Título: Dos métodos de diagnóstico de circuitos digitales de alta y muy alta escala de integración.

Autor: Dr. Ing. René J. Díaz Martinez. Profesor Titular. Dpto. de Automática y Computación. Fac. de Ingeniería Eléctrica. ISPJAE. Cuba.
Teléfono: 98-0244.
E-mail: renejdm@yahoo.com ó renejdm@electrica.ispjae.edu.cu

Abstract.
Two methods to detect the principal failures in LSI and VLSI digital circuits are presented in this paper. These methods can allow maintenance specialists to detect many faults in a brief time. These results can be used to create diagnostic software and BIOS for any microprocessors' based equipment.

Resumen.
En este trabajo se plantean dos métodos de diagnóstico que permiten detectar un grupo de fallos comunes en circuitos digitales de alta y muy alta escala de integración. Dichos métodos pueden ser de gran utilidad a los especialistas de mantenimiento para detectar fallos en un tiempo relativamente corto, así como emplearlos en el diseño de software de diagnóstico o en los programas BIOS de cualquier equipo con microprocesadores.

- Introducción.

Los sistemas electrónicos basados en la técnica de los microprocesadores (SEM) contienen dentro de sus arquitecturas circuitos de alta y muy alta escala de integración, por ejemplo microprocesadores, memorias, controladores programables, interfaces programables de E/S, etc. Dada su complejidad siempre resulta una tarea difícil la detección de un posible fallo, por lo que los diseñadores parten de una serie de consideraciones [1] que hacen más fácil dicha tarea, entre ellas:

1.- Consideraciones de diseño del hardware, esto es, la elaboración de una estrategia de diagnóstico dentro del hardware del sistema que puede constar de: sensores térmicos, técnica para la medición de IDDQ, pines para habilitar el autochequeo de los procesadores, un programa para la supervisión y control de estos dispositivos y un elemento de indicación de errores.

2.- Consideraciones de los bloques de trabajo, o sea, el diseño de una política de mantenimiento orientada a bloques o módulos del sistema (procesador, memorias, E/S).

3.- Consideraciones del sistema, representan el nivel superior y comprenden cómo detectar, localizar y reparar el módulo (componente) que falló.

Para materializar las consideraciones 2 y 3 será necesario diagnosticar los circuitos que componen el SEM a través de 3 alternativas posibles:

1.- Diagnóstico solo en el arranque del sistema, una de las más usadas por la sencillez de su materialización (memoria ROM y algunos bytes de RAM). Tiene como inconveniente que solo se ejecuta durante el arranque y por lo tanto si ocurre algún fallo durante la operación posterior del sistema, este no es comunicado, pudiendo acarrear grandes dificultades en su desempeño.

2.- Diagnóstico como tarea de fondo, el sistema es diseñado de manera tal que siempre esté ejecutando un programa de diagnóstico, y que cada vez que se desee realizar otra tarea, se le comunicará al sistema por interrupción. Esta alternativa requiere un mecanismo de interrupción adecuado, que valore las prioridades de las posibles tareas a realizar, así como de un análisis de tiempo para desarrollar las mismas sin entrar en conflictos.

3.- Diagnóstico como tarea del sistema, consiste en disponer de un programa de chequeo el cual puede ser invocado como una tarea más de las que puede realizar el sistema. Como en el caso anterior es necesario tener en cuenta el tiempo que dura esta tarea pues podría afectar el desarrollo de otras más prioritarias.

Tomando en consideración los elementos anteriores en este trabajo se proponen dos métodos para diagnosticar algunos fallos comunes a los circuitos mencionados.

- Propuesta de dos métodos de diagnóstico.

Un método de diagnóstico basado en pruebas funcionales determinísticas [2] se define a partir de un modelo o conjunto de fallos [3]. A su vez, el modelo de fallos se establece tomando en cuenta la arquitectura interna de los circuitos. De investigaciones realizadas por distintos autores puede notarse que existen fallos que son comunes a varios tipos de circuitos integrados, por ejemplo:

1.- Problemas en la decodificación de las direcciones de:

a) Las localizaciones de memoria.

b) Los registros generales de un microprocesador.

c) Los registros de trabajo de las interfaces y controladores programables.

2.- Problemas con los elementos de almacenamiento de la información.

a) Cortocircuito, puesta a cero o puesta a uno de las celdas que componen una localización.

b) Cortocircuito, puesta a cero o puesta a uno de los bits de los registros de microprocesadores, interfaces y controladores programables.

3.- Problemas a la hora de realizar operaciones básicas dentro del sistema.

a) Lectura y escritura de información en las memorias.

b) Lectura y escritura de los registros de microprocesadores, interfaces y controladores programables.

c) Decodificación de las instrucciones en los microprocesadores.

Considerando los elementos anteriores el autor de este trabajo definió los métodos de Marcha Modificado y Marcha por Columnas [1] derivados del método de Marcha estándar, con los cuales pueden detectar gran parte de los fallos planteados.
Marcha estándar es un método o prueba funcional determinística, derivada de un modelo de fallos y que se basa en la escritura/lectura de patrones binarios.
Marcha Modificado cambia los dos patrones originales del estándar, escogiéndose de la forma siguiente: sea n el número de bits de cada localización de la memoria, representándose cada bit de la localización o palabra como bi, con $1 \leq i \leq n$, y la palabra completa como b1b2b3 ... bn, si se divide en dos partes iguales la cantidad de bits, el patrón Pi puede representarse por:
Pi = [b1 b2 ... bn/2] [b(n/2)+1 b(n/2)+2 ... bn]
Y se hace:
f1 = [b1 b2 ... bn/2] y f2 = [b(n/2)+1 b(n/2)+2 ... bn]
Quedará de forma simplificada:
Pi = f1f2
Los valores de cada bit de los dos primeros códigos a utilizar por el método serán:
- P1, que cumplirá que:
1.- $f1 = \overline{f2}$
2.- b1 = b2 = ... = b(n/2) y b(n/2)+1 = b(n/2)+2 = ... = bn
- $P2 = \overline{P1}$
Con estos patrones se procede a ejecutar el método de Marcha estándar cuya secuencia de operaciones por cada localización i viene dada por las dos llamadas del procedimiento **Pass**, cuyo seudocódigo aparece en la figura 1. A continuación de ejecutar estos pasos, se escribirán y leerán los códigos de tantos unos como ceros restantes y que no fueron tomados (las dos instrucciones **for** al final del algoritmo).

```
MODIF MARCH TEST;
Const
    NC; {# de celdas de memoria}
    NE; {# máximo de elementos del arreglo}
Type
   Vector:= Array [1..NE] of Byte;
Var
   b: Vector;
   k, m, c: Integer;  {c: número de celdas, k: patrones, m: número de patrones}
Procedure Pass (b: Vector; c: Integer);
   for c:=1 to NC do  w^c_b  endfor;
   for c:=1 to NC do  r^c_b,  w^cj_b  endfor;
   for c:=NC downto 1 do  r^c_b,  w^c_b  endfor;
end Procedure;
Begin
   Pass (b1, c);
   Pass (b2, c);
   for k:=3 to m do
      for c:=1 to NC do  w^c_b[k],  r^c_b[k] endfor;
   endfor
End.
```

Figura 1. Algoritmo de Marcha Modificado.

Nota: Las operaciones r_b^c y w_b^c indican la lectura y escritura de un valor b en una celda de memoria c respectivamente, lo cual se puede extender a las restantes operaciones similares que aparecen en el algoritmo. Nótese que el algoritmo está diseñado para aplicárselo al arreglo de localizaciones de una memoria, pero de la misma forma se le puede aplicar al arreglo de registros de un microprocesador o de un controlador programable.
Haciendo un análisis del método se puede plantear que:
1º. Al ejecutar los dos primeros for del método se logra por cada unidad de almacenamiento "i" (localización o registro) escribir un patrón, leerlo y complementarlo, con lo cual se detectan problemas de selección y de cortocircuitos con las unidades de almacenamiento que le suceden.
2º. Con el tercer for se logra por cada unidad de almacenamiento "i" leer un patrón y complementarlo, con lo cual se detectan problemas de selección y de cortocircuitos con las unidades de almacenamiento que le anteceden, además de la puesta a cero y la puesta a uno.
3º. Al ejecutar por segunda vez Pass y los últimos for, se han escrito y leído los patrones de tantos unos como ceros que garantizan la detección del cortocircuito entre bits de una misma unidad de almacenamiento.
Si el método anterior se aplica por columnas deviene en el método de Marcha por Columnas **[1]**, cuyo algoritmo se muestra en la figura 2.

```
MARCH COL TEST;
Const
    NE; {# máximo de elementos del arreglo}
Type
   Vector:= Array [1..NE] of Byte;
Var
   b: Vector;
   i, j, R, S: Integer; {i: filas, R: máximo número de filas,j: columnas, S: máximo número
                         de columnas}
Procedure Pass (b: Vector; i, j, R, S: Integer);
   Var
      a_ij: Integer;
      {a_ij: memory cell address}
      for j:=0 to R-1 do
        for i:=0 to S-1 do w_b^a_ij endfor;
      endfor;
      for j:=0 to R-1 do
        for i:=0 to S-1 do r_b^a_ij, w_b^a_ij endfor;
      endfor;
      for j:=0 to R-1 do
        for i:=S-1 downto 0 do r_b^a_ij, w_b^a_ij endfor;
      endfor;
end Procedure;
Begin
   Pass (b1, i, j, R, S)
   Pass (b2, i, j, R, S)
End.
```

Figura 2. Algoritmo de Marcha por Columnas.

Este método detecta los fallos de V-acoplamientos simples para V=2, entre las celdas i y j de una misma columna del arreglo de una memoria, porque para ello ejecuta las operaciones planteadas y demostradas por Cokburn en **[4]**, y que son:

$2^{V-1}\, w_0^i$.

$2^{V-1}\, w_1^i$.

Una escritura inicial de la celda i además de las escrituras hechas en a) y b).

Una r^i cuando C^i se espere sea cero.

Una r^i cuando C^i se espere sea uno.

Del método se puede comprobar que por cada celda se ejecuta la secuencia $w_0^i, r_0^i, w_1^i, r_1^i, w_0^i, w_1^i, r_1^i, w_0^i, r_0^i, w_1^i$, que cumple con todas las operaciones enmarcadas en los cinco incisos anteriores.

Conclusiones.

Los métodos propuestos tienen algunas características que los hacen apropiados para detectar de forma rápida y sencilla problemas frecuentes en los circuitos de un SEM. En primer lugar son pruebas de escritura/lectura, cuya duración total guarda una dependencia lineal respecto al número de unidades de almacenamiento. Los códigos que utilizan pueden escogerse de entre una amplia gama de códigos binarios. Su programación resulta sencilla en cualquier plataforma que se materialicen, no ocupando espacios grandes de memoria, por lo que son apropiados tanto para las operaciones de mantenimiento como para su incorporación en los diagnósticos de autochequeo de encendido que incorporan los BIOS de cualquier SEM.

Referencias bibliográficas.

[1] Díaz, René J. Metodología para el análisis y cálculo de la mantenibilidad de los sistemas basados en la técnica de los microprocesadores. ISPJAE. Facultad de Ingeniería Eléctrica. Departamento de Automática y Computación. Ciudad de La Habana, julio 1999.

[2] Cockburn, B. F. Tutorial on semiconductor memory testing. Journal of Electronic Testing: Theory and Applications (Boston, EE.UU.), V 5, p. 321-336, 1994.

[3] Van de Goor, A. J. Testing semiconductor memories: Theory and Practice. Chichester, Gran Bretaña, John Wiley & Sons, 1991.

[4] Cockburn, B. F. Deterministic tests for detecting single V-coupling faults in RAMs. Journal of Electronic Testing: Theory and Applications (Boston, EE.UU.), V 5, p. 91-113, february 1994.