Realización en FPGA de Circuitos Secuenciales Síncronos a partir de su Diagrama de Estados

Realization on FPGA of Synchronous Sequential Circuits from its State Diagram.

Guillermo Tejada Muñoz[1], Steven Jesús Zarzosa Chávez[2], Víctor Benítez[3]

Facultad de Ingeniería Electrónica y Eléctrica, Universidad Nacional Mayor de San Marcos, Lima Perú

Resumen— El siguiente trabajo está dirigido a los estudiantes de ingeniería electrónica y de ramas afines con el objetivo que puedan construir circuitos lógicos secuenciales síncronos dentro de un FPGA, para ello solamente necesitan describir gráficamente el funcionamiento del circuito mediante su diagrama de estados sin la necesidad que conozcan su programación en VHDL. Como herramientas han sido utilizadas: el ISE WebPACK 10.1 y el sistema de desarrollo XUP Virtex-II Pro basado en el circuito integrado XC2VP30, el cual tiene inserto a un FPGA.

Abstract— The following paper is directed at students electronic engineering and related branches in order that they can build synchronous sequential logic circuits within an FPGA, for it only need to describe graphically the functioning of the circuit through its state diagram without the need to know programming in VHDL. As tools are used: the ISE WebPack 10.1 and the development system XUP Virtex-II Pro based on the XC2VP30 chip, which has inserted a FPGA.

Palabras claves— Circuitos secuenciales síncronos, diagrama de estados, FPGA, XUP Virtex-II Pro, ISE WebPACK 10.1, iMPACT, simulador ISE, ModelSim XE, VHDL.

Keywords— synchronous sequential logic circuits, state diagram, FPGA, XUP Virtex-II Pro, ISE WebPack 10.1, iMPACT, ISE simulator, ModelSim XE, VHDL.

I. INTRODUCCIÓN

La estructura general de un circuito secuencial síncrono, tal como se ilustra en la Fig. 1, ilustra a la memoria de estado como un conjunto de n flip flops que almacena el estado actual del circuito, y tiene 2^n estados distintos. Los flip flops se encuentran todos conectados a una señal de reloj común por lo que en sincronía con los flancos de subida o bajada cambian de estados todos a la vez. El estado siguiente del circuito está determinado por la lógica combinacional de estado siguiente y es función de las entradas y los estados actuales. La lógica combinacional de la salida determina la salida como una función de las entradas y los estados actuales, un circuito con estas características se denomina tipo Mealy. Sin embargo, en algunos circuitos secuenciales la salida depende solamente de los estados actuales, en este caso el circuito se denomina tipo Moore y es mostrado en la Fig. 2 [1].

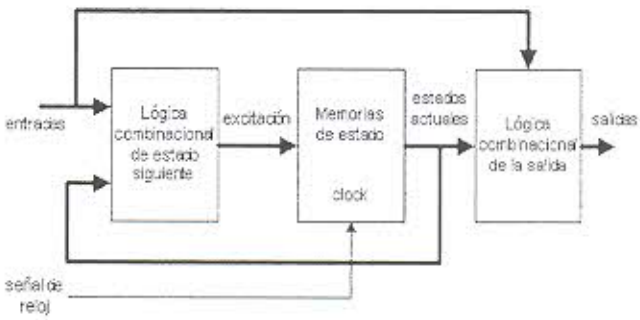


Fig. 1. Circuito secuencial síncrono tipo Mealy.

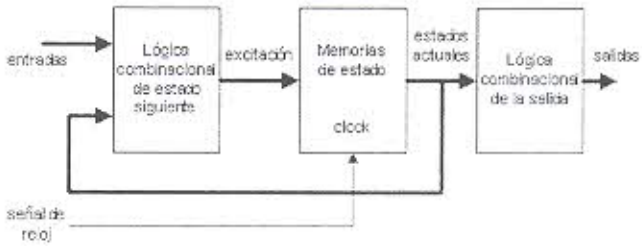


Fig. 2. Circuito secuencial síncrono tipo Moore

A continuación veamos cómo se procede a modelar y diseñar un circuito secuencial síncrono de una

[1] Guillermo Tejada Muñoz, gtejadam@unmsm.edu.pe
[2] Steven Jesús Zarzosa Chávez, Jesús_zch@yahoo.com
[3] Víctor Benítez Casma , vbc86pe@hotmail.com
Recibido: Mayo 2011 / Aceptado: Junio 2011

manera tradicional. A manera de ejemplo, en la Fig. 3, nos planteamos el problema de encontrar un circuito lógico de entrada "m" y salida "Z" ambas respondiendo en sincronía con los flancos del reloj que ingresan por la entrada clk. El circuito posee una entrada asíncrona (independiente del reloj) denominada "inicio" que reinicializa al circuito manualmente. El circuito opera de tal forma que la salida "Z" cambia de 0 a 1 si se lee por "m" en sincronía con cada flanco consecutivo del reloj los bits 101.

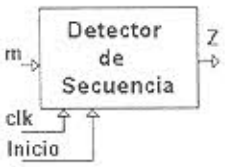


Fig. 3. Bloque lógico del circuito.

El Diagrama de Estados tipo Moore que cumple con la especificación del problema se muestra en la Fig. 4.

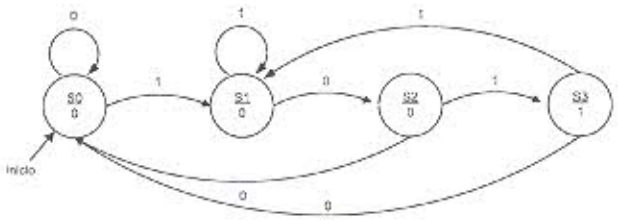


Fig. 4. Diagrama de Estados tipo Moore.

La Tabla de Estados, Tabla de Transición de Estados, Mapas del Karnaught de la lógica combinacional del estado siguiente, el mapa de Karnaught de la lógica combinacional de salida y la realización del circuito se muestran en las Figs. 5, 6, 7, 8, 9 y 10 respectivamente.

m	0	1	Z
S0	S0	S1	0
S1	S2	S1	0
S2	S0	S3	0
S3	S0	S1	1

Fig. 5. Diagrama de Estados.

$y_1 y_2$ \ m	0	1	Z
00	00	01	0
01	11	01	0
11	00	10	0
10	00	01	1

Fig. 6. Diagrama de Transición de Estados.

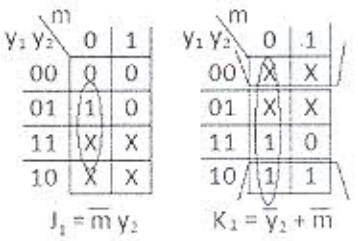


Fig. 7. Lógica combinacional de estado siguiente para el primer Flip Flop.

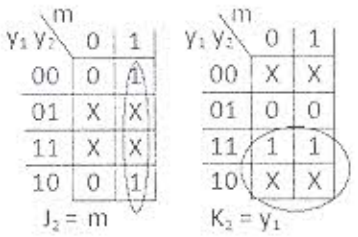


Fig. 8. Lógica combinacional de estado siguiente para el segundo Flip Flop.

y_1	y_2	Z
0	0	0
0	1	0
1	1	0
1	0	1

$Z = y_1 y_2$

Fig. 9. Lógica combinacional de salida.

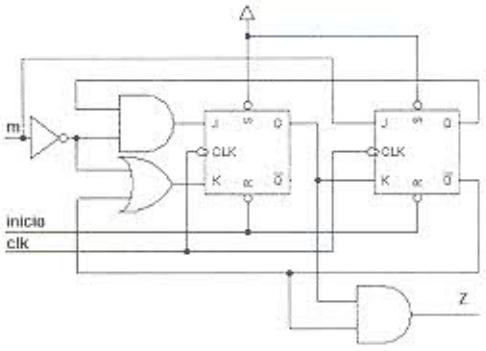


Fig. 10. Realización de circuito lógico.

II. METODOLOGÍA

Teniendo como único dato el Diagrama de Estados de la Fig. 4, se ha programado al FPGA inserto en el circuito integrado XC2VP30 del sistema de desarrollo XUP Virtex-II Pro utilizando como herramienta de software el ISE WebPACK 10.1 de distribución gratuita disponible en el sitio [2]. No se utilizó del software la opción de ingresar al código VHDL del circuito como en [3].

El Diagrama de Estados de la Fig. 4, como fue analizado, describe el funcionamiento de un Detector

de la secuencia 101. A continuación se indican los pasos para la Síntesis, Simulación, Implementación y Programación del FPGA que son los mismos como los tratados en detalle en [3].

A. Pasos a Seguir

1) Utilizando Xilinx ISE se crea un nuevo proyecto llamado detectsec101, tal como se muestra en la Fig. 11.

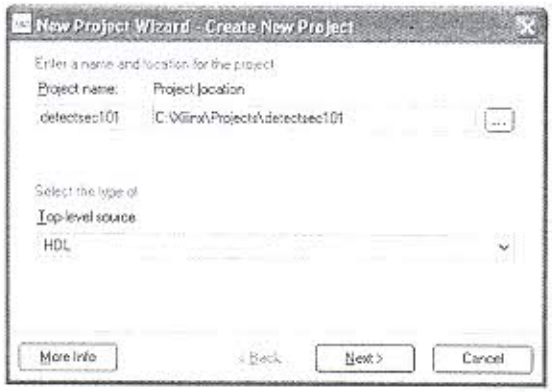


Fig. 11. Creación del Proyecto.

2) En la barra del menú se selecciona Project -> New Source. A continuación se elije la opción State Diagram y se escribe el nombre de la fuente así como el nombre del proyecto: detectsec101, tal como se muestra en la Fig. 12. Se presiona Next y luego Finish para salir del asistente.

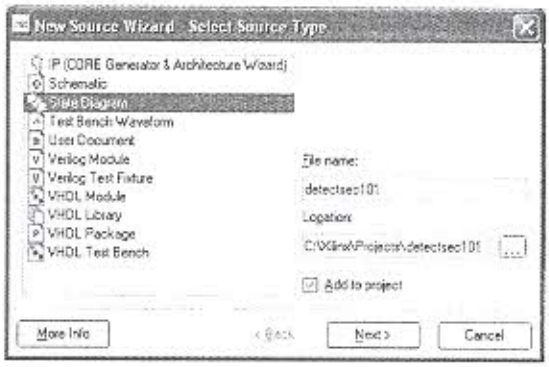


Fig. 12. Creación del archivo fuente.

3) Aparece una nueva ventana, ver Fig. 13, en donde se dibuja el diagrama de estados del detector de secuencia "101" de la Fig. 4.
4) Se Selecciona de la barra de herramientas la opción Add State , luego se desplaza el cursor a la hoja de diseño y se presiona el botón izquierdo del mouse ("click") sobre la hoja de diseño, por cada click realizado aparecerá un estado. Se repite el proceso para colocar los cuatro estados según el diseño de la Fig. 4, el esquema queda como la mostrada en la Fig. 14.
5) Se selecciona de la barra de herramientas la opción Add Transition , esta herramienta permite dibujar las curvas de transición entre los estados, que son flechas, que en sus extremos poseen puntos de color negro y en la parte central de ellas poseen un punto de color azul y otro verde para cambiar la forma, tal como se muestra en la Fig.15.

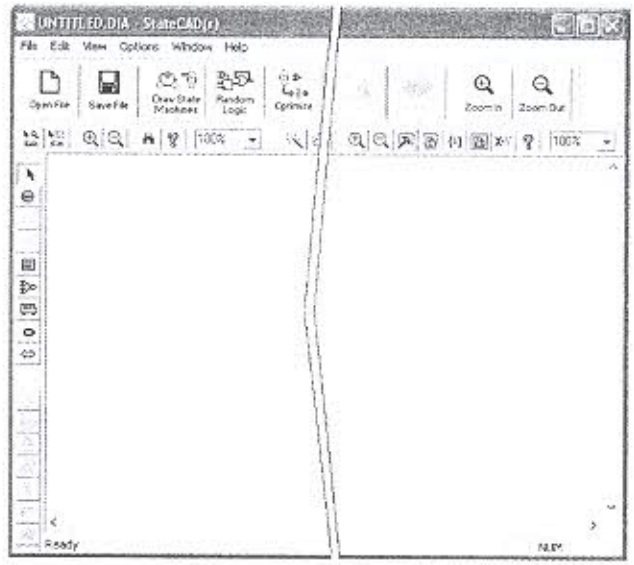


Fig. 13. Ventana de Diseño del esquemático.

Fig. 14. Estados del diseñó detector de secuencia

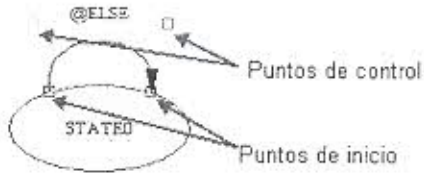


Fig. 15. Tipos de puntos en Líneas de Transición

6) Para dibujar las transiciones en lazo cerrado, ver Fig. 16a, primero se selecciona el estado, luego se realiza un primer click para el punto de inicio sobre el borde del estado y el otro segundo click para el punto final en otra zona del borde.
7) Para dibujar transiciones de un estado a otro, ver Fig. 16b, primero se selecciona el estado de inicio, luego se ubica el mouse sobre el borde de ese estado y hacer click, se debe observar que aparece

un primer punto llamado punto de inicio, luego se posiciona el mouse sobre la hoja de diseño, en un zona intermedia entre ambos estados, se ubica el segundo (color verde) y el tercer punto (color azul) los cuales determinan la curvatura de la transición. Finalmente se ubica el mouse sobre el borde del estado final y se hace click para finalizar la transición. El esquema queda como en la presentada en la Fig. 17.

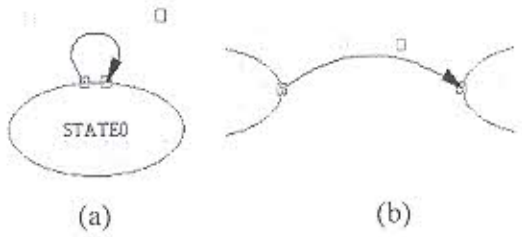


Fig. 16. Tipos de transiciones en un estado.

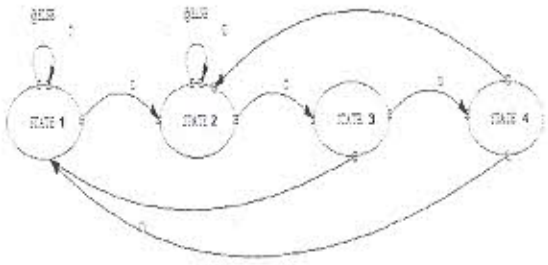


Fig. 17. Diagramas de estados preliminar tipo Moore del detector de secuencia 101.

8) Para ingresar las condiciones de transición entre estados, se hace doble click sobre una curva de transición, en la ventana emergente se ingresa la condición de salto en el campo Condition, ver Fig. 18. Cómo se trata de un circuito tipo Moore el campo Output debe quedar vacio éste sólo se llenará en el caso que el circuito sea de tipo Mealy.

Fig. 18. Ventana de asignación para la condición de transición.

9) Para asignar un nombre a cada estado y el valor de salida que le corresponde a ese estado (si fuera tipo Moore), se hace doble click sobre un estado, en la ventana emergente se coloca el nombre del estado (en este caso se ha optado por dejar la que aparece por defecto). Se asigna el valor de la salida en el campo Output, ver Fig. 19. Si se tratará de un circuito tipo Mealy este último campo debe quedar vacio.

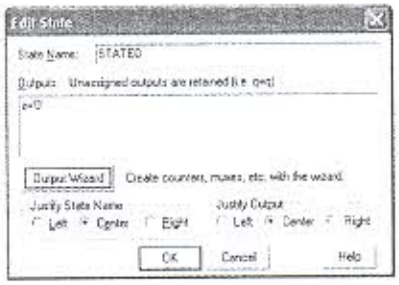


Fig. 19. Ventana de asignación para el nombre del estado y el valor de la salida.

10) Para agregar la señal de inicio, la cual permite iniciar o reiniciar el circuito como se describe en la Fig. 4, se selecciona de la barra de herramientas la opción Add Reset , se hace un primer click en una zona vacía de la hoja de diseño para posicionar el primer punto de inicio y luego ubicar el mouse sobre el borde del estado donde se desea que se posicione la señal. El nombre por defecto es Reset, para cambiar el nombre se hace doble click sobre el nombre y se modifica su contenido, tal como muestra la Fig. 20. El esquema final resultante se muestra en la Fig. 21.

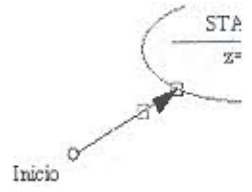


Fig. 20. Cambiando de nombre al Reset por Inicio

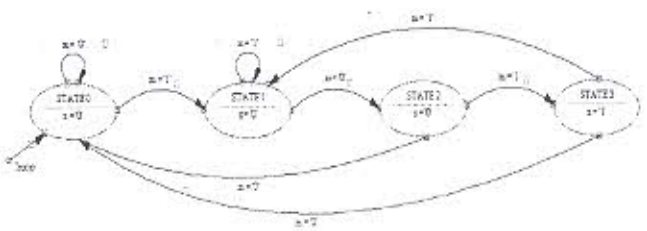


Fig. 21. Diagrama de estados final para el detector de secuencia 101.

11) Se guarda el diseño con el nombre detectsec101 en el directorio del proyecto. El archivo guardado tiene por extensión *.DIA.
12) Se configura los parámetros del programa StateCAD. De la barra de menú se selecciona Options → Configuration. En el bloque Language se debe verificar que la opción VHDL este seleccionada y en el bloque Language Vendor se selecciona la opción Xilinx XST, ver Fig. 22. Las demás opciones se deben dejar por defecto, se presiona OK para confirmar. Estos parámetros determinan el tipo de lenguaje de descripción de hardware que se generará al finalizar la compilación y además con que herramienta será sintetizada e implementada. Finalmente se guarda el diseño.

Fig. 22. Configuración de parámetros del StateCAD.

13) Para realizar el proceso de revisión de los errores de interconexión que pudieran existir en el diagrama de estados, se debe seleccionar de la barra de herramientas la opción Compile . La ventana emergente Results, ver Fig. 23, mostrará los posibles errores o problemas que pudiesen existir o su correcta compilación. Además mostrará el resumen de la compilación así como información del diseño.
14) Al presionar el botón Close aparece una nueva ventana emergente que muestra el código VHDL equivalente al Diagrama de Estados que el software ha generado, ver Fig. 24. En los comentarios se indica que el código VHDL fue creado y guardado en la carpeta de nuestro proyecto, en este caso como: C:\XILINX\PROJECTS\DETECTSEC101\DETECTSEC101.vhd.
15) Al regresar a la ventana Proyect Navigator de Xilinx, aparecen los archivos DETECTSEC101.vhd y DETECTSEC101.DIA en la ventana source. Si no fuera así, adjuntarlos, abrir Proyect/addsource y seleccionarl uno por uno del directorio del proyecto.
16) Se sintetiza con la opción Synthesize-xst de la ventana Processes.
17) Utilizando ModelSim, se simula el comportamiento del circuito para un intervalo de simulación de 850 nseg. La Fig. 25, muestra los resultados de la simulación.

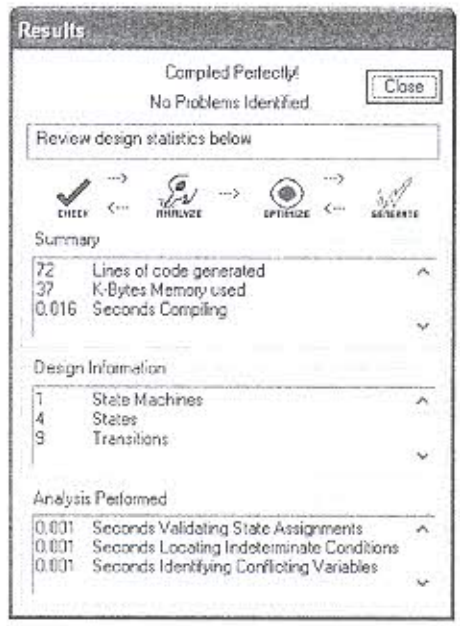


Fig. 23. Ventana emergente Results.

Fig. 24. Código VHDL equivalente

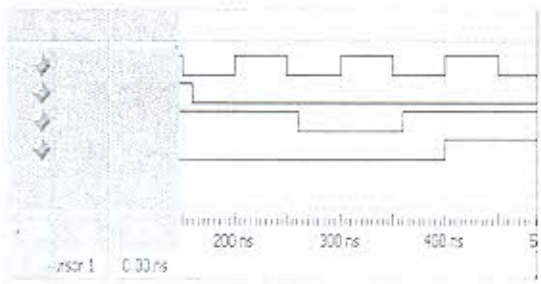


Fig. 25. Simulación del comportamiento del Multiplicador en ModelSim.

18) Se implementa con la opción Implementation de la ventana Sources.
19) Se asigna los puertos I/O a los pines del dispositivo xcv2p30: la señal "clk" al pin AF8 del FPGA que corresponde a SW_2, la señal "inicio" al pin AF9 del FPGA que corresponde a SW_3, la señal "m" al pin AC11 del FPGA que corresponde a SW_0 y la señal "z" al pin AC4 del FPGA que corresponde al LED_0. Para más detalles de la configuración de los pines del FPGA puede consultar el Manual de Referencia de Hardware ubicado en la referencia [4]. Finalmente, los switchs y LEDs quedan configurados como se indica en la Fig. 26a.
20) Programar el FPGA mediante la opción Configure Target Device de la ventana Sources y la opción Program tal como fue descrito en [3].

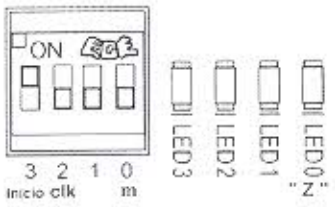
III. RESULTADOS

Los resultados de la programación del FPGA se muestran en las fotos de las Figs. 26b, 26c y 26d respectivamente. Hay que precisar que de acuerdo al hardware de la tarjeta XUP Virtex-II Pro hay que tener en cuenta la lógica invertida de los LEDs, es decir un LED encendido representa un cero lógico y viceversa. Así como también, un switch en la posición ON representa un 0 lógico y en la posición de OFF representa a 1 lógico.

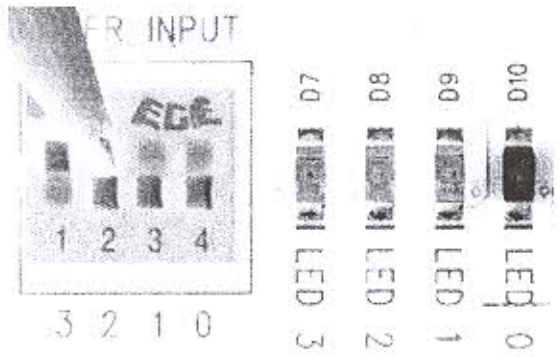
En la figura 26b, se muestra el momento en que con el primer flanco de subida del reloj es leído el 1 lógico de la entrada y con la salida en 0 lógico, en el flanco inmediato del reloj que se muestra en la Fig. 26b ingresa un 0 lógico permaneciendo la salida en 0 lógico, finalmente en el tercer flanco del reloj ingresa por un 0 lógico cambiando la salida a 1 lógico, verificándose el correcto funcionamiento del circuito al detectar la secuencia 101 tal como se esperaba.

IV. CONCLUSIONES

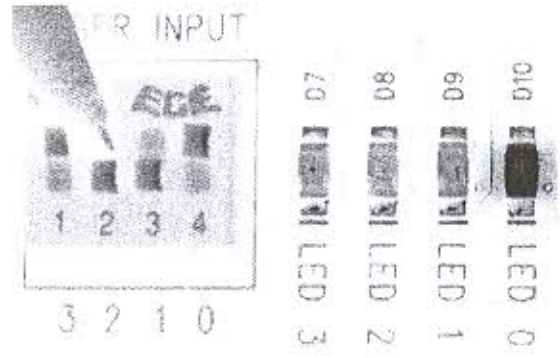
Se ha demostrado la facilidad con que es posible programar un FPGA sin necesidad de describir el funcionamiento del circuito lógico en código VHDL, sólo se ha precisado describir su funcionamiento gráficamente a través de su Diagrama de Estados siendo asistido por el programa WEBPACK de distribución gratuita.



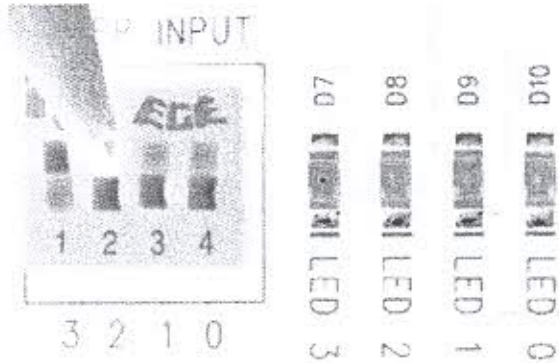
(a)



(b)



(c)



(d)

Fig. 26. Resultados experimentales en el sistema de desarrollo XUPV2P.

REFERENCIAS

[1] Digital Design Principles and Practices, John F.Wakerly, Pearson, 2001.

[2] ISE WebPACK Design Software. Disponible en: http://www.xilinx.com/tools/webpack.htm. Acceso: Junio 2011.

[3] Realización de Circuitos Lógicos en FPGA a partir de su Código en VHDL; Guillermo Tejada Muñoz, Steven Jesús Zarsosa Cháves, Víctor Benítez Casma; Electrónica-UNMSM, N° 26, diciembre 2010.

[4] Xilinx University Program Virtex-II Pro Development System - Hardware Reference Manual. Disponible en: http://www.xilinx.com/univ/XUPV2P/Documentation/ug069.pdf. Acceso: junio 2011.