Por último, se ordena la realización de la suma en la ALU y al mismo tiempo se traslada el resultado hacia el registro A.

Nótese que con la nueva arquitectura se logra realizar la operación con tan solo dos eventos. El algoritmo se redujo a la mitad de eventos. Aún así, se debe prever que dentro de conjuntos de eventos que se ejecutan en paralelo existe la secuencialidad. Por ejemplo, en el evento 2 del anterior algoritmo hay que dar tiempo a que la ALU obtenga el resultado antes de que se traslade al registro A. La mayor eficiencia en un diseño se logra sabiendo combinar el paralelismo con la secuencialidad, para lograr de la mejor manera el objetivo del hardware que se está diseñando. Por último, se analiza la influencia de los retardos de tiempo en el funcionamiento de las máquinas secuenciales sincrónicas.

5. ESTUDIO DE LOS RETARDOS DE TIEMPO INVOLUCRADOS EN CADA EVENTO DIGITAL

Como se mencionó, los algoritmos de hardware son sintetizados en máquinas secuenciales sincrónicas (o asincrónicas). Estos circuitos se componen de un decodificador de entrada, otro de salida y de elementos de memoria como flip-flops (solo para las máquinas secuenciales sincrónicas, ver la figura 6).

Asociemos a cada elemento de la máquina de retardo de tiempo "DT" así:

DT_i = retardo de tiempo asociado al decodificador de entrada

DT_o = retardo de tiempo asociado al decodificador de salida

DT_m = retardo de tiempo asociado a los elementos de memoria.

Tanto los decodificadores de entrada como de salida se sintetizan comúnmente de mapas de Karnaugh. Esto implica que estén expresados como suma de productos o como productos de sumas, es decir, se tiene una primera columna de compuertas AND y una compuerta OR, o una columna de compuertas OR y una compuerta AND, respectivamente (ver fig. 7).

Para facilidad del análisis se supone que todas las compuertas pertenecen a una misma familia de circuitos integrados, en ese caso: $DT_m + DT_i$ puede ser mayor que $DT_m + DT_o$ (I)

o

$DT_m + DT_i$ puede ser menor que $DT_m + DT_o$ (II)

Supóngase que la condición (II) ocurre. Se puede analizar la respuesta temporal mediante un diagrama como el de la figura 8.

Podríamos acordar el período de reloj hasta el retardo máximo que en este caso corresponde a $DT_m + DT_o$. Bajo estas condiciones las salidas ocurrirían desfasadas por un período de reloj respecto del estado presente y cada salida sería válida por un período de reloj, pero aún así la máquina funcionaría.

Para el diseño de las máquinas se debe tener en cuenta los retardos

cualquiera de los decodificadores es:

$$DT_i = DT_o = DT_{max-and} + DT_{max-or}$$

Donde esta ecuación es válida para las representaciones en suma de productos y producto de sumas. Tomando en cuenta los retardos mencionados puede verificarse que una vez ocurra el flanco de la señal de reloj se tiene:

estado presente:

ocurre luego de DT_m

formación del estado siguiente:

ocurre luego de $DT_m + DT_i$

formación de las salidas: ocurren luego de $DT_m + DT_o$.

Nótese que se ha tomado como referencia la ocurrencia del flanco del reloj (para el ejemplo es ascendente).

Estas ecuaciones indican que el período del reloj que coordina una máquina secuencial debe tener una frecuencia máxima. Sin tener en cuenta ninguna otra condición de tiempo se puede decir que el mínimo período de reloj (o máxima frecuencia) corresponde al valor mayor entre $DT_m + DT_i$ y $DT_m + DT_o$. Estos valores pueden cambiar en el caso en que no se cumpla con la condición de que las compuertas de todos los decodificadores pertenezcan a una misma familia de circuitos integrados, en ese caso: $DT_m + DT_i$ puede ser mayor que $DT_m + DT_o$ (I)

o

$DT_m + DT_i$ puede ser menor que $DT_m + DT_o$ (II)

Supóngase que la condición (II) ocurre. Se puede analizar la respuesta temporal mediante un diagrama como el de la figura 8.

Podríamos acordar el período de reloj hasta el retardo máximo que en este caso corresponde a $DT_m + DT_o$. Bajo estas condiciones las salidas ocurrirían desfasadas por un período de reloj respecto del estado presente y cada salida sería válida por un período de reloj, pero aún así la máquina funcionaría.

Para el diseño de las máquinas se debe tener en cuenta los retardos

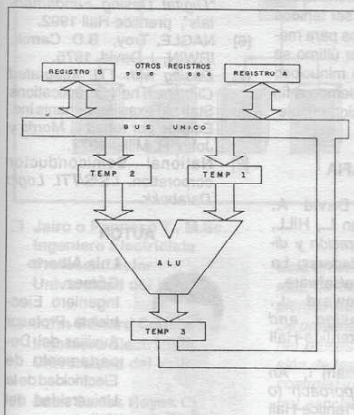


Figura 5. Arquitectura de la figura 1 modificada para un mayor paralelismo.

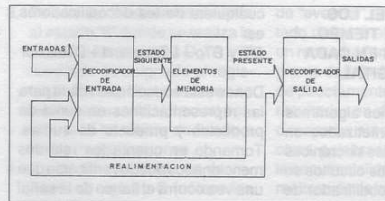


Figura 6. Esquema general de una máquina secuencial sincrónica.

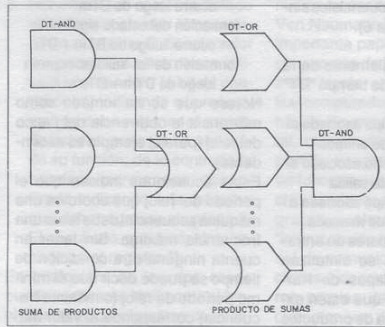


Figura 7. Representación en suma de productos de cualquier producto de sumas.

de tiempo y por consiguiente las familias de los circuitos integrados con que se fabricará el hardware. Por supuesto, si el diseño se llevará a un ship, se debe contar con simuladores para el análisis de los mismos. Para prevenir posibles carreras de tiempo (que provocarían mal funcionamiento en las máquinas, etc.

6. CONCLUSION

En el diseño de hardware se conjugan muchos conceptos importantes como retardos de tiempo, algoritmos, diagramas de estado, secuencialidad y paralelismo. Estos conceptos son completamente universales y aplicables en nuestra vida cotidiana. Los algoritmos y su comprobación mediante un diagrama de estados son una herramienta

fundamental para el diseño de cualquier hardware digital que se requiera. La secuencialidad y el paralelismo deben ser tenidos en cuenta en los diseños para mejorar su eficiencia y por último se debe hacer un estudio minucioso sobre los retardos de tiempo a fin de que el sistema funcione fiablemente.

7. BIBLIOGRAFIA

- [1] PATTERSON, David A, HENNESSY, John L., HILL, McGraw. *Organización y diseño de computadoras*, La interfaz, Hardware/software.
- [2] POLLARD, Howard L. *"Computer Design and Architecture"*. Prentice-Hall 1990.
- [3] FLETCHER, William I, *"An Engineering approach to Digital Desing"*. Prentice-Hall 1980.

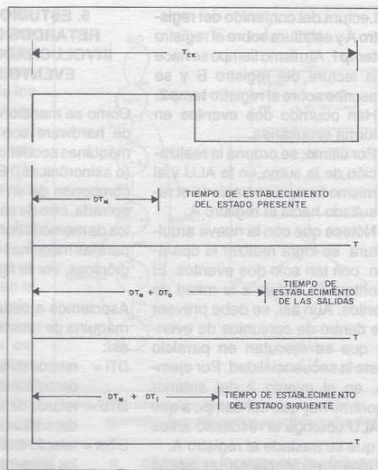


Figura 8. Respuesta temporal de una máquina secuencial sincrónica bajo la condición (II).

- [4] WAKERLY, Jonh F, *"Micro-computer Architecture" and Programming*, the 68.000 family. John and Sons 1989.
- [5] BREEDING, Kenneth J, *"Digital Desing Fundamentals"*, prentice-Hall 1992.
- [6] NAGLE, Troy, B.D. Carroll, IRWIN, J. David, 1975.
- [7] *Desing with TTL Integrated Circuits*. The I.C. applications Staff of Texas Instruments Inc. Editado por robert L. Morris y John R. Miller 1971.
- [8] National Semiconductor corporation. *LS/S/TTL Logic Databokk*.

AUTOR

Luis Alberto

Gómez

Ingeniero Electricista. Profesor Auxiliar del Departamento de Electricidad de la Universidad del Valle

