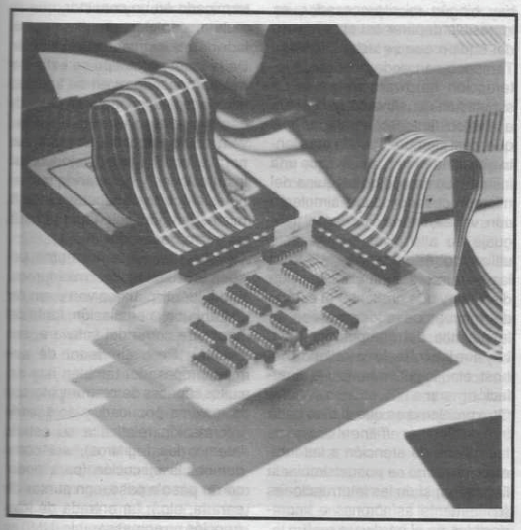
Nuevo sistema para emular microprocesadores



Angel García Baños
Ingeniero de Telecomunicación
Profesor Asistente
Departamento de Electricidad
Universidad del Valle

RESUMEN

Hay varias maneras de depurar y reparar tarjetas basadas en microprocesadores. Se muestran los métodos más tradicionales y se propone una nueva alternativa, ideada por el autor, la cual está en proceso de construcción: el emulador híbrido. Las características de este nuevo sistema se analizan sobre un diseño previo también realizado por el autor.

ABSTRACT

There are many ways to debug and repair microprocesor-based hardware. Some traditional methods are showed and then a new approach, conceived by the author, is proposed: the hibrid emulator. The features of this new system are analyzed on a previous design, also made by the author.

PALABRAS CLAVES (KEYWORDS)
Simulación. emulación, microprocesador

1. INTRODUCCION

Para depurar equipos basados en microprocesador durante la fase de diseño y construcción de prototipos, se emplean habitualmente cuatro tipos de herramientas: los simuladores, los emuladores, los analizadores lógicos y los depuradores embebidos. Estas herramientas también se usan durante la fase de mantenimiento de los mencionados equipos, para reparar las averías que se presenten. Además, son imprescindibles para la enseñanza en asignaturas que tengan que ver con microprocesadores.

Se realizará una breve descripción

de todos ellos, para luego pasar a describir el nuevo sistema de depuración propuesto: el emulador híbrido. Este sistema se está construyendo en una tesis de maestría. Para describir más en detalle su funcionamiento se mostrará una primera versión de características más limitadas, realizada para emular el microprocesador 6809, y que funcionó a plena satisfacción.

2. METODOS TRADICIONALES DE DEPURACION

2.1. Simulador

Se trata de un paquete de software que simula la ejecución de cada instrucción del microprocesador. El paquete corre sobre un computador *host* de propósito general y está escrito en un lenguaje de alto nivel (usualmente C).

Diseñar un simulador para un microprocesador es relativamente sencillo: si el microprocesador posee un juego de registros (por ejemplo A, B, X, Y, PC, SP), el simulador los implementa como variables (en lenguaje C sería algo así como: int A, B, X, Y; long PC, SP;). El simulador va leyendo las instrucciones del programa ensamblador de un fichero, las decodifica y simula su ejecución, afectando a las variables que juegan el papel de registros del microprocesador. Por ejemplo, las siguientes instrucciones de ensamblador del microprocesador 68000:

```
MOVE D5,D2
ADD (A3)+,D2
```

darán lugar a la ejecución del siguiente código en lenguaje C:

```
D2=  D5;
D2=  D2+*A3;
A3=  A3+1;
```

O, escrito más compacto:

```
D2=  D5;
D2+= *A3++;
```

En realidad las cosas son un poco más complicadas por que también hay que reflejar, en el registro de estado, los cambios producidos por la ejecución de cada instrucción (el bit de *carry*, el de cero, el de *overflow*, etc.). Asimismo, se debe construir, en lenguaje C, una función capaz de calcular la dirección efectiva de cualquiera de los modos de direccionamiento que posea el microprocesador en cuestión. Pero ambas cosas son de fácil implementación.

Puesto que se trata de una simulación por software y realmente no hay ningún microprocesador, es imposible depurar así el hardware del equipo que se está diseñado. Tampoco se puede depurar la interacción hardware-software. La simulación usualmente se efectúa a la velocidad del computador *host* que, casi siempre, resulta más lento que el hardware real, ya que una instrucción de código máquina del microprocesador debe simularse con varias instrucciones del lenguaje de alto nivel que se esté utilizando. Además, hay que traer las instrucciones de un fichero de disco, hay que presentar el estado del microprocesador simulado en la pantalla del computador host, hay que leer el teclado y el ratón del host, etc., lo cual ralentiza la simulación.

Otro problema es que si bien nada impide ejecutar *off-line* el código de las rutinas de atención a las interrupciones, no se puede simular la llegada en sí de las interrupciones (usualmente asíncronas e impredecibles en el hardware real).

Dado que todo está hecho por software, los simuladores ofrecen muchas herramientas muy potentes de depuración (ejecución paso a paso, ejecución en sentido inverso, presentación y modificación de registros, memoria, stack, etc., puntos de parada (*breakpoints*) que pueden depender de condiciones complejas, como el estado de variables o el acceso a ellas, etc. Y todo ello en un ambiente gráfico, interactivo, con ventanas, menús y ayudas en línea.

El hecho de que sea sólo software y relativamente fácil de diseñar, hace que los simuladores sean herramientas muy económicas al alcance de cualquier persona, empresa o institución, dedicada al diseño de hardware basado en microprocesadores. Y es fácil encontrar simuladores de diversas marcas para todo tipo de microprocesadores.

2.2. Emulador

Es un equipo del que sale un cable terminado en un conector, que simula mecánicamente el encapsulado del microprocesador a emular. De la tarjeta prototipo se extrae el microprocesador y en su lugar se introduce este conector. A partir de ese momento, el emulador sustituye al microprocesador de una manera exacta (o casi), ejecutando instrucciones y generando las señales eléctricas adecuadas con la misma temporización que lo haría un microprocesador real.

Usualmente, los emuladores poseen en su interior un microprocesador idéntico al que van a emular (de ahí que la emulación, tanto del hardware como del software, sea exacta). Pero alrededor de ese microprocesador también hay circuitos capaces de interrumpirlo, con la máxima prioridad, y de sacarle información relativa a su estado interno (los registros), así como detener la ejecución (para poder correr paso a paso, con puntos de parada, etc.). La entrada de interrupción no enmascarable (NMI) es la que usualmente está dedicada a esta tarea, con lo cual esa señal no se puede emular.

Estos sistemas permiten depurar el hardware, el software y la interacción entre ambos (salvo la entrada de interrupción NMI, dedicada a la emulación). La ejecución de programas ocurre en tiempo real, es decir a la misma velocidad que el microprocesador real.

Los emuladores disponen de algu-

nas herramientas para la depuración interactiva, pero son pobres e incómodas de manejar debido a que no pueden dedicar mucho tiempo ni recursos a la interfaces de presentación, ya que están dedicados casi exclusivamente a la emulación.

Los precios de los emuladores son altos o muy altos y usualmente existen sólo 1 o 2 modelos para cada microprocesador. El emulador es un equipo dedicado a un sólo microprocesador, de modo que si se desea trabajar con otro microprocesador hay que comprar otro emulador. Y aunque existen emuladores "universales", su costo es mayor aún, y se basan siempre en adaptar módulos de "personalización" al microprocesador deseado, con lo cual también hay que hacer gastos bastante elevados cuando se quiere trabajar con otro microprocesador.

2.3. Analizador lógico

También se puede emplear un analizador lógico conectado a las patas del microprocesador real que se quiera depurar. El analizador lógico debe tener la posibilidad de desensamblar las instrucciones y datos que capture del respectivo bus. La desventaja de este método es que no puede correrse paso a paso el software a depurar, y que no es fácil leer variables ni tampoco pueden modificarse sobre la marcha. Además, al trabajar con este sistema se termina inevitablemente con un gran listado de código que hay que analizar. Y no hay etiquetas simbólicas, por lo que la depuración puede ser muy dificultosa. Su principal ventaja es que la ejecución es en tiempo real y no se sacrifica ninguna entrada de interrupción para realizar la emulación.

2.4. Depurador embebido

Consiste en diseñar, desde un principio, el depurador dentro del microprocesador. Para ello hay que dedicar un canal de comunicación (usualmente RS-232C) para recibir comandos y enviar datos hacia un computador externo desde donde se controla la depuración. Y hay que reservar una pequeña porción de la memoria EPROM y de la RAM para el depurador. Desde el computador externo se pueden enviar comandos para leer los registros, para modificarlos, para ejecutar código a partir de una dirección dada, etc. Como ventaja está su bajo costo. Como desventaja cabe destacar que si el hardware tiene algún tipo de problema el depurador no funcionará, ya que está embebido en el sistema. Por tanto, no sirve para depurar el hardware. Además usa recursos del microprocesador (EPROM, RAM y puerto RS-232C), que pueden ser escasos.

3. LA NUEVA ALTERNATIVA: EL EMULADOR HIBRIDO

En un sistema basado en microprocesador hay dos *escalas de tiempo* distintas:

- La *escala de tiempo* para las señales hardware.
- La *escala de tiempo* para las instrucciones software.

En la vida real (en las aplicaciones finales y en los emuladores), ambas escalas coinciden con la escala de tiempo intuitiva que todo el mundo conoce. Sin embargo, para el propósito que perseguimos, es conveniente diferenciarlas a partir de ahora.

La idea básica es que se puede diseñar un emulador que genere las mismas señales hardware que el microprocesador y con la misma temporización (es decir, en la misma escala de tiempo del hardware), pero en el cual, las instrucciones se ejecuten simuladamente, por software (es decir, que la escala de tiempo del software no se respeta sino que se alarga).

Esto se puede lograr diferenciando, en el microprocesador a emular, dos partes (que podrían no existir como tales físicamente, pero que siempre existen conceptualmente):

- La UE (unidad de ejecución): es la unidad encargada de la ejecución de las instrucciones.
- La UI (unidad de interface): es la unidad encargada de interactuar con el hardware, generando los ciclos de bus de lectura, de escritura, aceptando interrupciones y reset, manejando el DMA, etc.

El emulador híbrido dispondrá entonces de dos partes también:

- Un computador de propósito general (PC compatible, para nuestro caso) sobre el cual correrá un software UE, encargado de la simulación de la ejecución de las instrucciones.
- Una tarjeta de lógica digital reprogramable (basada en FPGAs) en la cual se programará la UI, para que emule las señales hardware del microprocesador.

La conexión y sincronización de ambas partes se hará siguiendo la siguiente filosofía:

- Los ciclos de lectura y escritura de bus se desencadenan en la UI a petición de la UE.
- Los ciclos de reconocimiento de interrupciones, petición de DMA, petición de bus, etc. se realizan completamente en la UI, que a su vez informará de tal ocurrencia a la UE.
- Hay que identificar el ciclo de bus equivalente a NOP (no-operación) en el microprocesador que se quiera emular. Esto es, hay qué averiguar (consultando el manual de operación del microprocesador) cual debe ser el estado de las señales hardware del microprocesador para que éste no realice ninguna operación.
- La UI insertará automáticamente estos ciclos NOP, mientras la UE esté atareada realizando la simulación lenta de cualquier instrucción.

De esta manera se puede conseguir un sistema híbrido entre el

simulador y el emulador, que combina el bajo costo del primero con la habilidad de manejo de hardware del segundo.

El trabajo de construcción del emulador híbrido se llevará a cabo en una tesis de maestría en el Departamento de Electricidad de la Universidad del Valle. La unidad de ejecución (UE) se realizará en lenguaje C para Windows, corriendo en un computador PC 486. La unidad de interface (UI) se realizará como una tarjeta para el bus ISA-16 del mencionado computador. Esta tarjeta se diseñará alrededor de una FPGA (Field Programmable Gate Array) de XILINX, para poder emular las señales hardware de cualquier microprocesador de 8 y de 16 bits, con encapsulados de hasta 64 patas. Se recuerda que las FPGAs son circuitos lógicos que contienen miles de puertas y flip-flops, y cuya configuración interna y conexionado se pueden programar. Las FPGAs del fabricante XILINX almacenan su configuración lógica interna en una memoria RAM, de modo que se pueden reprogramar tantas veces como se necesite, permitiendo así la emulación de cualquier tipo de microprocesador de complejidad no muy grande.

3.1. Trabajo previo realizado

Hace unos años el autor del presente artículo ya había desarrollado un emulador híbrido (de nombre EM6809) de características muy limitadas (por las tecnologías disponibles en aquel momento), pero que servirá de experiencia para realizar este otro.

Concretamente, el emulador ya realizado está basado en un computador Radio Shack (cuyo corazón es un microprocesador 6809). Debido a ello (y a que su diseño fue realizado con lógica convencional no reprogramable) este equipo únicamente sirve para emular el microprocesador 6809.

El computador Radio Shack posee un conector de expansión que sirve habitualmente para colocar allí una unidad de disco. En ese conector está disponible todo el bus del microprocesador 6809, y además una señal de *chip select* **SCS** para el disco. Como el disco está mapeado múltiplemente en memoria, se pudo estrechar su zona de memoria ocupada a la que realmente usa, generando un nuevo *chip select* **ESCS** para el disco, y creando así un espacio en memoria para los registros internos del emulador. Estos registros son cinco (tabla 1) y más adelante se explicará para que sirven.

Circuito	Señal de chip select	R/W	Función
U3	CSH\	W	Registro con la parte alta de la dirección
U4	CSL\	W	Registro con la parte baja de la dirección
U7	CSCONF\	W	Registro de configuración
U6	CSSTATUS\	R	Registro de estado
U5	CSDAT\R/	W	Registro de dato

Tabla 1: Registros del EM6809

El diagrama de bloques simplificado de este emulador (figura 1) muestra tres conectores: el conector K1 procede del computador Radio Shack, mientras que K2 es el nuevo conector de salida para la unidad de disco. El conector K3 (al final del cable plano, en la fotografía 1) es la pinza de 40 patas que simula físicamente el encapsulado del 6809 a emular. Todas las señales procedentes de K1 van a parar a K2 (salvo la mencionada SCS\ que se modifica a ESCS\) para así lograr que el disco continúe operativo.

En cuanto al emulador propiamente dicho (K3), recibe directamente las señales E, Q y R/W\ de K1, que son las señales de sincronismo del bus del 6809. Por tanto, cualquier hardware externo conectado a K3, ve en este conector una réplica idéntica a la temporización de bus de un 6809 real.

No se va a mostrar en detalle el software del emulador, porque tiene cierta complejidad y extensión. Baste decir que las instrucciones del 6809 se simulan con macros, que realizan la misma función que la instrucción simulada más un acceso al bus. Si se trata de un acceso de escritura, primero se escribe la parte alta de la dirección (a la cual se desea acceder) en el registro U3. Luego se escribe la parte baja de la dirección en U4 y, por último, se escribe el dato en U5. Esta última escritura desencadena en el emulador un ciclo de escritura real, ya que se habilitan las salidas respectivas de U3, U4 y U5 en sincronismo con las señales de temporización de bus E, Q y R/W\. A cualquier tarjeta externa conectada a K3 le dará la impresión de que hay un microprocesador 6809 real accediendo al bus.

El acceso en lectura es similar: la macro primero escribe en U3 la parte alta de la dirección que quiere leer, luego escribe en U4 la parte baja y luego lee de U5 el dato. En ese momento el emulador inicia un ciclo de lectura real del dato sobre el hardware externo conectado a K3. El dato pasa a través del buffer bidireccional U5 y llega hasta el computador Radio Shack, donde lo recibe la macro que desencadenó la lectura.

En definitiva, todas las instrucciones se ejecutan simuladamente sobre el computador Radio Shack, pero afectan al hardware real a través del conector K3 de emulación.

Conviene resaltar que hay muchos momentos en los cuales no hay ciclos de acceso de lectura ni de escritura, ya que el computador

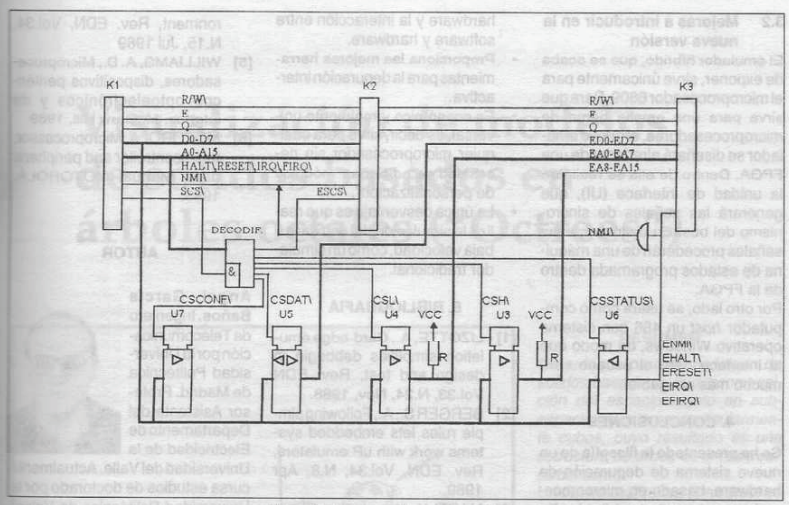


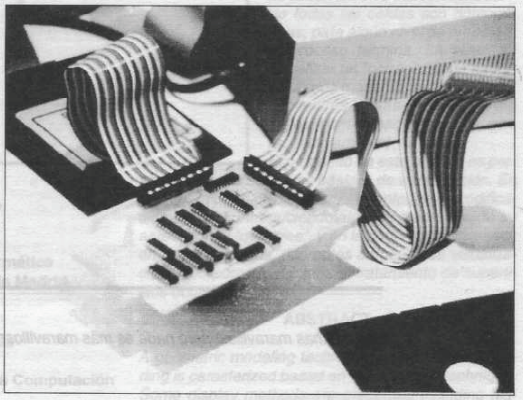
Figura 1: Diagrama de bloques del EM6809

Radio Shack se encuentra simulando por software la ejecución de las instrucciones. En esos momentos hay que engañar al hardware externo, para que no se dé cuenta que las instrucciones se ejecutan muy lentamente. Para ello se insertan ciclos de NOP (no-operación) en el bus. En el 6809 un ciclo de no-operación se caracteriza porque el bus de direcciones mantiene $FFFF. Y eso se logra con las resistencias de *pull-up* a la salida de U3 y U4.

La emulación no resultaría completa sin las entradas de ERESET\, EHALT\, y las interrupciones ENMI\, EIRQ\ y EFIRQ\. Estas cinco entradas se capturan en U6. Además, cada vez que alguna de ellas se activa se produce una interrupción NMI\ real al Radio Shack. En éste, la respectiva rutina de atención de interrupciones está programada para leer el registro U6 y simular por software la correspondiente acción que haya que realizar.

El registro de configuración U7 sirve para realizar un autotest del emulador, así como para enmascarar la NMI\ real que se acaba de mencionar y también para habilitar el conector K3 (ello no está reflejado en la figura 1, con el fin de simplificar el dibujo).

Todo esto funcionó a la perfección en la practica, resultando ser este equipo EM6809 de mucha utilidad para realizar y depurar diseños basados en el 6809.



Fotografía 1: Emulador híbrido EM6809

3.2 Mejoras a introducir en la nueva versión

El emulador híbrido, que se acaba de exponer, sirve únicamente para el microprocesador 6809. Para que sirva para una amplia gama de microprocesadores, el nuevo emulador se diseñará alrededor de una FPGA. Dentro de ella se realizará la unidad de interface (UI), que generará las señales de sincronismo del bus. En definitiva, esas señales procederán de una máquina de estados programada dentro de la FPGA.

Por otro lado, se usará como computador *host* un 486 con sistema operativo Windows, de modo que la interface para el usuario será mucho más amigable.

4. CONCLUSIONES

Se ha presentado la filosofía de un nuevo sistema de depuración de hardware, basado en microprocesadores: el emulador híbrido. En resumen, sus características más importantes son:

- La simulación es exacta en todos los aspectos (hardware y software).
- Permite depurar el software, el hardware y la interacción entre software y hardware.
- Proporciona las mejores herramientas para la depuración interactiva.
- Es económico y realmente universal, es decir, sirve para cualquier microprocesador sin necesidad de adicionar "módulos de personalización".
- La única desventaja es que realiza la simulación del software a baja velocidad, como un simulador tradicional.

5. BIBLIOGRAFIA

[1] LIZOTTE, A., Card-edge emulation simplifies debbugin in design and test, Rev. EDN Vol.33, N.24, Nov, 1988.

[2] BERGER S., A., Following simple rules lets embedded systems work with uP emulators, Rev. EDN, Vol.34, N.8, Apr 1989.

[3] MARTI, K., Use a logic analyzer to dêbug real-time software, Rev. EDN, Vol.34, N.8, Apr 1989.

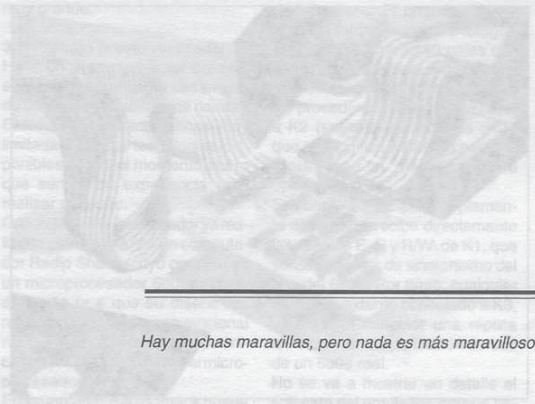
[4] LEIBSON, S. H., In-circuit emulators for uCs probe software activity in a closed environment, Rev. EDN, Vol.34, N.15, Jul 1989

[5] WILLIAMS, A. B., Microprocesadores, dispositivos periféricos optoelectrónicos y de interfaz, McGraw Hill, 1989

[6] MOTOROLA, Microprocessor, microcontroller and peripheral data, Manual MOTOROLA, 1985.

AUTOR

Angel García Baños. Ingeniero de Telecomunicación por la Universidad Politécnica de Madrid. Profesor Asistente del Departamento de Electricidad de la Universidad del Valle. Actualmente cursa estudios de doctorado por la Universidad Politécnica de Valencia. Dirige la Especialización de Computadores y Sistemas Digitales de PPIEEC. Desarrolla su trabajo de investigación en la línea "LOGICA DIGITAL REPROGRAMABLE BASADA EN FPGAs"



Hay muchas maravillas, pero nada es más maravilloso que el hombre.

Sófocles