Implementación de máquinas de estados basadas en rom en dispositivos FPGA de XILINX de bajo costo

State Machine Implementation Rom-based in XILINX Low Cost FPGA Devices'

Recibido: 9 de Mayo de 2013
Aprobado: 20 de mayo de 2013

Miguel Pérez Pereira*, Edwar Jacinto Gómez** y Mario Fernando Robayo Restrepo***

Resumen

El objetivo del presente trabajo es describir una técnica para la realización de máquinas de estados usando una memoria ROM para la descripción de la lógica de estado siguiente y lógica de salida, dicho bloque ROM es un bloque funcional dentro de una FPGA de Xilinx, aprovechando que estos son nativos para la mayoría de dispositivos programables de esta compañía.

Palabras clave

Máquinas de estados, circuitos digitales, arreglos de compuertas programables en campo.

Abstract

The objective of this paper is to describe a technique for performing state machines using a ROM for the description of the next state logic and output logic, the block ROM is a functional block in a Xilinx FPGA, taking advantage of these are native to most programmable devices of this company.

* M. Sc. (c). Electrónica. Ingeniero en control electrónico e instrumentación. Profesor de planta de la Facultad Tecnológica de la Universidad Distrital Francisco José de Caldas. Parte del Semillero «Sistemas Digitales Inteligentes DIGITI». E-mail: mperez@udistrital.edu.co

** M. Sc. (c). Ciencias de la Información y las Comunicaciones. Ingeniero en control electrónico e instrumentación. Profesor de planta de la Facultad Tecnológica de la Universidad Distrital Francisco José de Caldas. Parte del Semillero «Sistemas Digitales Inteligentes DIGITI». E-mail: ejacintog@udistrital.edu.co

*** Estudiante de pregrado, Ingeniería en Control Electrónico, Tecnólogo en electrónica, Universidad Distrital Francisco José de Caldas. Parte del Semillero «Sistemas Digitales Inteligentes DIGITI» mrobayo@udistrital.edu.co

Keywords

State machine, digital circuits, field programmable gate arrays.

I. Introducción

Existen dos limitantes básicas cuando se realizan diseños en dispositivos de lógicos programables específicamente en FPGA, la primera es la cantidad de hardware utilizado en la aplicación y la otra es la frecuencia máxima a la cual puede ser usado el diseño, por dicha razón se debe buscar usar la menor cantidad de recursos y además de ello verificar que la implementación tenga un retardo de propagación tolerable.

Las máquinas de estados finitos (siglas en inglés FSM [1], se utilizan para describir cualquier tipo de sistema secuencial como control de otros bloques digitales o simplemente con descripción de muchas otras funciones digitales. La descripción de estas, en un lenguaje de alto nivel se realiza de una forma sencilla, pero en ciertas ocasiones dicha descripción utiliza bastantes recursos del dispositivo, y además el tiempo de respuesta dado por los retardos de propagación es difícil de controlar. De aquí la necesidad de aprender a describirlas de formas diferentes buscando el diseño más eficiente. [2-4].

Debido a esto, se plantean un conjunto de ideas obtenidas a partir de la compilación de una serie de trabajos en diseño digital avanzado, que pueden dar una perspectiva al diseñador de cómo enfocar sus esfuerzos para la optimización de diseños en VHDL. Específicamente se describe cómo realizar una máquina de estados de tamaño significativo usando una arquitectura que toma como base el uso de un bloque ROM de la FPGA.

II. Máquina de estados finitos

Una máquina de estados finitos es un sistema sincrónico (gobernado por un reloj), el cual tiene un número fijo de estados, una cantidad de transiciones o saltos entre estados, los cuales son gobernados por unas entradas que pueden o no influir directamente en las salidas del sistema [5], estas salidas pueden ser totalmente combinatorias figura 1 (a) o pueden tener un registro a la salida para evitar instantes de tiempo inesperados «glitch» [1]figura 1 (b),

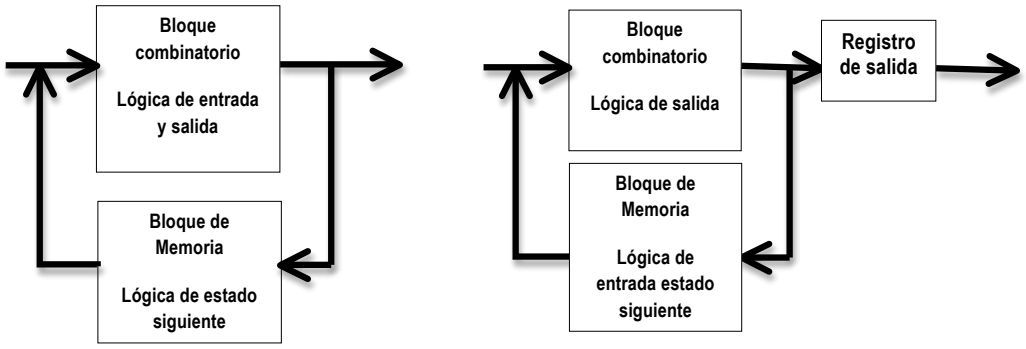

Fig.1. (a). Bloques básicas de una máquina de estados, (b) Bloques básicos con registro de salida para evitar «Glitch»

III. Caracterización de las máquinas de estados

Cuando se diseña una máquina de estado se pueden tener en cuenta varios factores, entre los cuales están la cantidad de estados, la geometría de la máquina como tal y la codificación de los estados, seguramente se lograrían mejores resultados en cuanto a la optimización del hardware, si el diseñador se detuviera un instante a verificar el tipo de máquina de estado la cual quiere implementar. A continuación se muestran algunas caracterizaciones de las máquinas de estados logradas a través de la práctica. [6-9], estas caracterizaciones no se encuentran en ninguna literatura formal, y son un posible objeto de estudio de un trabajo posterior.

3.1 Caracterización en dependencia de su geometría

3.1.1 Lineal: en este tipo de máquinas los estados están geométricamente alineados uno después del otro, son ideales para procesos de comunicaciones, en que el reinicio depende de una señal externa. En la figura 2, se muestra un diagrama ejemplo de una máquina lineal. [1,8]

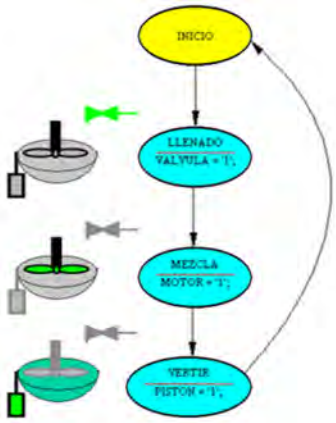

Fig.2. Máquina de estados lineal

3.1.2 Multicolumna: son máquinas de estados donde existe un punto que se bifurca dependiendo de las señales de entrada y cada brazo retorna a un estado inicial. Es usado típicamente para las unidades de control de procesos complejos.

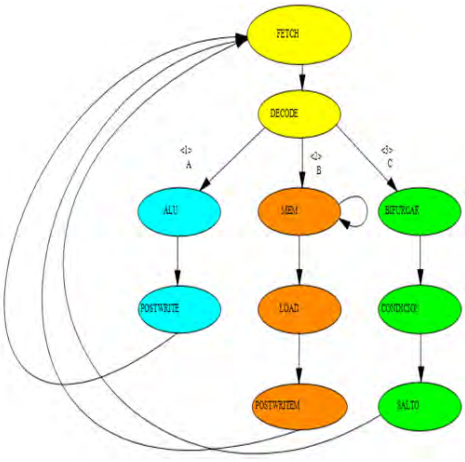

Fig.3. Máquina de estados multicolumna para la unidad de control de un procesador

3.1.3 Multicolumna bidireccional: este tipo de máquinas de estado se diferencian de las anteriores porque estas retornan al estado inicial por el mismo brazo. Son utilizadas en secuencias con un punto común.

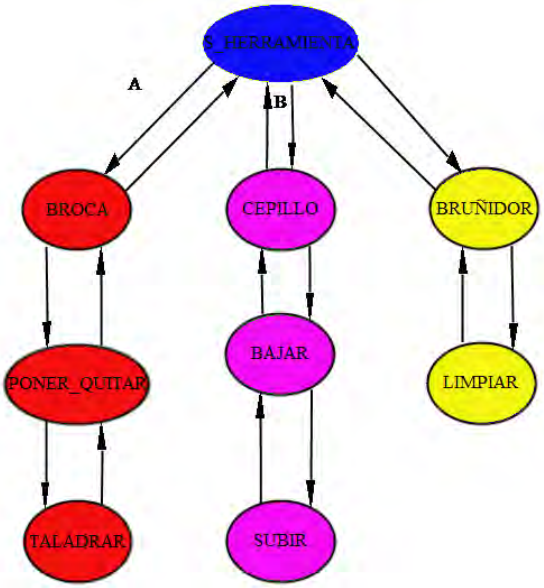

Fig.4. Máquina de estados multicolumna bidireccional para el control de un robot en una carpintería

3.1.4 Caótica: las máquinas de estado caóticas no tienen una forma geométrica definida.

3.1.5 Concéntrica: el ejemplo típico de estas máquinas es el sistema de control de luces de un semáforo vista en la figura 5, secuenciadores o controles de motores de paso. Se diferencia de las lineales en que estas últimas son mucho más largas y no necesariamente son cíclicas.

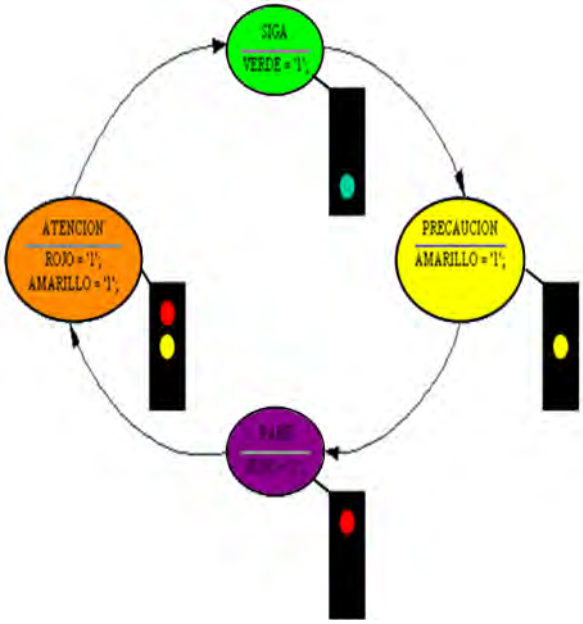

Fig.5.Máquina de estados concéntrica

3.2 Caracterización en dependencia de su codificación de estados

Codificación de estados hecha automáticamente por las herramientas CAD:

- → **One Hot:** codificación de estados donde hay solamente un uno por código y se utilizan tantos flip-flops como estados existan.
- → **Mínimo cambio de Bit (código Grey):** codificación de estados donde solamente hay un cambio de bit entre códigos adyacentes.
- → **Conteo enumerado (binario):** codificación de estados que utiliza la numeración binaria como asignación de códigos.

Codificación de estados realizada directamente por el diseñador:

- → **Aleatorio (Random):** codificación de estados sin orden aparente.
- → **Two Hot:** codificación de estados donde hay dos unos por código. También utiliza tantos Flip-Flops como estados existan.
- → **Igual a las salidas:** codificación de estados donde el código del estado es igual a la salida de la máquina de estados.

IV. Formas de implementación de las máquinas de estados

Como primera medida si dentro de una aplicación existe una máquina de estados finitos muy compleja se debe contemplar la posibilidad de dividirla en varias máquinas más pequeñas, con funciones mucho más simples. En este caso las herramientas CAD soportan el diseño de más de un diagrama de estados y su posterior unión en una sola entidad final. [1,10-11].

Una máquina de estados se puede desarrollar en una de las siguientes formas según la descripción en alto nivel que de ella se haga, específicamente en este caso se da mediante un estilo de diseño donde se ve una única entidad y se dividen cada una de sus partes internas en procesos independientes y paralelos. Para este caso es bueno recordar que cuando se utiliza este estilo de descripción de hardware, el diseñador debe crear tantas señales como sean necesarias para concatenar cada uno de los procesos o partes del diseño implícitas, y que estas señales deben estar nombradas en las listas de sensibilidad indicadas, para que el proceso que describe cada una de las partes de la máquina de estados cambie cuando sea el momento indicado. Nombrando como ejemplo un caso específico como lo es la lógica de salida de la máquina, si dentro de la lista de sensibilidad, se nombra el reloj y dentro de la descripción se hace referencia a un flanco del mismo, cada una de las salidas de este proceso será asignada a un Flip-Flop. [12-14].

4.1. Pasos para la implementación de una máquina de estados en general

A partir de la caracterización, se plantea la siguiente metodología de diseño de máquinas de estados con la cual se podría lograr reducción de hardware, todo esto se realizó sobre los parámetros de la metodología Top-Down:

- Diseño del diagrama de estados. En este paso el diseñador, debe dibujar el diagrama de estados examinando si hay estados repetidos, o señales que se comporten igual utilizando solo una de ellas, o si la máquina es de mucha complejidad y lo apropiado es subdividirla.
- En seguida el diseñador observando el diagrama de estados, puede determinar su tamaño, forma y número de estados y salidas como principales características. En este paso es donde el usuario debe determinar si vale la pena realizar la descripción por código o simplemente, se debe utilizar una herramienta CAD que arroje una descripción estándar. También las herramientas poseen estándar.

El diseñador tomando en cuenta las características de la máquina de estados y el reporte de la herramienta CAD usada debe tener en cuenta las siguientes recomendaciones:

a. Ocupa muchos Flip-flops: debería tratar de codificar los estados por código, o contemplar la posibilidad de utilizar una ROM (como se muestra posteriormente).
b. Las salidas presentan Glitch: eL diseñador debería realizar la lógica de salida, donde la salida que presente problemas se encuentre en un proceso dependiente del reloj.
c. Ocupa muchas LUT o compuertas: El diseñador debería utilizar una codificación One-hotóTwo-hot.

Si el usuario considera que los resultados siguiendo los pasos anteriormente descritos no son satisfactorios o simplemente la máquina de estados descrita es considerada grande en cuanto a la cantidad de estados [14-16] se muestran un par de arquitecturas posibles en la figura 7 para este tipo de máquinas de estados, que son muy apropiadas cuando se trabaja con FPGA ya que se pueden implementar con lógica distribuida «LUTs» o con los Bloques RAM/ROM que posee cualquier dispositivo de Xilinx que fácilmente se puedan convertir en ROM [2] según la figura 6.

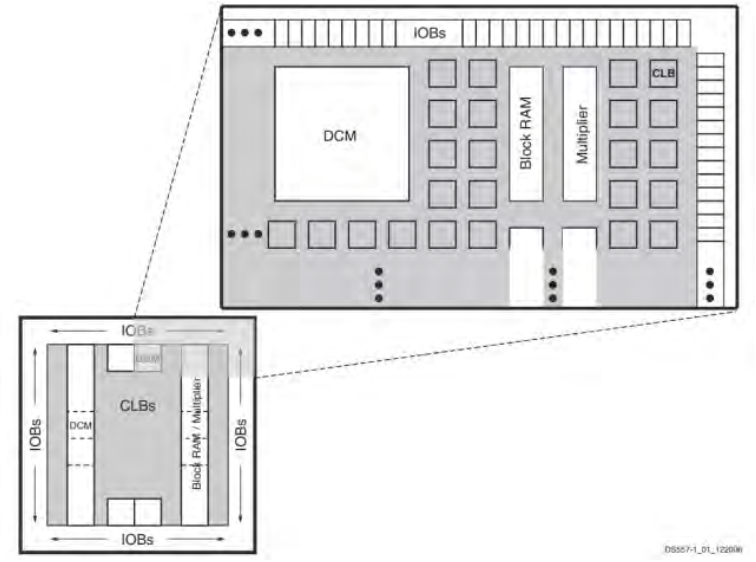

Fig. 6. Arquitectura de la familia Spartan-3AN [17]

Las máquinas de estados fueron implementadas en dispositivos de la familia Spartan 3E y Spartan 3AN y probados con los siguientes sistemas de desarrollo: Spartan-3A/3AN FPGA Starter Kit, Spartan-3E FPGA Starter Kit y Basys II, estas FPGA poseen entre 6 y 20 bloques RAM/ROM que funcionan a la máxima velocidad del dispositivo, lo que hace posible que las siguientes arquitecturas sean fácilmente realizables y que además tengan un excelente desempeño en cuanto a la velocidad de funcionamiento y los retardos de propagación.

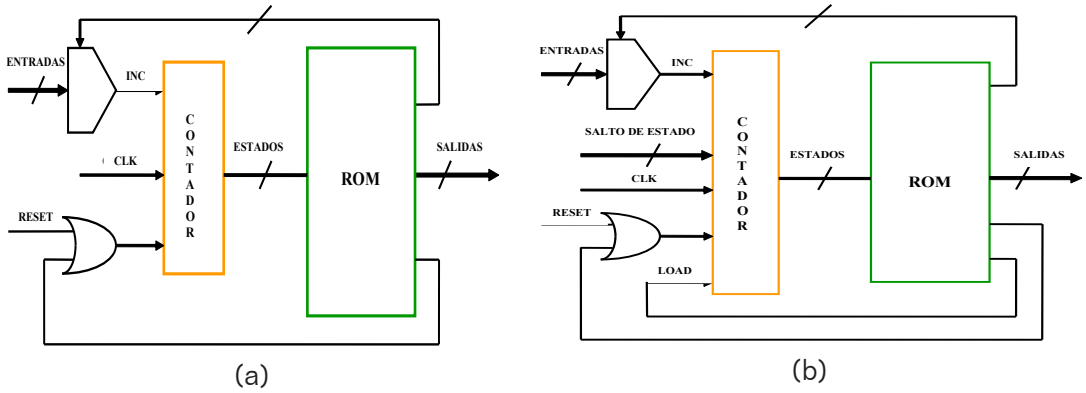

Fig. 7. (a)Máquina de estados lineal con ROM . 7(b)Máquina de Estados multicolumna con ROM «el contador tiene carga paralela»

V. Resultados de las implementaciones realizadas

Tomando como base la implementación de 50(cincuenta) máquinas de estados descritas en forma tradicional comparándolas con las mismas pero usando la arquitectura dada en la figura 7, se realizó la tabla 1 donde se tabulan los datos arrojados por el informe de la herramienta ISE de Xilinx para los primeros 10 casos, en dicha tabla se tomaron en cuenta datos de retardo por de propagación máximo y porcentaje usado del dispositivo para los dos casos.

Número de Ejemplo	Máximo retardo de propagación dado en ns		Porcentaje de reducción al usar ROM
	Normal	Usando ROM	
1	0,19	0,199	-4,70%
2	0,19	0,199	-4,70%
3	0,194	0,203	-4,60%
4	0,175	0,18	-2,90%
5	2,092	0,174	91,70%
6	3,339	0,197	94,10%
7	1,524	0,196	87,10%
8	1,155	0,196	83,00%
9	1,155	0,199	82,80%
10	0,553	0,197	64,40%
Promedio	1,1290909	0,19445455	52,30%

Tabla n.° 1. Comparación retardo de máquina normal y maquina en ROM

VI. Conclusiones

Se genera una clasificación de máquinas de estado por su forma y codificación, conocimiento que ofrece herramientas al diseñador para escoger la metodología de diseño más apropiada al momento de realizar implementaciones de sistemas digitales secuenciales.

Se planteó una metodología de diseño para máquinas de estados finitas usando lenguajes de descripción de hardware, dando diferentes opciones de implementación teniendo en cuenta las características de la máquina como tal, además de la arquitectura interna del dispositivo, dicha metodología disminuye notablemente el uso de los recursos de los diferentes PLDs. Se muestran unos resultados experimentales que no son conclusiones absolutas, sino más bien son la base para generar un pensamiento crítico en los diseñadores de hardware.

Se generó una caracterización de las máquinas de estados por su forma geométrica, su forma de descripción y el número de estados que ella misma posea, a partir de la cual se dan unas sugerencias de diseño que podrían producir una optimización del hardware, sin necesidad de que se experimente con los diferentes tipos de implementación.

Se desarrollaron diferentes estilos de programación en lenguajes de descripción de hardware, buscando la reducción del uso de recursos en diferentes diseños, todos estos soportados por el correcto uso de las herramientas CAD que ofrecen gran variedad de posibilidades de entradas de diseño.

Según los datos tabulados en la tabla 1, se evidencian ocasiones en las cuales las descripción de las distintas máquinas de estado usando bloques ROM no supone una reducción en la cantidad de hardware usado, dicha condición se da cuando la máquina de estados no posee la cantidad de estados suficiente para considerarse «grande», en dichos casos para lograr mejores resultados se recomienda retomar aspectos como el tipo de codificación de estado usado, la cantidad de salidas y si dichas salidas requieren de un registro para evitar «Glitch», y guiarse por las recomendaciones ofrecidas en el numeral 4.1 del presente trabajo.

En cuanto a la velocidad de propagación del dispositivo, se puede notar en la tabla 4.1que cuando la cantidad de estados aumenta el retardo de propagación aumenta, es decir, la frecuencia máxima de funcionamiento del diseño disminuye, condición que cuando se usa el bloque ROM como base de la implementación de la máquina de estado se mantiene casi constante, ya que sin importar la cantidad de estados que la máquina posea el retardo es el mismo.

Referencias

[1] Garcia, E.: «Xilinx: Creating Finite State Machines», *Xcell Journal*, 2000, 38.

[2] R. Senhadji-Navarro, I. García-Vargas, G. Jiménez-Moreno and A. Civit- Ballcels «ROM-based FSM implementation using input multiplexing», *Electronics Letters*, Vol. 40, n.º 20, September 2004.

[3] Rawski, M., Selvaraj, H., and Łuba, T.:«An Application of Functional, Decomposition in ROM-Based FSM Implementation in FPGA Devices», Proc. Euromicro Symposium on Digital System Design, 2003, Belek-Antalya (Turkey), pp. 104-110

[4] Valery, Sklyarov, «Synthesis and Implementation of RAM-Based Finite State Machines in FPGAs», Proc. Field Programmable Logic and its applications (FPL), 2000, pp. 718-727

[5] De Micheli, G., «Synthesis and Optimization of Digital Circuits», New York: McGraw-Hill, 1994

[6] Katz, R. H., «Contemporary Logic Design» (The Benjamin/Cummings Publishing Company, Inc., California: 1994).

[7] Krueger, R. «Xilinx Virtex Devices: Variable Input LUT Architecture», *The Syndicated*, Vol. 4, 2004, Issue I.

[8] Xilinx, Inc.: «Using Dedicated Multiplexers in Spartan-3 Generation FPGAs», XAPP466, Vol. 1.1, May 20, 2005.

[9] Xilinx, Inc.: *XST User Guide*, 2005.

[10] McElvain, K. *IWLS'93 Benchmark Set: Version 4.0*, 1993.

[11] Garey, M. R. and Johnson, D. S. *Computers and Intractability: A Guideto the Theory of NP-Completeness*. New York: W. H. Freeman, 1983.

[12] Burns, M., Perkowski, M., Jówiak, L. «An EfficientApproach to Decomposition of Multi-OutputBoolean Functions with Large Set of Bound Variables», Proc. of the EuromicroConference,Vasteras, 1998

[13] Brzozowski, I., Kos A., «Minimisation of Power Consumption in Digital Integrated Circuits by Reduction of Switching Activity», Proc. of the Euromicro Conference, pp. 376-380. Vol. 1, Milán, 1999

[14] Brzozowski J. A., Luba T., «Decomposition of Boolean Functions Specified by Cubes», Research Report CS-97-01, University of Waterloo, Waterloo; REVISED October 1998.

[15] Jozwiak L., Chojnacki A., «Functional Decomposition Based on Information Relationship Measures Extremely Effective and Efficient for Symmetric Functions», Proc of the Euromicro Conference, Vol. 1, 1999 pp.150-159, Milan.

[16] Kravets, V. N., Sakallah, K. A., «Constructive library-aware synthesis using symmetries», Proc. Of Design, Automation and Test in Europe Conference, 2000

[17] Spartan 3A/AN Family Datasheet, DS557 April 1, 2011

[18] Chang, S.C., Marek-Sadowska M., Hwang T.T.,«Technology Mapping for TLU FPGAs Based on Decomposition of Binary Decision Diagrams», *IEEE Trans. on CAD*, Vol. 15, 1996, n.° 10, pp. 1226-1236.

[19] De Micheli, G. *Synthesis and Optimization of Digital Circuits*, New York: McGraw-Hill, 1994.

[20] Hartmanis, J., Stearns, R.E., *Algebraic Structure Theory of Sequential Machines,* Prentice-Hall, 1966.