Técnicas de Diseño con Capacidad de Detección de Fallas para Circuitos Programables

Ing. Aquilino Cervantes Avila
Alumno de la Maestría del CINTEC-IPN.

En la implantación de diseños con base en circuitos programables surgen a menudo errores que son bastante difíciles de localizar. Lo anterior puede originarse por diferentes razones, desde escasa información del circuito por programar, hasta una gran complejidad del diseño. Sin embargo, uno de los peores casos reside en el momento en que se reunen varios circuitos aparentemente ya probados, pero que en conjunto no funcionan. En la mayoría de tales casos el desconcierto del usuario es bastante. Si además la implantación se realizó en un solo circuito, no es posible localizar exactamente el lugar de la falla, y si aunado a lo anterior tenemos que la simulación del diseño se comporta en forma adecuada, entonces existe un grave problema que requerirá gran cantidad de tiempo para corregir el o los posibles errores.

El propósito de este trabajo es presentar alternativas dediseño para concebir circuitos con la máxima capacidad de detección de fallos, a tales técnicas se les conoce como PRUEBABILIDAD. En pocas palabras, las técnicas de pruebabilidad presentan una forma de diseño de completa desconfianza, teniendo como lema «Primero piensa mal de tu diseño y posteriormente piensa peor». De esta manera es posible detectar problemas clásicos en sistemas digitales como son las carreras criticas, los riesgos estáticos y dinámicos, la metaestabilidad, además de problemas de otro tipo inherentes al propio diseño en cuestión.

Los métodos presentados a continuación fueron estudiados pensando en Dispositivos Lógicos Programables (PLD's) y particularmente en dispositivos MAPL ("Multiple Array Programmable Logic", Arreglos Logicos Programables Múltiples). Sin embargo, también es posible utilizarlos para circuitos programables tales como GATE ARRAY's y FPGA's.

Existen varias técnicas de prueba para circuitos LSI, cada una de las cuales posee sus propios métodos de generación y procesamiento de pruebas. No obstante, es posible dividir al conjunto de técnicas en dos grandes áreas a) Métodos concurrentes y b) Métodos explícitos.

Métodos concurrentes

El desarrollo de este método establece lógica redundante dentro del diseño para obtener diferentes puntos de prueba o de detección de errores. Se hace necesario definir algunos conceptos fundamentales para esta técnica, como son:

- Controlabilidad (CY).- Consiste en la habilidad de obtener un nivel alto o bajo en determinada salida, a partir de las entradas primarias.

- Observabilidad (OY).- Consiste en la habilidad para determinar el estado de un nodo interno a partir de las entradas primarias.

- Trayectoria de error.- Es el camino por el cual un error puede ser propagado hasta las salidas.

- Trayectoria paralela.- Describe la existencia de un camino alternativo para propagar una entrada para la misma compuerta y lograr la propagación hacia la salida.

- Trayectoria critica.- Se denomina de esta manera a la ruta lógica que se desea observar por su importancia dentro del diseño o por su susceptibilidad de errores.

El propósito de sumar lógica adicional al diseño principal es el obtener puntos de prueba para aumentar la detección de errores en el circuito. Debido a que las entradas y salidas se encuentran limitadas, es necesario agregar lógica de control, tal como decodificadores, multiplexores y registros de corrimiento para obtener de esta manera las características de controlabilidad y observabilidad necesarias.

Sin embargo, la selección de los puntos de prueba dependerá principalmente del tipo de diseño que se este ejecutando y de las consideraciones propias del diseñador.

Algunos de los nodos que pueden considerarse para agregar puntos de prueba son:

- A través de toda la trayectoria critica (pueden existir varias en un mismo diseño).
- Señales de reloj y control.
- Puntos de unión de lógica que pueda propagar un error, tales como contadores, sumadores, registros de corrimiento, codificadores/decodificadores, etc.

NOTA: El uso de las técnicas de pruebabilidad por estos métodos puede llegar a tal fascinación que un proyecto se torne muy complejo por la cantidad de elementos de redundancia que se agregaron al diseño, por tanto se recomienda ser moderado en la aplicación del método.

De forma general, la controlabilidad y observabilidad que posean los nodos internos del diseño influirá en forma directa en las consideraciones a tomar para generar y aplicar vectores de prueba. Para demostrar el uso del método expuesto considere la **figura 1**.

De esta figura, es posible observar que se aumentan 2 multiplexores (M) para poder tener entradas directas al bloque B; es decir, se aumenta la controlabilidad en este punto. Por otra parte también se agregan 2 demultiplexores (D) para poder observar las salidas del bloque B; en otras palabras, se incrementa la observabilidad de esenodo.

De esta manera es posible indicar que el circuito representado por la **figura 1b** representa un diseño orientado a pruebabilidad, en el cual es mas fácil identificar un error (para este caso se detectaría en que bloque se localiza el error).

Dependerá del tipo de diseño las consideraciones a tomar para aumentar la controlabilidad y observabilidad del circuito propuesto; es decir, para prueba de circuitos secuenciales podría ser útil emplear uno o mas circuitos de retención para garantizar que las entradas o salidas del circuito sean fijas. Pero también podría ser deseable para un determinado circuito, utilizar flip-flops como control y obtener de esta manera un diseño que trabaje por flancos y no por nivel (como trabaja regularmente un circuito combinatorio).

El propio diseñador es quien puede ofrecer las mejores heurísticas para decidir que tipo de elementos deberá colocar para lograr un diseño orientado a pruebabilidad por el método concurrente, ya que nadie mejor que el mismo sabrá los posibles puntos de falla en su diseño (teniendo en mente el lema de las técnicas de pruebabilidad).

Antes de continuar con el siguiente método, es conveniente decir que en el caso de que un diseño requiera de sincronización de tiempos, el método concurrente podría arrojar bastantes errores los cuales podrían ser dados por los retardos obtenidos por la lógica redundante. De esta forma, el circuito podría demostrar que funciona al realizar las pruebas particulares de los bloques internos del diseño y sin embargo fallar en el momento de trabajar todos como un solo proyecto. De estamanera, también se hace necesario trabajar de tal forma que toda o la mayor parte de la lógica asociada sea desechada en el producto final.

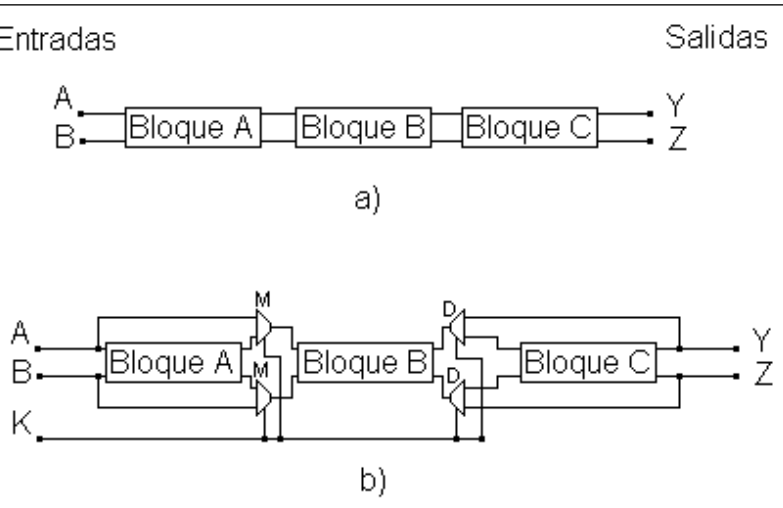


Figura 1. Diagrama a bloques de un diseño lógico.
a) Diseño sin pruebabilidad, b) Diseño con pruebabilidad

Métodos Explícitos

Todos los métodos explícitos separan el proceso de prueba del de operación normal. En general, los métodos cuentan con tres pasos.

1.- Generación de Patrones de Prueba.

El objetivo de este paso es el de generar patrones de entrada al sistema para que pueda detectarse cualquier error, bajo diferentes modos de operación. Este proceso es el paso mas importante de un método explícito y puede obtenerse por medio de cuatro técnicas

1.1 Generación Manual.

El diseñador debe hacer un análisis cuidadoso del diseño para establecer los patrones que puedan excitar o producir un error de la lista de errores considerados como posibles.

Este método es ampliamente utilizado para diseño de circuitos pequeños y tal vez algunos medianos; sin embargo, el proceso para optimizar la longitud de los patrones de prueba requiere una gran cantidad de trabajo y habilidades especiales en el área.

1.2 Generación Algorítmica

Existe una serie de algoritmos de uso especifico que permiten desarrollar los patrones de prueba necesarios para localizar una gran cantidad de errores, tales como las técnicas de Thatte - Abraham y Abadir - Reghbati. La primera describe la manera de generar patrones para microprocesadores y se basa en el desarrollo de operaciones o instrucciones y seguimiento del diagrama de estados correspondiente al microprocesador bajo prueba. Por otra parte, el segundo algoritmo se basa en una división del sistema total y en el seguimiento de rutas criticas a través de un diagrama de estados; de esta manera se localizan las posibles combinaciones de prueba que puedan arrojar un error.

1.3 Generación Aleatoria.

Este es el método mas sencillo de obtener y consiste en generar patrones de entrada al circuito (de forma aleatoria) y comprobar su comportamiento bajo las muestras indicadas.

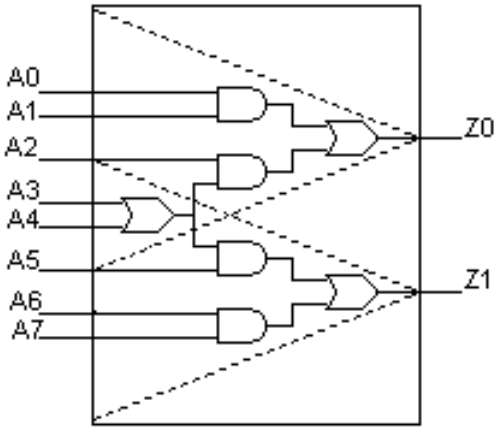


Figura 2. Demostración de reducción de patrones de entrada.

1.4 Generación por Simulación.

Para este método se utilizan algunasde las técnicas anteriores para desarrollar los patrones de prueba y además una herramienta CAE para realizar una simulación del desenvolvimiento del diseño con las entradas propuestas. Para lograr una simulación por lo menos deben conocerse los siguientes factores:

a) Descripción de diseño de compuertas escrito en algún lenguaje especial.
b) Condiciones iniciales de elementos de memoria.
c) Una lista de los posibles errores.

Antes de continuar con el segundo punto del método explícito se darán algunas consideraciones en base a las dos primeras técnicas, para desarrollar patrones de prueba, recordando que la experiencia y el conocimiento de determinado proyecto darán las mejores heurísticas para el desarrollo de tales patrones.

a) Obtención reducida de patrones de entrada.

Por lo regular se recurre a dividir un diseño especifico de tal manera que se reduzca el número de vectores que requiere un proyecto. Para la **figura 2**, si se toma el proyecto como una caja negra, debido a que posee 8 entradas se requerirán entonces $2^8 = 256$ vectores de prueba; no obstante, tomando en cuenta el interior del cuadro, es posible observar que se puede realizar una división de la manera mostrada, para la cual se tomarán $2^5 = 32$ vectores para el circuito superior y exactamente los mismos para el circuito de abajo, dando un total de 64 vectores en lugar de 256.

b) Algoritmo-D.

El algoritmo-D emplea la técnica de seguimiento de trayectorias criticas. El procedimiento que sigue el algoritmo se basa en lo siguiente: Considera A como la entrada capaz de propagar un error **ε** en la salida Z; si la entrada A en 1(0) tiene como respuesta ideal Z=0, en ese caso es condición suficiente Z=1 para considerar un error a través de esa trayectoria.

De esta manera, tenemos como antecedente los siguientes puntos :

- **Excitación**.- Los patrones de entrada deben ser especi-

ficados tomando en cuenta cual es el valor de salida para cada uno de ellos.

- **Propagación de error**.- Se requiere que las posibles trayectorias de error posean un camino hacia un punto que pueda ser observado (La observación puede ser directa o indirecta).

- **Justificación de línea**.- Los patrones de entrada deben recorrer o forzar un camino a través de las trayectorias del paso anterior, considerando todas las señales asociadas a ese camino.

Para facilitar el proceso de seguimiento de errores, se definirá la siguiente nomenclatura

D.- Representa una señal que tiene un valor 1 para un circuito sin error y 0 en caso contrario.

E.- Caja negra que contiene un circuito especifico.

Bloque_D .- Específica las entradas mínimas que deben ser aplicadas al elemento E para producir una señal de error D en las salidas de E.

Propagación_ D.- Específica las condiciones mínimas que son requeridas para lograr la propagación de errores a través de E o bloques E, hasta la salida.

Con este antecedente, el algoritmo posee la siguiente rutina:

1.- Excitación de fallos.- Se elige el Bloque_D apropiado y se obtienen las señales D del primer elemento E.

2.- Implicación.- En el paso 1, se elige un elemento E que puede o no tener retroalimentaciones internas o externas. Las implicaciones se dan considerando los diferentes valores que puede tomar una señal no conocida (porque pertenece a otro elemento E) y con esto localizar el valor de D.

3.- Propagación.- Todos los elementos en el circuito que no poseen salidas al exterior, y que tienen una señal D, son colocados en unlista llamada Limite_D. Cada elemento de Limite_D se asigna para que propague sus entradas hacia su salida; si esta frontera brinda resultados (que las salidas sean observadas) termina, en caso contrario se crea un nuevo elemento E estableciendo un nuevo límite.

4.- Implicación de propagación_D.- Se aplica la implicación para el elemento E localizado en el punto anterior.

5.- Se repiten los pasos 3 y 5 hasta que los errores hayan sido propagados hasta las salidas.

6.- Justificación de línea.- La ejecución de los pasos 1 a 5 puede resultar en especificar las salidas de un elemento E sin considerar algunas entradas no especificadas. Para corregir este error debe realizarse una unión de bloques E para someter a consideración las salidas en conflicto.

7.- Implicación de justificación de línea.- Se lleva a cabo cualquier implicación del bloque generado en el paso anterior.

8.- Se repiten los pasos 6 y 7 hasta que todas las salidas en todos los elementos sean completamente justificadas.

Es posible observar en el propio algoritmo que todas las combinaciones que puedan llevar a un error serán consideradas por él.

2.- Aplicación de Patrones de Prueba al Circuito.

Se pueden tomar dos consideraciones diferentes para el desarrollo de esta etapa.

2.1 Aplicación Externa

Esta etapa necesita el uso de equipo especial para inyectar los patrones de entrada en el circuito (tales como analizadores lógicos, generadores de señales, etc.).

2.2 Aplicación Interna

Se considera un sistema interno que aplique una serie de patrones establecidos, de tal manera que el propio circuito pueda probarse (Autocomprobación).

Lógicamente la primer opción brinda como ventajas el mejor control sobre el proceso de prueba, además de brindar todo el espacio del ASIC (Circuito integrado de aplicación especifica) únicamente para el diseño.

3.- Evaluación de las Respuestas del Circuito

Consiste en la identificación, a través de las respuestas de salida, de los bloques internos, trayectorias, etc.. en los cuales se detectó uno o más errores. Se tienen también dos maneras de implantar esta etapa:

3.1 Generación de Respuestas Extensivas

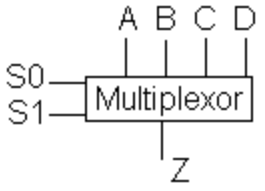
Desarrolla una estrategia en la cual se tienen las respuestas correctas que arroja el diseño y se compa-

ran con las que va encontrando el sistema al recibir los patrones de entrada al circuito. La forma de comparación lleva nuevamente a dos procedimientos diferentes: en el primero se almacena de forma previa el conjunto de respuestas del sistema (posiblemente en una PROM) y se efectúa la comparación. Por otra parte, el segundo método establece un diseño (que podría ser el del propio simulador) y efectúa la comparación de los valores obtenidos en el circuito ideal contra los que son generados al mismo tiempo en él circuito real (la diferencia se establece básicamente si la comparación se hace contra una serie de datos ya almacenados o contra un dato que es generado al momento).

3.2 Generación de Respuestas Compactas

El objetivo de esta técnica es reducir el espacio de almacenamiento requerido para establecer la comparación del modelo ideal contra el real; de esta manera se establecen dos técnicas para lograrlo.

La primera técnica es conocida como «Conteo de transiciones», y consiste en contar el número de transiciones lógicas de 1 a 0 o viceversa, para saber el número de cuenta necesario para comparar contra los resultados del sistema real. Para hacer más clara esta técnica, considere un multiplexor de 4



S0	S1	Z
0	0	A
0	1	B
1	0	C
1	1	D

Figura 3. Multiplexor 4 a 1

S0	S1	A	B	C	D	Z
0	0	1	X	X	X	1
0	0	0	X	X	X	0
0	1	X	1	X	X	1
0	1	X	0	X	X	0
1	0	X	X	1	X	1
1	0	X	X	0	X	0
1	1	X	X	X	1	1
1	1	X	X	X	0	0

Tabla 1. Tabla de transiciones de un multiplexor 4 a 1.

a 1, con su tabla de verdad en la **figura 3**.

Debido a que la salida Z puede cambiar entre 0 y 1 de acuerdo a la señal que sea elegida por S0, se tiene la **tabla 1**.

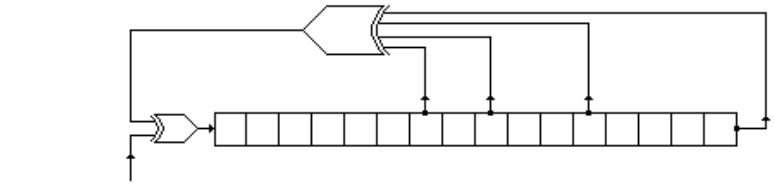


Figura 4. Registro retroalimentado de corrimiento de 16 bits.

En la tabla es posible verificar que Z sufre 7 transiciones considerando el patrón establecido; así, el método tomaría como patrón de entrada al circuito real las entradas para esta tabla (incluyendo señales de control) exactamente en ese orden; si el número de transiciones obtenidas por el circuito es diferente de 7 en ese caso el módulo especifico posee uno o varios errores.

Por otra parte, la segunda técnica es conocida como «análisis de retroalimentaciones» y consiste básicamente en un análisis repetitivo de las salidas con respecto a las mismas entradas. Cada salida obtenida de una combinación de entradas particular se somete a un registro de corrimiento, en donde algunas de las salidas de los flip-flops de ese registro son retroalimentadas a la entrada del mismo.

La información de las salidas se maneja en formato serial y se suministra al registro de corrimiento. El número de bits que debe poseer el registro de corrimiento es igual al número total de salidas que posee el circuito. En pocas palabras, el registro de corrimiento se utiliza básicamente para comprobar que las salidas del circuito anterior (que al inicio podrían ser las del modelo ideal), sean exactamente iguales a las del presente; es posible decir que lo que hace este circuito es una especie de verificación de paridad.

Conclusiones

Las técnicas de pruebabilidad brindan una mayor seguridad en el diseño con base en circuitos programables. De forma general, tales técnicas de diseño permiten mantener un mayor grado de seguridad y confiabilidad en los mismos, puesto que garantizan la detección de un gran número de errores, que de otra forma únicamente serán detectados cuando los diseños se encontraran desarrollando su trabajo. El uso de tales técnicas se justifica principalmente en los casos en que se requiera un prototipo de diseño para producciones en masa; de esta forma se estaría completamente seguro del producto.

Bibliografía

[1] "Programable Logic Devices Databook and Design Guide".National Semiconductor 1993.

[2] Aquilino Cervantes A. "Programación de Dispositivos Programables Lógicos", Proyecto de Tesis para Ingeniero en Comunicaciones y Electrónica. Julio 1995.

[3] Di Giacomo. "Designing With High Performance ASIC's". Prentice Hall 1991.

[4] David Pellerin/Michael Holley. "Practical Design Using Programable Logic". Prentice Hall 1991.