SYMSIM: UN SIMULADOR PARA EL ESTUDIO DE CIRCUITOS DIGITALES A NIVEL DE TRANSISTOR

Ll. Ribas y J. Carrabina
Dept. Informàtica,
Universitat Autònoma de Barcelona, UAB.
08193 Bellaterra, Barcelona.
Tel. (93) 581 10 78
Fax. (93) 581 30 33
eMail. Ribas@cnm.es

RESUMEN.- La complejidad de los circuitos digitales actuales y de los procesos de su simulación puede hacerse más comprensible mediante un programa de ayuda al análisis simbólico de circuitos basado en la simulación de éstos. En particular, se presenta un simulador dirigido por eventos capaz de procesar una gran gama de circuitos digitales descritos a nivel de transistor. Este simulador no emplea, a diferencia de los más habituales, valores discretos como contenidos de los nodos de los circuitos, sino que éstos son representados por funciones lógicas. Esto, además de permitir una mayor comprensión de aquello que sucede en un circuito, permite sentar las bases de la realización de la verificación automática o de la generación de test.

1.- INTRODUCCIÓN

La verificación y el test de circuitos digitales se basan en principios similares. Ambos procesos buscan las diferencias que se pueden observar entre la especificación del circuito y su realización a nivel de diseño o en fabricación. En particular, la verificación pretende comprobar si un diseño cumple con su especificación. De no ser así, cabe la posibilidad de detectar cual ha sido la fuente del error. En el test de un circuito se buscan posibles fallos que hayan ocurrido durante su fabricación. Este proceso topa con la limitación que supone el no poder observar todos los nodos de un circuito, y tener que deducir dónde se ha producido el fallo o fallos a partir de las salidas del mismo [1].

Aunque existen técnicas especiales tanto para la verificación como para el test, ambos problemas recurren también a la simulación para resolver algunos de sus aspectos particulares. Concretamente, en verificación es necesario obtener la funcionalidad del circuito analizado para poder compararla con la de su especificación. En cuanto al test, la generación de vectores de test se complementa habitualmente con una simulación de fallos que, además de validar los vectores generados y su cobertura, puede contribuir a la reducción de las secuencias de test y/o a la generación de otros test complementarios.

El uso de simuladores lógicos simplifica las tareas de verificación, pero los resultados que se obtienen con ellos son difíciles de tratar, es decir, exigen exámenes visuales de las formas de onda de las salidas o bien complejos métodos de comparación automática.

Estos mismos simuladores, adaptados al proceso paralelo, concurrente o iterativo, permiten la validación de patrones de test, especialmente para un modelo de fallos de tipo *stuck-at* (se suponen líneas del circuito fijadas a 1 ó a 0).

Aun así, la enumeración de todos los posibles vectores de estímulos en un circuito para una simulación completa en cualquiera de los dos ámbitos mencionados se encuentra con el problema de la explosión combinatoria que dificulta su proceso. Así pues, se impone el uso de técnicas que reduzcan el tamaño de estas pruebas. Una de las técnicas de enumeración implícita más habitual en sistemas digitales es la del empleo de fórmulas booleanas o lógicas [2, 3], con las que se pueden representar conjuntos de valores binarios que las satisfacen (el resultado de la sustitución de estos valores en las variables que los representan es 1).

El empleo de fórmulas booleanas también topa con el problema de la comparación entre ellas, pues dos fórmulas distintas pueden representar la misma función. Por ello es necesario escoger una representación canónica de funciones como la que proporcionan los diagramas de decisión binarios ordenados o OBDDs [2]. De esta manera, es posible realizar comparaciones entre funciones.

Habitualmente, la mayoría de circuitos se realizan en lógica complementaria. Es decir, cuando se establece un camino entre una de sus salidas y Vdd (1 lógico), todos los caminos que le pueden conectar a Vss (0 lógico) están cortados eléctricamente. Por ello, una única función podría representar el comportamiento lógico en estos nodos.

Contemplar otros estilos de diseño en lógicas no complementarias (con precarga o señales de reloj) en la que un nodo puede aislarse eléctricamente de su entorno o estar en un camino resistivo entre Vss y Vdd, requiere separar aquella función (*on-set*) que representa todos los casos en que un nodo puede ser llevado a un 1 lógico de aquélla (*off-set*) que representa los casos en que es llevado a un 0 lógico [4]. Por tanto, la función que se observa en un nodo se caracteriza por dos funciones lógicas que representan su *on* y *off-set*, respectivamente:

$$f_n = (f^1, f^0)_n$$

Esta separación permite, además, analizar circuitos incorrectos en los que se presenten memorizaciones o cortocircuitos no deseados. Concretamente, éstos últimos se darán en las condiciones (asignaciones concretas a variables lógicas) en las que ambas funciones de *on* y *off-set* sean ciertas a la vez. Los estados de memorización se presentan cuando el nodo no puede ser conducido a 0 ó 1 lógico, es decir, en los casos en que la función de *on-set* y la función que representa su *off-set* son cero:

$$\text{cortocircuitado: } f^X = f^1 \cdot f^0$$
$$\text{aislado: } f^Z = \overline{f^1} \cdot \overline{f^0}$$

En los siguientes apartados, se ofrece una visión de las características del programa SymSim y, con ellas, las áreas en donde su aplicación puede servir de ayuda. El primer apartado se dedica a explicar brevemente cómo se efectúa una simulación dirigida por eventos y los elementos que intervienen en ésta cuando se procesan circuitos a nivel de transistor de forma simbólica. Los siguientes apartados comentan los posibles usos de SymSim, que se muestran esquemáticamente en la Figura 1. En resumen, las aplicaciones cubren el análisis de circuitos, su verificación a nivel funcional y la generación de patrones de test, que debe ser tomada como complementaria a una generación de test por métodos dedicados. En las conclusiones se destacan los aspectos más interesantes del programa y la ayuda que puede ofrecer a los interesados en estudiar circuitos digitales a nivel de transistor.

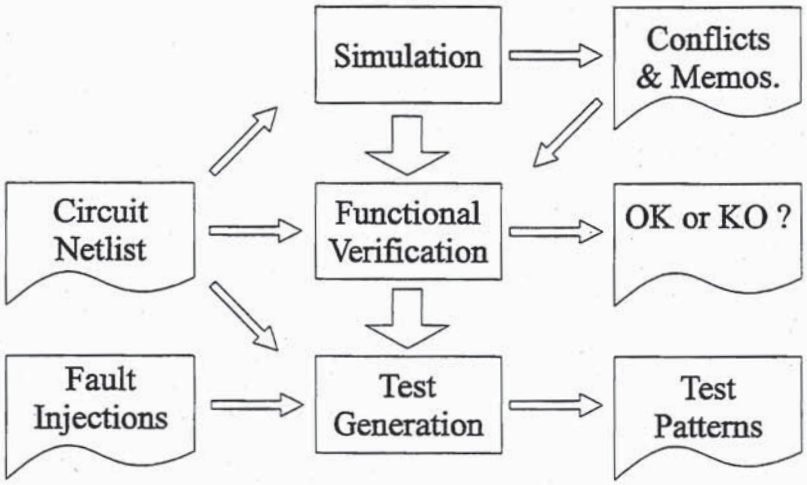

Figura 1.- Procesos en los que un simulador simbólico puede intervenir.

2.- SIMULACIÓN DIRIGIDA POR EVENTOS

En un simulador simbólico, las entradas (y los estados) de los circuitos digitales se representan con variables lógicas. El cambio de contenido de un nodo, como el que sucede cuando se asigna a una entrada una variable, afecta sólo a los dispositivos que están conectados al mismo. Éstos pueden cambiar su respuesta a consecuencia de ello y producir, a su vez, nuevos cambios de contenidos en los nodos de sus salidas; es decir nuevos eventos. Esta manera de proceder evita tener que calcular el contenido de todos los nodos de un circuito cuando sólo deben cambiar unos cuantos [3, 5, 6].

En SymSim, el contenido de los nodos consiste en dos funciones booleanas que representan su *on-set* y su *off-set*. Así pues, las entradas y los nodos de estado contienen una variable en su *on-set* y su complemento en el *off-set*, los cuales son asignados a través de las fuentes de tensión que existan en la descripción SPICE del circuito.

Los transistores serán los encargados de transformar estas literales en expresiones posiblemente más complejas. En este caso, los transistores se comportan como conmutadores ideales cuyo modelo se presenta en la siguiente expresión:

$$f_{drenador} = f_{drenador} + f_{puerta}^{B} \times f_{fuente}$$

en la que drenador y fuente deben intercambiarse para obtener la nueva función del terminal fuente. Además, B es 1 en caso de un transistor NMOS y 0 para un PMOS; es decir, el primero entra en conducción en las condiciones señaladas por el *on-set* de su puerta mientras que el segundo lo hace según el *off-set* de su puerta. Cabe insistir en que las funciones de los nodos drenador y fuente son, de hecho, un vector de funciones lógicas.

Todo el proceso de simulación se hace evidente cuando se observa la cola de eventos que se genera y los contenidos de los nodos afectados. Aun así, cabe hacer notar que la visualización de la cola de eventos sólo resulta útil para circuitos pequeños, ya que la mayor complejidad de otros causa un volcado de información demasiado grande.

3.- ANÁLISIS DE CIRCUITOS

El resultado de una simulación de este tipo permite ver qué tipo de función realiza un circuito o un subcircuito dentro de un circuito mayor. Es posible comprobar cómo los circuitos secuenciales presentan memorizaciones de valores (las funciones de *on-set* y *off-set* pueden ser cero a la vez) y que, asignando variables de estado a los nodos de memorización, las salidas dependen del contenido de nodos internos (los estados).
También se pueden detectar fácilmente los conflictos entre valores lógicos que puedan suceder en algún nodo, ya que el *on-set* y el *off-set* del nodo presentarán unas condiciones de cortocircuito no nulas. Cabe tener en cuenta que esta situación se da en todas las salidas de las etapas con precarga como, por ejemplo, en las salidas de las celdas diseñadas con lógica pseudo-nMOS o CMOS dinámica (con relojes) [7]. Así pues, es necesario indicar aquellos conflictos que están planificados y se resuelven en favor de un valor lógico determinado.
Una vez determinados aquellos nodos que presentan memorizaciones o cortocircuitos (en fase de precarga) válidos, como los que se observan en las salidas de *flip-flops* o de etapas de lógica dominó [7], se pueden detectar todos aquellos errores de diseño que produzcan otras memorizaciones y cortocircuitos no previstos.
Por tanto, el análisis de los resultados de una simulación simbólica ofrece una visión detallada del funcionamiento interno de un circuito. Además, después de una correcta identificación de aquellos nodos que presentan memorizaciones y cortocircuitos previstos, se pueden detectar errores de diseño.

4.- VERIFICACIÓN Y TEST

La verificación funcional de un circuito puede realizarse analizando las expresiones lógicas de cada nodo de salida, o bien proporcionando sus especificaciones directamente a SymSim, en cuyo caso comprobará que éstas coincidan con el resultado de la simulación en los nodos correspondientes.
En el programa se permite la simulación de varios circuitos en secuencia. De esta manera, y suponiendo que el primer circuito se corresponde con el circuito correcto y los demás son circuitos con fallos, es posible generar vectores de test. Éstos se construyen mediante diferencias lógicas (operaciones XOR) entre la función correcta y la funciones en los circuitos con fallos [1, 8, 9]:

$$f_{test}(\vec{x}) = f_{good}(\vec{x}) \oplus f_{i-fault}(\vec{x})$$

Dado que se dispone de información sobre los posibles cortocircuitos presentes en un circuito, también pueden generarse directamente vectores para test de Iddq [10] (medidas de corriente de alimentación en reposo):

$$f_{test-IDDQ}(\vec{x}) = f^{X}_{i-fault}(\vec{x})$$

En el programa se incluye una opción para reducir el número de vectores de test obtenidos. Se aplica un algoritmo basado en una técnica *branch & bound* para generar el conjunto mínimo de funciones de test que den cobertura a todas aquellas funciones que fueron generadas en la etapa de simulación de fallos. Este proceso no se realiza de manera simultánea al proceso de obtención de vectores de test porque esta generación se concibe como un complemento a las herramientas de generación de test y, por tanto, para una cantidad pequeña de fallos [9].

5.- CONCLUSIÓN

La plena comprensión de aquello que sucede en un circuito y de los procesos que deben seguirse para su validación tanto en su diseño como cuando ya está fabricado contribuye positivamente a una mejor concepción de los circuitos y favorece el uso adecuado de las necesarias herramientas de CAD.

El simulador simbólico SymSim puede ayudar en este proceso en tanto que se pueden analizar los circuitos a un nivel más comprensible desde el punto de vista humano. Además, la posibilidad de observar cómo evoluciona una simulación a través del seguimiento de la cola de eventos que ésta produce permite entender mejor los mecanismos de funcionamiento de este tipo de simuladores. También hace posible a sus usuarios adentrarse en los campos de la verificación automática de circuitos y de su test, tanto en la vertiente de la simulación de fallos como en la de la generación de patrones de test.

A pesar de ello, conviene recordar que no se trata de un programa capaz de trabajar con circuitos muy complejos debido a la limitación que supone el uso de BDDs para la representación de las funciones de algunos de ellos. Y, por tanto, no puede sustituir a otras herramientas disponibles comercialmente.

6.- REFERENCIAS

[1] Abradomici, M., Breuer, M.A. y Friedman, A.D. "Digital Systems Testing and Testable Design", Ed. W.H. Freeman and Co., 1990.

[2] Bryant, R.E. "Graph Based Algorithms for Boolean Function Manipulation", *IEEE Transactions on Computers*, Vol. C-35, n° 8, 1986.

[3] Bryant, R.E., Beatty, D., Brace, K., Cho, K. y Sheffler, T. "COSMOS: A Compiled Simulator for MOS Circuits", *Proceedings of the 24th ACM/IEEE DAC*, pp. 9-16, 1987.

[4] Ribas, Ll. y Carrabina, J. "Analysis of Switch-Level Faults by Symbolic Simulation", *Proceedings of the 32th ACM/IEEE DAC*, pp. 352-357, 1995.

[5] Lamb, P.R. "Object-oriented Techniques for Mixed-Mode Circuit Simulation", *Series in Microelectronics*, Vol. 11, Ed. Hartung-Gorre, 1991.

[6] French, R.S., Lam, M.S., Levitt, J.R. y Olukotun, K. "A General Method for Compiling Event-Driven Simulations", *Proceedings of the 32th ACM/IEEE DAC*, pp. 151-156, 1995.

[7] Weste, N.H.E. y Eshraghian, K. "Principles of CMOS VLSI Design: A Systems Perspective", Ed. Addison-Wesley, 1994.

[8] Cho, K. y Bryant, R.E. "Test Pattern Generation for Sequential MOS Circuits by Symbolic Fault Simulation", *Proceedings of the 26th ACM/IEEE DAC*, pp. 418-423, 1989.

[9] Bolchini, C., Fummi, F., Gemelli, R. y Salice, F. "A BDD Based Algorithm for Detecting Difficult Faults", *Proceedings of ISCAS*, pp. 2015-2018, 1995.

[10] Nigh, P. y Maly, W. "Test Generation for Current Testing", IEEE Design and Test of Computers, Feb., 1990.