CBLT PARA EL ALGORITMO D¡Error! Marcador no definido.

M. J. LLORIS, A. LLORIS[1]
Departamento de Electrónica y Tecnología de Computadores. Facultad de Ciencias. Universidad de Granada. Campus de Fuentenueva. 18014. Granada.

En esta comunicación se presenta un CBLT (Computer Based Learning Tool) para el aprendizaje del algoritmo D, que constituye una buena iniciación a los algoritmos deterministas para el test de circuitos digitales combinacionales. El CBLT desarrollado consta de dos programas: el primero obtiene vectores de test para un circuito predefinido en el propio programa, detallando en la pantalla todos los pasos que se siguen; el segundo obtiene vectores de test para circuitos definidos por el usuario. El CBLT se ha utilizado en las prácticas de asignaturas de grado y se ha mostrado como una herramienta eficaz.

1. Introducción

El desarrollo de los Circuitos Integrados (CI) a un ritmo tan rápido y continuado como el que se viene manteniendo desde el nacimiento de la microelectrónica plantea problemas de creciente complejidad, destacando entre ellos los relacionados con el test de los productos finales [1]. De hecho el coste del test es el componente más importante del coste total en el desarrollo y fabricación de los CIs, aproximandose en algunos casos concretos de circuitos digitales al 50 % del coste final de cada chip [2]. Esto debe tener su repercusión a nivel docente e implica que en las titulaciones relacionadas con la Electrónica se debe dedicar cada vez más atención al test de los CIs, bien con asignaturas específicas, bien como bloques relevantes de asignaturas mas generales.

Refiriendonos concretamente a los Circuitos Digitales, el algoritmo D es de los primeros ATPGs (Automatic Test Pattern Generator) deterministas para circuitos combinacionales que aparecieron [3], y a su vez ha sido germen de otros más eficientes, como el PODEM [4] y al FAN [5], y se ha extendido su aplicación a circuitos secuenciales [6]. Además, por razones didácticas, conocer en detalle el algoritmo D es casi imprescindible, o al menos puede ser una buena iniciación, para entender los diversos aspectos del test de los circuitos digitales.

2. El algoritmo D

El algoritmo D permite obtener patrones de test para circuitos combinacionales no redundantes a

[1] Este artículo ha sido parcialmente financiado por la Dirección General de Enseñanza Superior a través del proyecto PB98-1354.

nivel de puertas, utilizando como modelo de faltas las de anclaje simple. Dada una falta para la que se quiera desarrollar un test, el algoritmo D garantiza obtener un patrón de test si dicho patrón existe [7]. Es un algoritmo determinista y se basa en el principio de sensibilización de un camino [8], modelando los comportamientos del circuito correcto y cuando tiene una determinada falta de anclaje simple en cualquier punto del circuito, para la que se quiere obtener un vector de test.

La primera acción al aplicar el algoritmo es sensibilizar la falta para la que se está obteniendo el vector de test, obligando con las entradas externas a que el punto en cuestión asuma, en el circuito correcto, un valor diferente del que corresponda a la falta; esta discrepancia se denota con D (de ahí la denominación del algoritmo). Es decir, lo primero es generar el valor D en el punto con falta. A continuación el algoritmo consta, básicamente, de dos fases: "propagación" hacia la salida de los efectos de la falta, y "justificación" hacia las entradas de las condiciones generadas en la fase de propagación.

3. CBLT desarrollado

La herramienta CBLT desarrollada para el aprendizaje del algoritmo D consta de dos programas diferentes, uno denominado "Circuito ejemplo" y otro llamado "Generador D", que se detallan a continuación.

Aunque los programas no exigen ajuste previo alguno, sí lo admiten. De hecho se presupone la acción de un tutor que, en cada caso, sintonizará los detalles de su ejecución, fijando los parámetros a las necesidades de cada estudiante o de cada grupo.

3.1. Circuito ejemplo

Este programa introductorio utiliza un circuito residente en el propio programa; es decir, la descripción del circuito está "cableada" en el código. Con referencia a este circuito, el programa muestra detalladamente los pasos que ejecuta, y los correspondientes valores en los diferentes puntos del circuito, para obtener el vector de test para una falta previamente seleccionada por el usuario. Concretamente empieza "sensibilizando" la falta, asignando al punto anclado el valor complementario al de anclaje; a continuación "propaga" la falta, hasta la salida primaria; por último viene la fase de "justificación", llevando hasta las entradas primarias el efecto de las acciones anteriores, detectando posibles conflictos y seleccionando las correspondientes alternativas para aplicar de nuevo las fases anteriores. Finalmente proporciona un vector de test, si es que lo hay.

Es obvio que el circuito que se seleccione como ejemplo debe ser sencillo. El que está incluido por defecto es el de la Figura 1, pero el tutor puede introducir, en cada caso, cualquier otro que considere más oportuno.

Al ejecutar el programa se muestra al usuario el circuito ejemplo y se le pide que seleccione un punto de anclaje y un valor de anclaje. A continuación muestra los resultados de las diferentes etapas, es decir, los valores a los que se ponen los diferentes nodos del circuito, siendo el usuario

quien decide el paso de una etapa a la siguiente. Por supuesto, el usuario puede testear tantas faltas del circuito ejemplo como desee.

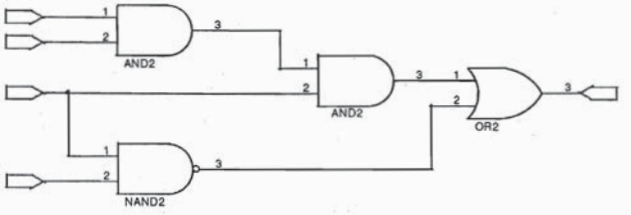

Figura 1: *Circuito ejemplo*

3.2. Generador D

Este segundo programa permite aplicar el algoritmo D a cualquier circuito que defina el usuario, siempre dentro de las limitaciones en cuanto a tamaño que imponga el sistema en el que se ejecute el programa. Dado su carácter de herramienta de aprendizaje, se ha preferido admitir sólo circuitos con una sola salida, aunque la extensión a circuitos multisalida es sencilla.

La primera acción al utilizar el generador D es, lógicamente, introducir la descripción del circuito combinacional a testear mediante un diálogo sencillo contestando menús que secuencialmente aparecen en la pantalla. Los primeros datos para definir un circuito son el número de entradas externas y el número de nodos (o puertas). Las entradas se numeran consecutivamente de 1 a n, y los nodos internos del n+1 en adelante, correspondiendo el valor más alto a la puerta de salida. Para cada nodo se ha de dar el tipo de puerta del mismo y los nodos precedentes (incluidas las entradas primarias) que actúan como entradas a esa puerta.

Tras definir el circuito, el programa construye la matriz predecesora (MP), en la que cada fila corresponde biunívocamente a un nodo del circuito, dando en ella el tipo de puerta y las entradas de ese nodo. También construye la matriz sucesora (MS), en la que, de nuevo, cada fila corresponde biunívocamente a un nodo, dando todos los nodos a los que alimenta (sus sucesores). Estas dos matrices se utilizarán ampliamente en los procesos de propagación y justificación del algoritmo D.

A continuación el programa pide que se seleccione una falta a testear (es decir, que se seleccione el nodo y el tipo de falta), y aplica el algoritmo reflejado en el organigrama simplificado de la Figura 2. En este caso en la pantalla aparecen menos detalles que con el programa "Circuito ejemplo".

Tras obtener un vector de test para la falta seleccionada (o determinar que la falta es redundante), el programa ofrece la posibilidad de seleccionar otra falta, o de intruducir otro circuito, o de acabar la ejecución del programa.

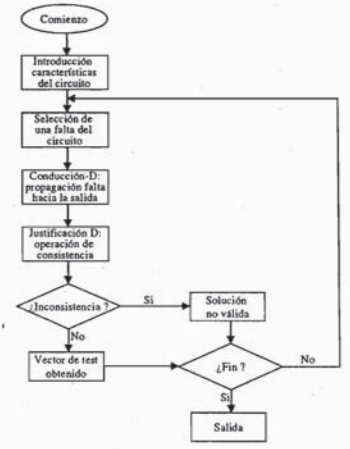

Figura 2. *Organigrama del algoritmo D*

4. Conclusiones

En esta comunicación se ha descrito a grandes rasgos una herramienta pedagógica para la iniciación al test de los circuitos digitales. El CBLT está desarrollado en C++ y se utiliza bajo Windows. Se ha empezado a utilizar en las prácticas de asignaturas de grado en las que se aborda el test de los circuitos digitales, y los resultados hasta el momento son plenamente satisfactorios, siendo de gran ayuda para que los estudiantes capten rápidamente los diferentes aspectos implicados en la aplicación del algoritmo D.

Referencias

[1] *The 1997 National Technology Roadmap for Semiconductors*, Semiconductor Industry Assoc., San José, Calif., 1997.

[2] Sachdev, Manoj: *Defect Oriented Testing for CMOS Analog and Digital Circuits*, Kluwer Academic Publishers, 1998.

[3] Roth, J.P.: *Diagnosis of Automata Failures: A Calculus and a Method*, IBM Journal of Research and Development, vol. 10, n°4, pp. 278-291, July 1966.

[4] Goel, P.: *An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits*, IEEE Trans. on Comp., vol. C-30, n°. 3, pp. 215-222, March 1981.

[5] Fujiwara, H.; Shimono, T.: *On the Acceleration of Tests Generation Algorithms,* IEEE Trans. on Comp., vol. C-32, n°. 12, pp. 1137-1144, December 1983.

[6] Putzolu, G.; Roth, J.P.: *A Heuristic Algorithm for the Testing of Asynchronous Circuits*, IEEE Trans. on Comp., vol. C-20, n°. 6, pp. 639-647, June 1971.

[7] Abramovici, M.; Breuer, M. A.; Friedman, A. D.: *Digital Systems Testing and Testable Design*, IEEE Press, 1995.

[8] A. Miczo: *Digital Logic Testing and Simulation*, John Wiley & Sons, 1987.