Realización de Circuitos Lógicos en FPGA a partir de su Código en VHDL

Realization of FPGA Logic Circuits from the VHDL Code

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

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

Resumen— El propósito del siguiente trabajo es aprender la metodología por la cual a partir de un archivo escrito en código VHDL puede ser realizado un circuito lógico en un FPGA, con tal finalidad, como muestra didáctica se ha tomado como ejemplo la realización de una puerta lógica XOR dentro de la FPGA, pudiéndose aplicar el mismo procedimiento para circuitos muchos más complejos. Se detalla el proceso de síntesis, implementación, simulación y la programación del FPGA, se utiliza el sistema de desarrollo "XUP Virtex-II Pro" que contiene al circuito integrado "XC2VP30" dentro del cual está inserto un FPGA. El ISE WebPACK 10.1, de distribución gratuita, y opcionalmente el simulador ModelSim XE son utilizados como herramientas de software.

Abstract— The purpose of this paper is to learn the methodology by which from a file written in VHDL code can be made a logic circuit in an FPGA, for this purpose, as a sample didactical has been taken as an example the realization of an XOR logic gate inside the FPGA, the same procedure could be applied to many more complex circuits. It details the process of synthesis, implementation, simulation and FPGA programming, it uses the development system "XUP Virtex-II Pro" which contains the integrated circuit "XC2VP30" within which is embedded an FPGA. The ISE WebPack 10.1, free distribution, and optionally the simulator ModelSim XE are used as software tools.

Palabras clave— FPGA, XUP Virtex-II Pro, ISE WebPACK 10.1, iMPACT, simulador ISE, ModelSim XE, VHDL.

Key words— FPGA, XUP Virtex-II Pro, ISE WebPack 10.1, iMPACT, ISE simulator, ModelSim XE, VHDL.

I. INTRODUCCIÓN

A. El sistema de desarrollo XUP Virtex-II Pro

El sistema de desarrollo XUP Virtex-II Pro se muestra en la Fig. 1, en la parte central se encuentra el Circuito Integrado (C.I.) XC2VP30 y en torno a él se sitúan los periféricos. El C.I. XC2VP30 está compuesto internamente por un (1) FPGA (del inglés Field Programmable Gate Array), dos (2) microprocesadores Power-PC y ocho (8) transceptores Multi Gigabit. Las características del FPGA que está insertado en el C.I. XC2VP30 se muestran en la Tabla I.

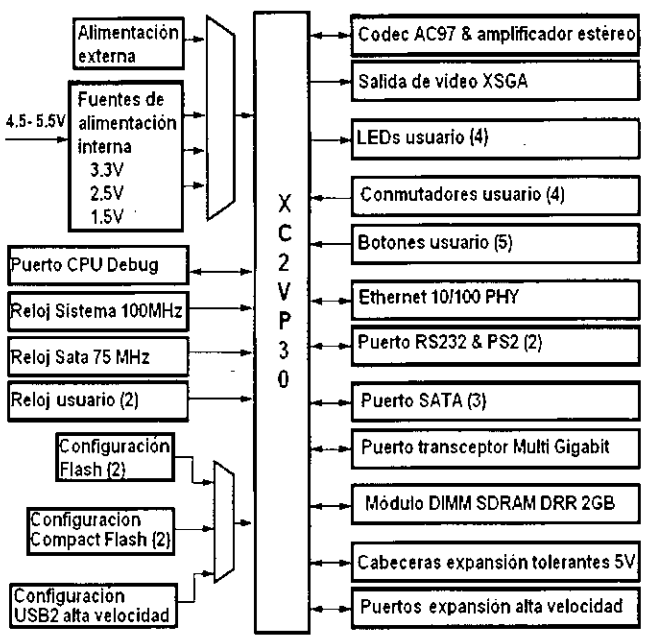


Fig. 1. Diagrama de bloques del sistema de desarrollo XUPV2P.

[1] Guillermo Tejada Muñoz. E-mail: gtejadam@unmsm.edu.pe
[2] Steven Jesús Zarzosa Chávez. E-mail: jesús_zch@yahoo.com
[3] Víctor Benítez Casma. E-mail: vbc86pc@hotmail.com

TABLA I
CARACTERÍSTICAS DEL FPGA INSERTO EN EL C.I. XC2VP30

Slices	13969
Tamaño de arreglo	80 x 46
RAM distribuida	428 Kb
Bloques multiplicadores	136
Bloques de RAM	2448 Kb
Administradores de reloj digital	8

B. Software para la programación de FPGA

1. Xilinx WebPack ISE 10.1

Xilinx WebPack ISE (Integrated Software Enviroment) es un programa de distribución gratuita que permite realizar fisicamente un diseño circuital lógico dentro de un FPGA. Se puede trabajar a partir de un archivo escrito en VHDL (de VHSIC y HDL, donde VHSIC es el acrónimo de Very High Speed Integrated Circuit y HDL es el acrónimo de Hardware Description Language) que describe el comportamiento del circuito, del diseño lógico o de su diagrama de estados. En este trabajo se detalla el procedimiento que se sigue para programar al FPGA a partir de un archivo escrito en VHDL. Brevemente, en la Fig. 2 se describen las etapas que se sigue para la programación del FPGA. El programa Xilinx WebPack ISE puede ser descargado de la página de la referencia [1].

Una vez que se ingresa al programa Xilinx mediante su Interface Gráfico de Usuario (GUI) llamado Project Navigator el usuario crea y da nombre al Proyecto. Los archivos fuentes pueden ser creados y editados mediante esquemas o circuitos lógicos combinacionales o diagramas de estados para el caso de circuito secuenciales, en ambos casos el programa los traducirá en un archivo que contenga el funcionamiento de los circuitos en código VHDL.

También se tendrá la alternativa que el usuario pueda directamente agregar un archivo que contenga el código VHDL de cualquier circuito lógico, como se describe en el presente trabajo.

Posteriormente, luego del proceso de Síntesis se pasará a la etapa de Simulación, en donde mediante el Simulador ISE o el Modelsim-SE se simulará el funcionamiento del circuito. Luego se pasará al proceso de implementación con el software incluido PACE (*Pinout and Area Constraints Editor*) que asignará las direcciones de los puertos de entrada y salida a los pines del FPGA. Finalmente, se realizará la programación del FPGA, localizado en el sistema de desarrollo de la placa VIRTEX II Pro, mediante el software incluido denominado iMPACT, con lo cual el proceso concluye quedando el circuito lógico realizado en el FPGA.

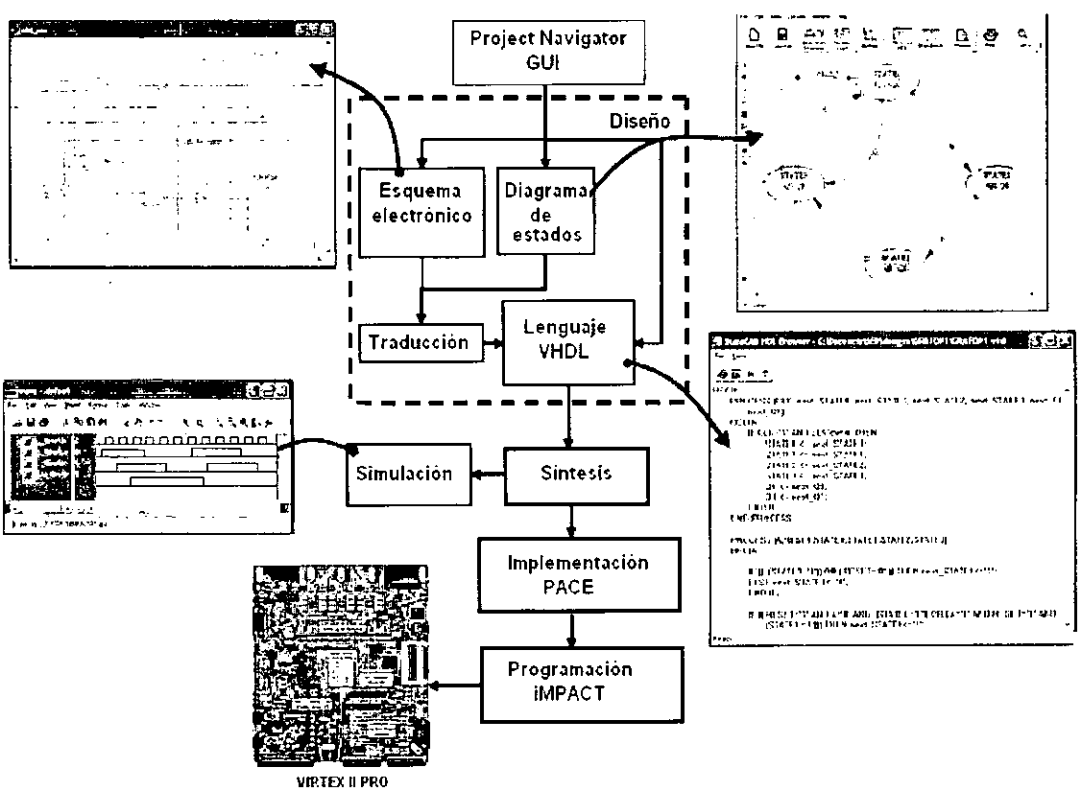


Fig. 2. Etapas para la programación del FPGA con el uso de XILINX WEB PACK 10.1.

2. ModelSim XE y Simulador ISE

ModelSim XE (Xilinx Edition) es un simulador HDL utilizado para simular diseños de Xilinx, está basado en la máquina de simulación de ModelSim PE de Mentor Graphics y requiere ser instalado, el programa puede ser descargado de la página de Xilinx, ver tutorial de descarga de la referencia [2]. El simulador ISE en cambio está inserto dentro de Xilinx WebPack, en este trabajo se describirá ambos simuladores.

II. METODOLOGÍA

A. Creación un nuevo Proyecto

Abrir Project Navigator, desde Inicio > Programas > Xilinx ISE Design Suite 10.1 > ISE > Project Navigator o haciendo doble clic en el ícono Xilinx ISE 10.1 del escritorio. En la barra de herramientas, seleccionar File > New Project, asigne un nombre y una carpeta de trabajo tal como se indica en la Fig. 3. Presione Next para continuar. Introducir en la ventana emergente la información del dispositivo y de las herramientas a utilizar, tal como se muestra en la Fig. 4. Los detalles del dispositivo se encuentran el circuito integrado ubicado al centro del sistema de desarrollo XUPV2P, ver la Fig. 5. Presione Next para continuar.

No se crearán ni agregarán archivos fuente utilizando el asistente, por lo que debe hacer clic en Next en las dos ventanas emergentes que continúan.

Verificar la configuración del proyecto en la ventana emergente, tal como se muestra en la Fig. 6. Presione Finish para finalizar la creación del proyecto.

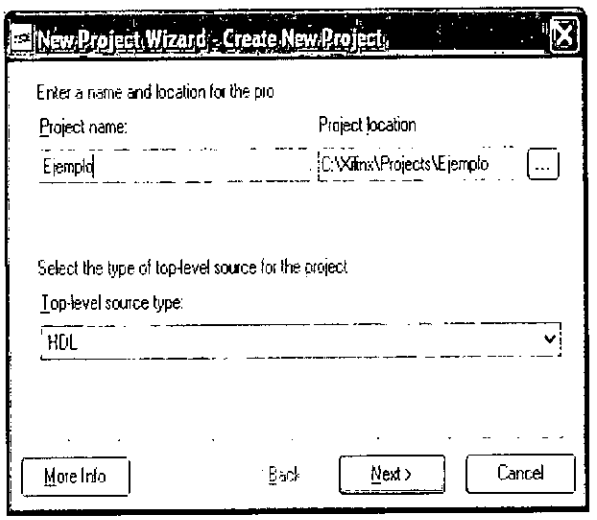


Fig. 3. Creación de un nuevo proyecto.

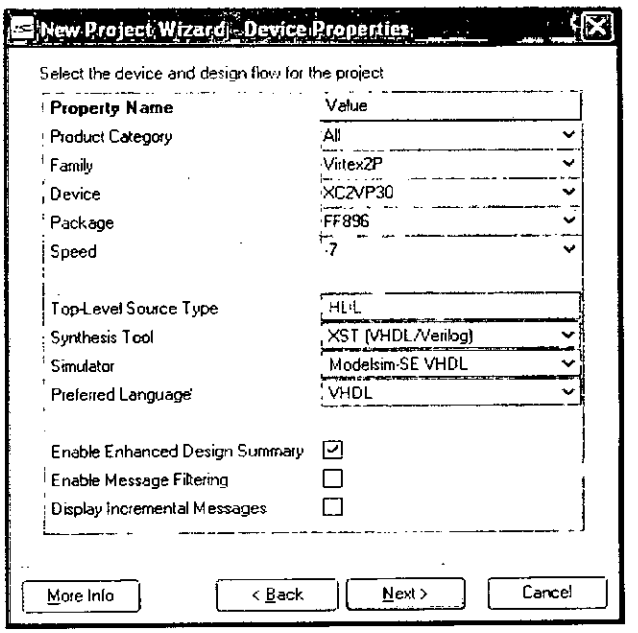


Fig. 4. Propiedades del dispositivo a utilizar.

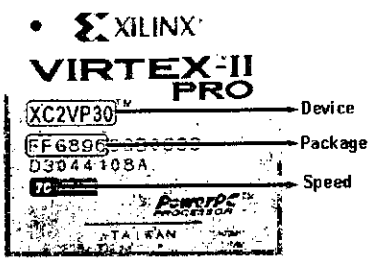


Fig. 5. Información impresa en el circuito integrado.

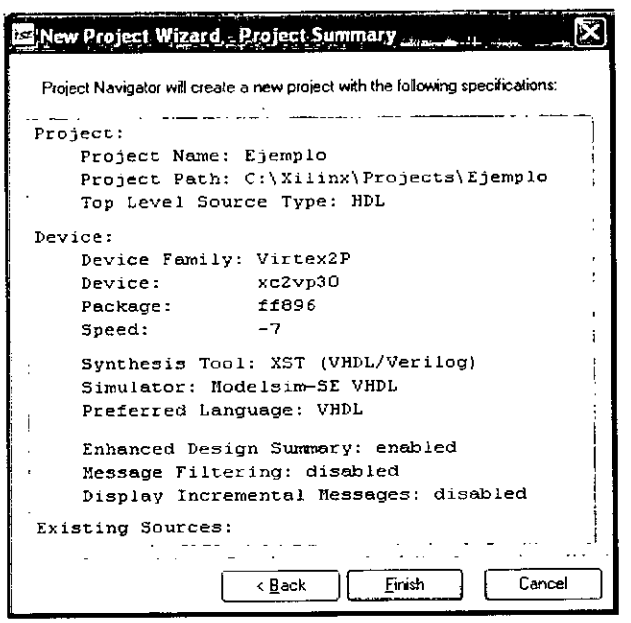


Fig. 6. Resumen de las especificaciones del proyecto.

B. Asignación del archivo fuente

Crear el archivo ejemplo_xor.vhd con la ayuda de un procesador de texto que contenga el código que se muestra en la Fig. 7, y guardarlo en la carpeta principal del proyecto: C:\Xilinx\Projects\Ejemplo.

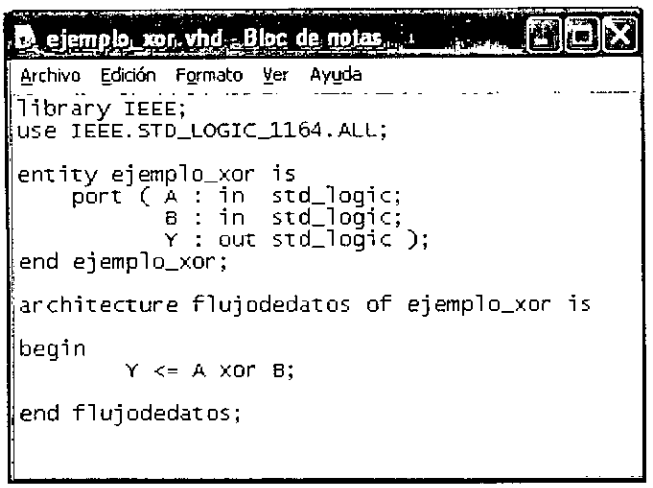


Fig. 7. Código VHDL del archivo ejemplo_xor.vhd.

En la barra de herramientas, seleccionar Project > Add Source y cargar el archivo ejemplo_xor.vhd. Presione OK en la pantalla emergente, ver Fig. 8.

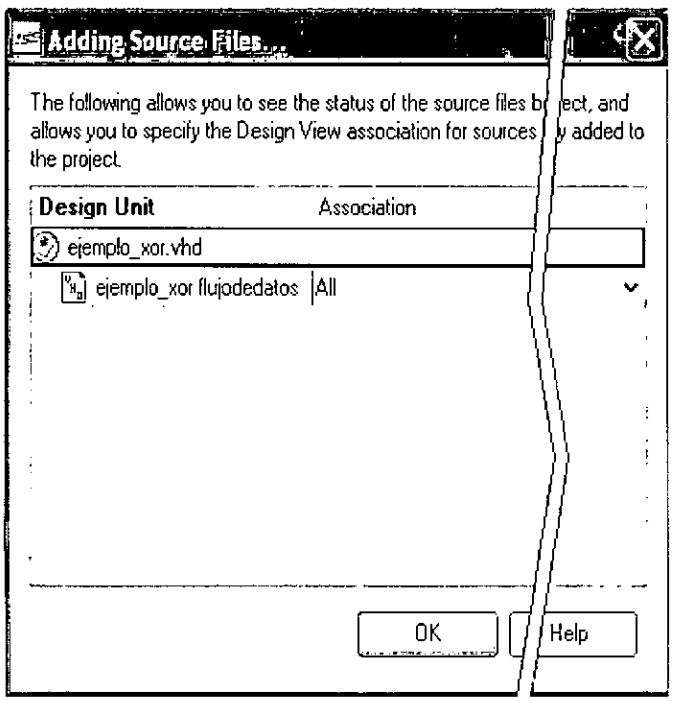


Fig. 8. Confirmación del archivo agregado.

Para observar el código VHDL que ha sido agregado hacer doble clic sobre el nombre del archivo en la ventana Sources y éste aparecerá en el área de trabajo, como se muestra en la Fig. 9. Ahora el área de trabajo funciona como un editor de texto.

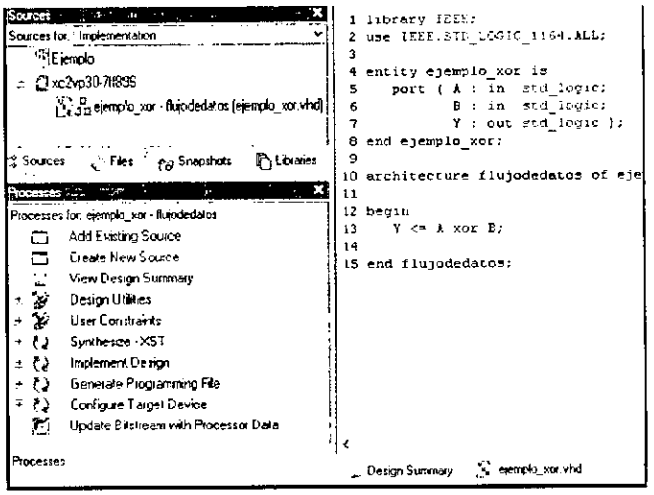


Fig. 9. Vista del código VHDL agregado.

C. Síntesis

En la ventana Processes, hacer doble clic en la opción Synthesize – XST. Si la síntesis fue exitosa, en la ventana Processes aparecerán un círculo verde como se muestra en la Fig. 10, en caso contrario aparecerá un círculo rojo, en este caso leer la ventana Transcript a fin de identificar y corregir el error.

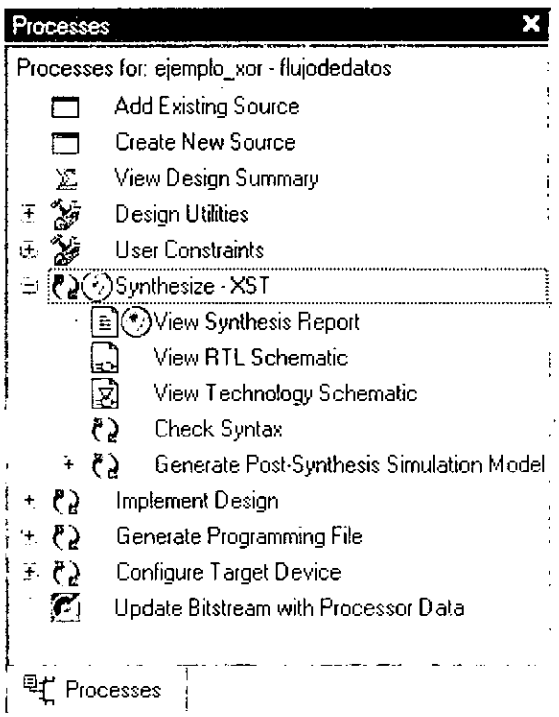


Fig. 10. Finalización de la síntesis.

D. Simulación mediante el simulador ISE

Crear una nueva fuente denominada tb_ejemplo_xor por medio del asistente, como se indica a continuación.

En la barra de herramientas, seleccionar Project > New Source. A continuación seleccione la opción Test Bench Waveform y escriba el nombre de la fuente; se aconseja que el nombre del archivo tenga el formato tb_<nombre del archivo vhd>, donde tb hace referencia a "test bench". Ver Fig. 11.

Seleccionar el tipo de fuente al cual está asociado nuestro archivo fuente de simulación, para este caso será ejemplo_xor. Ver Fig. 12.

Hacer click en Finish para salir del asistente.

Observar la ventana emergente que aparece, llamada Initial Timing and Clock Wizard de la Fig. 13, aplique sobre ella los siguientes pasos:

- En el bloque Clock Information, seleccionar la opción Combinatorial (or internal clock).
- No seleccionar la opción GSR (FPGA) del bloque "Global Signals".
- En el cuadro Initial Length of Test Bench colocar el tiempo estimado de simulación de acuerdo a las escala de tiempo seleccionado.

Hacer click en Finish para salir del asistente de configuración.

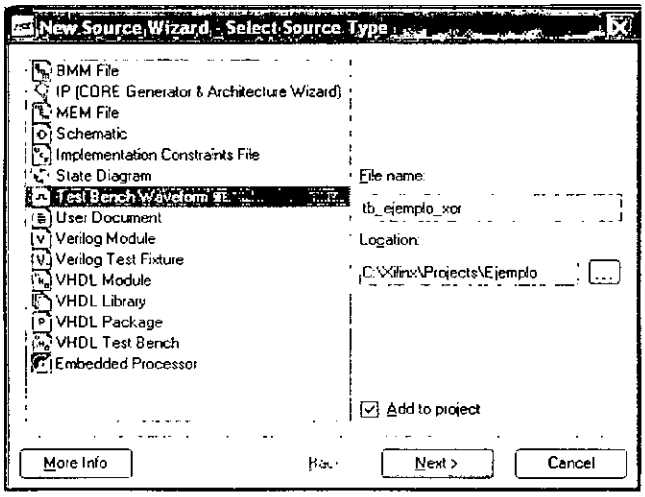


Fig. 11. Creación de un archivo fuente Test Bench Waveform.

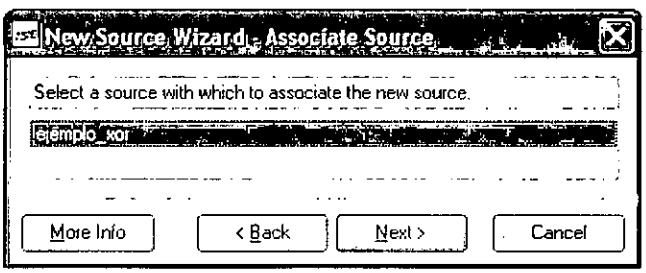


Fig. 12. Selección del archivo asociado al archivo fuente.

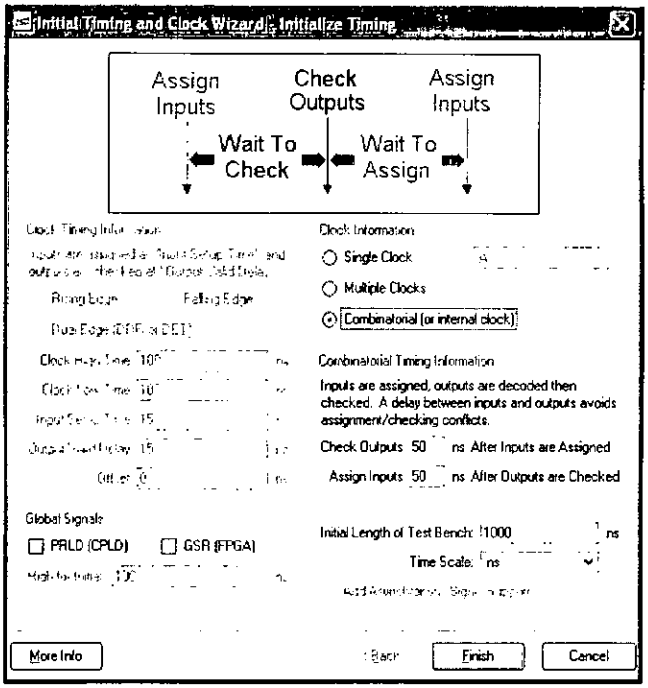


Fig. 13. Configuración para la escala de tiempos de la simulación.

Observar que aparece una nueva pestaña de nombre tb_ejemplo_xor.tbw en el área de trabajo, en ella se asignara los valores de simulación para las señales de entrada del diseño, ver Fig. 14.

Las señales de simulación pueden ser especificadas manualmente, en este caso, hacer un click en las áreas de color de la señales A y B para cambiar su estado lógico de tal manera de dibujar progresivamente las señales de acuerdo a los requerimientos del usuario.

En el caso que las señales de simulación van a ser especificadas automáticamente siga los siguientes pasos:

- Hacer click derecho en el área de asignación de la señal A y seleccionar Set Value.
- Presionar el botón Pattern Wizard, ver Fig. 15.
- Configurar la nueva ventana emergente llamada Patter Wizard como se muestra en la Fig. 16.

Para el caso de la señal B se debe repetir los pasos anteriores, se debe configurar la pantalla Pattern Wizzard como se muestra a continuación:

- Pattern Type: Pulse
- Number Cycles: 5
- Initial Value: 0
- Pulse Value: 1
- Initial Delay: 2
- Pulse Width: 2

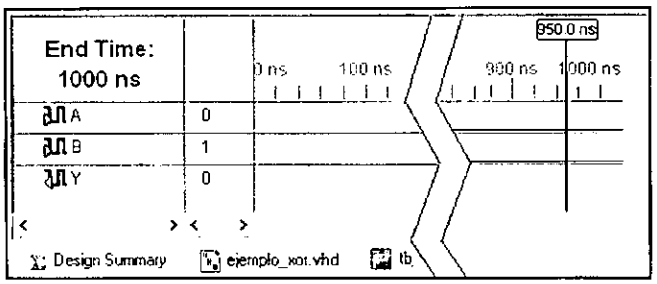


Fig. 14. Parte de la ventana Test Bench Waveform en el área de trabajo.

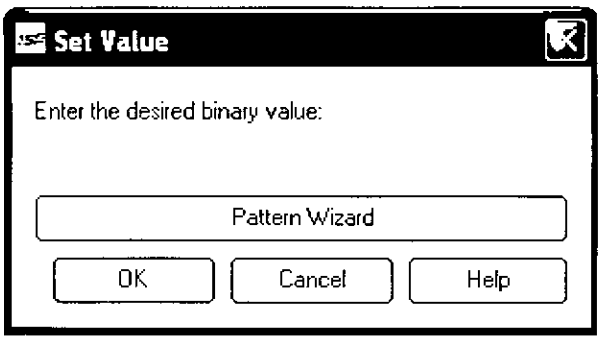


Fig. 15. Ventana Asignación de valor de señal.

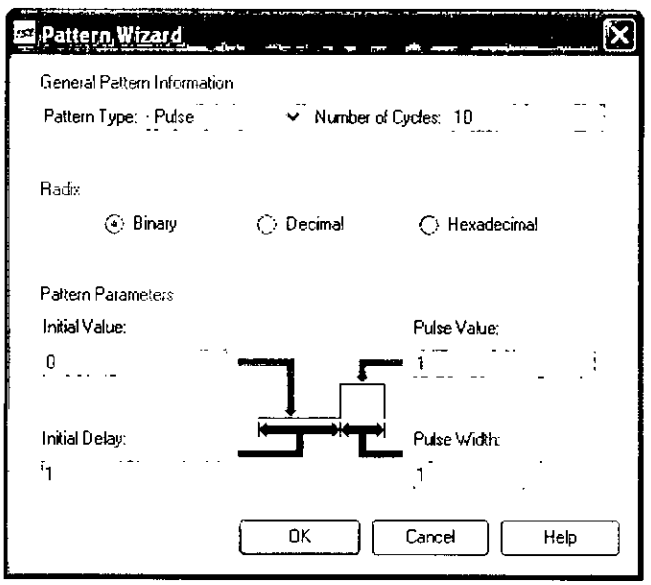


Fig. 16. Ventana Asistente de Asignación de valores de señal.

Para extender la escala de tiempo de la pestaña Test Bench de la Fig. 13 (que se mostró), hacer click derecho sobre una zona libre del área de trabajo, seleccionar la opción Set End of Test Bench y asignar un valor de tiempo final de 2000 ns, como se muestra en la Fig. 17.

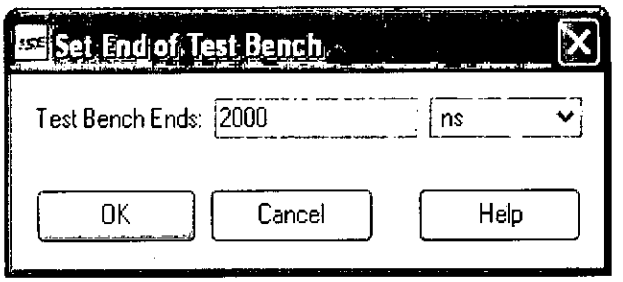


Fig. 17. Asignación del tiempo final de la ventana Test Bench.

Guardar el archivo y observar que en la ventana de trabajo aparecen las señales con los valores y la escala establecida. Ver Fig. 18.

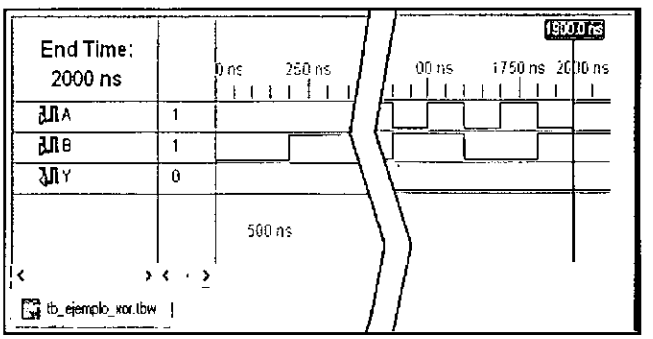


Fig. 18. Señales de simulación con los valores establecidos.

En la ventana Sources, seleccionar la opción Behavioral Simulation de la lista desplegable, luego seleccionar xc2vp30-7ff896 > tb_ejemplo_xor (tb_ejemplo_xor.tbw), tal como se muestra en la Fig. 19.

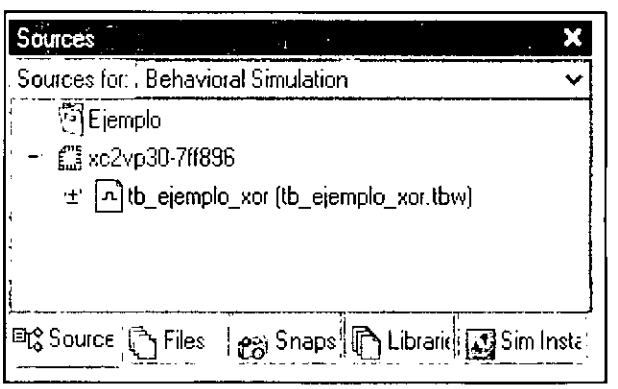


Fig. 19. Selección del archivo fuente a simular.

En la ventana Processes, seleccionar la pestaña del mismo nombre, en ella desplegar la opción Xilinx ISE Simulator y hacer doble click en Simulate Behavioral Model, ver Fig. 20.

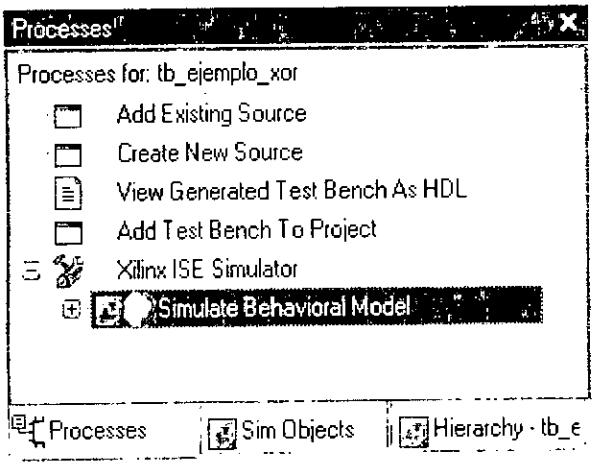


Fig. 20. Inicio de la Simulación.

Observar que aparecerá una nueva ventana con los resultados de la simulación, ver Fig. 21. Para extender el tiempo de simulación, como por ejemplo a 1000 ns, seleccione en la barra de herramientas la opción Run for Specified Time , ver Fig. 22.

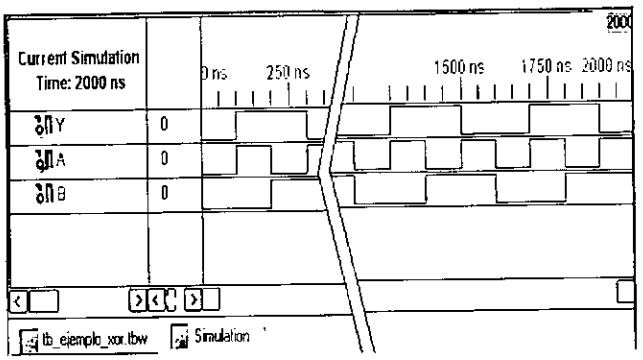


Fig. 21. Resultados de la simulación.

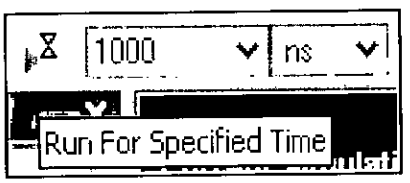


Fig. 22. Especificación de tiempo de simulación.

E. Simulación mediante ModelSim XE

1. En la ventana Sources, seleccionar la opción Behavioral Simulation de la lista desplegable, luego seleccionar xc2vp30-7ff896 > ejemplo_xor – flujodedatos (ejemplo_xor.vhd), tal como se muestra en la Fig. 23.
2. En la ventana Processes, seleccionar ModelSim Simulator > Simulate Behavioral Mode, tal como se muestra en la Fig. 24. Hacer doble clic para abrir ModelSim.

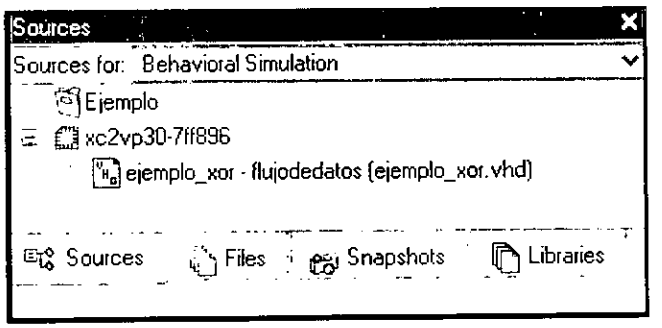


Fig. 23. Selección del archivo fuente a simular.

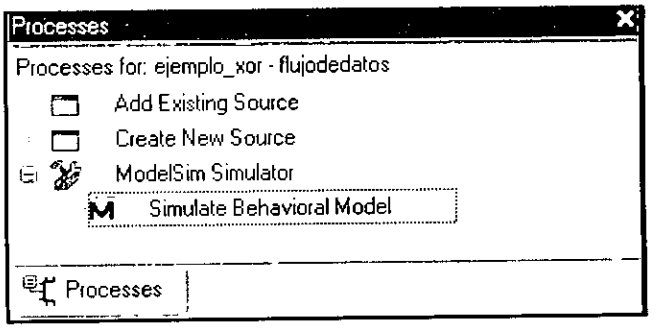


Fig. 24. Selección de ModelSim como simulador.

3. En ModelSim, hacer clic en el botón Unlock de la ventana Wave, ver Fig. 25, a fin de permitir que se visualicen con mayores detalles las señales de prueba que se ingresarán. Luego, maximizar la ventana Wave.
4. En la barra de herramientas de simulación, haga clic en el botón Restart, luego establezca el tiempo de simulación escribiendo 1000 ns en el cuadro de texto, ver la Fig. 26.

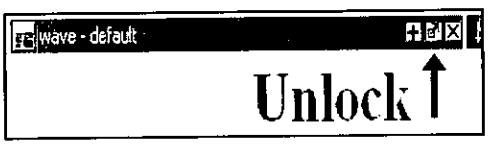


Fig. 25. Desbloqueando la ventana Wave.

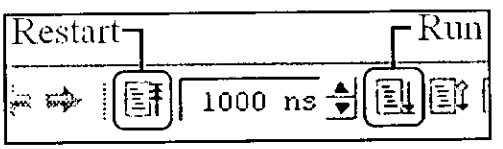


Fig. 26. Barra de herramientas de simulación.

5. En la ventana Wave de ModelSim, seleccionar la señal /ejemplo_xor/a, luego haga clic derecho y seleccione Clock, como se indica en la Fig. 27.
6. En la ventana emergente, ingrese el período de la señal de reloj, para este caso 200 ns, como se muestra en la Fig. 28. Luego haga clic en OK.

7. Repita los pasos 3 y 4 para la señal /ejemplo_xor/b, asigne período de 100 ns.
8. En la barra de herramientas de simulación, haga clic en el botón Run, ver Fig. 26. El contenido de la ventana Wave se actualizará, para observar los resultados de la simulación con más detalle haga clic en el botón Zoom Full de la barra de herramientas Zoom, ver la Fig. 29.
9. Hacer clic sobre la línea de tiempo y arrastrar el mouse para poder observar los valores de las señales de entrada y la respuesta del diseño lógico, tal como se muestra en la Fig. 30.

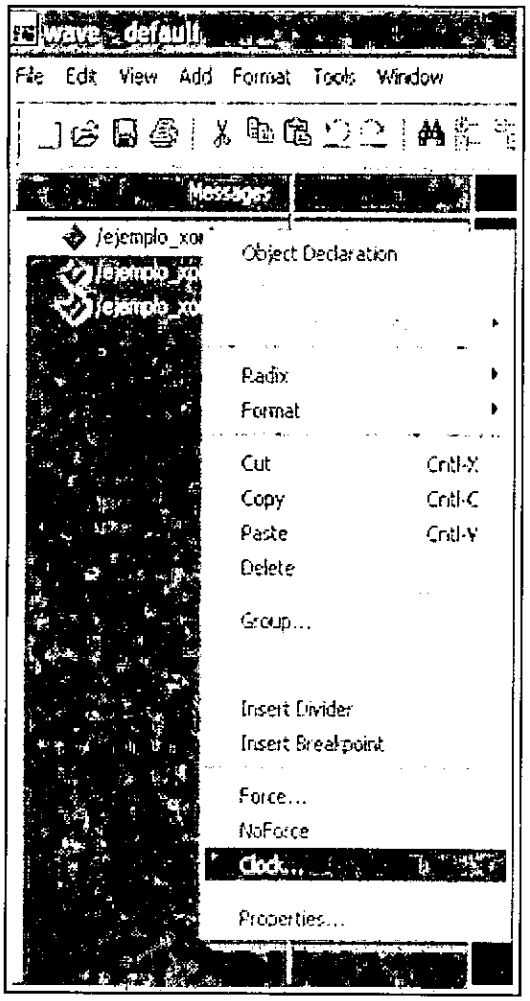


Fig. 27. Asignación de una señal de reloj a la señal A.

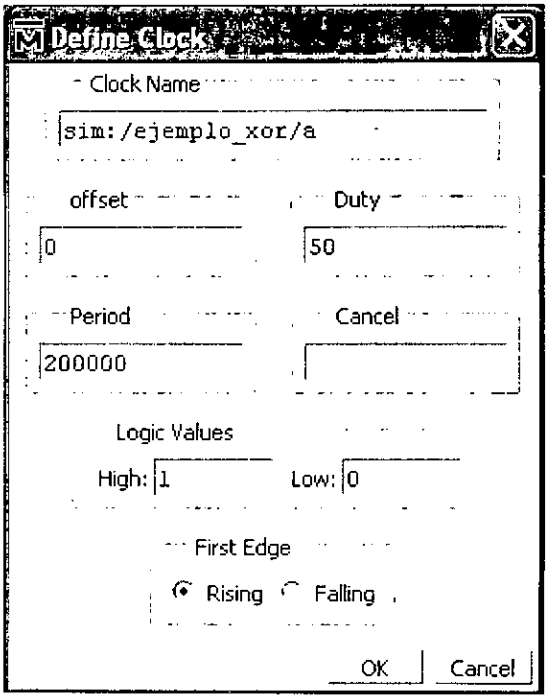


Fig. 28. Características del reloj.

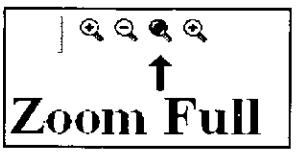


Fig. 29. Barra de herramientas de Zoom.

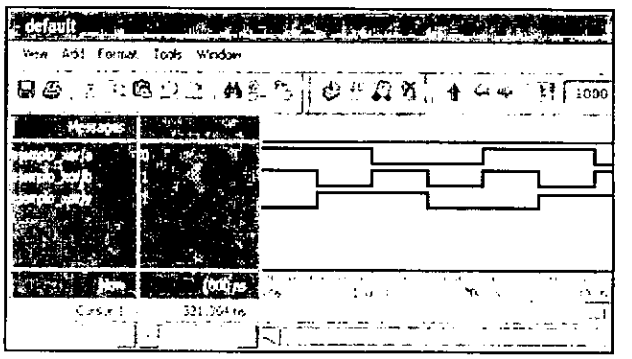


Fig. 30. Resultados de la simulación.

F. Implementación

En Project Navigator, seleccionar la opción Implementation de la lista de opciones en la ventana Sources, como lo indica la Fig. 31.

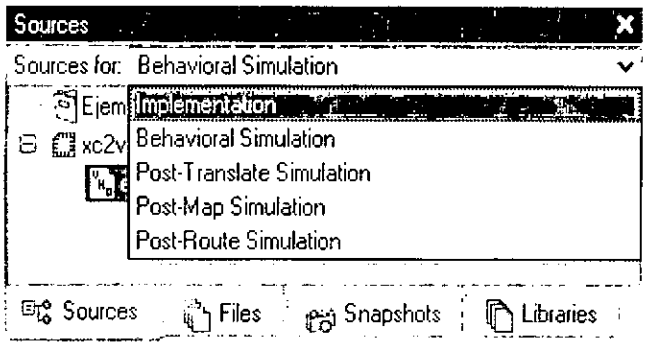


Fig. 31. Regresando de la simulación a la implementación.

Hacer doble clic sobre la opción Implement Design de la ventana Processes. En la ventana Transcript aparecerán mensajes relativos al proceso de implementación de diseño, una vez terminada la implementación aparecerán los círculos verdes mostrados en la Fig. 32.

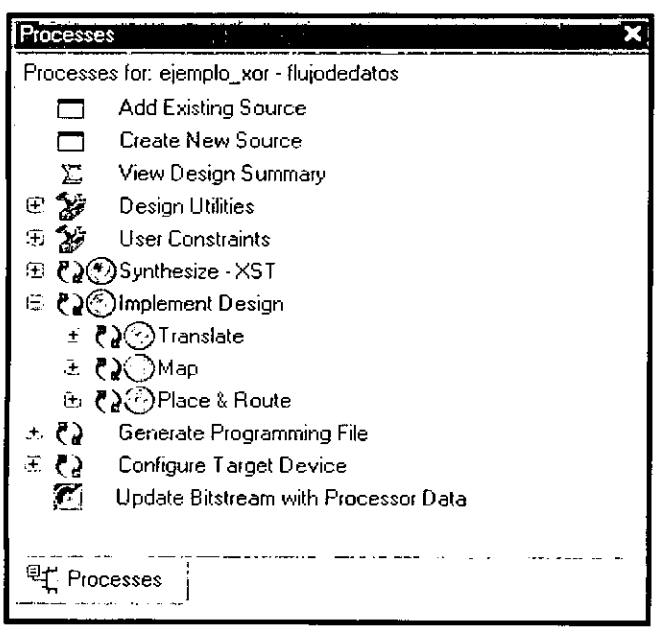


Fig. 32. Resultados del proceso de implementación.

Hacer doble clic sobre la opción Floorplan Area / IO / Logic – Post-Synthesis, dentro de la sección User Constraints en la ventana Processes, tal como lo muestra la Fig. 33. Este paso es obligatorio para posteriormente asignar a los pines del CI XC2VP30 las entradas y salidas del circuito (puertos).

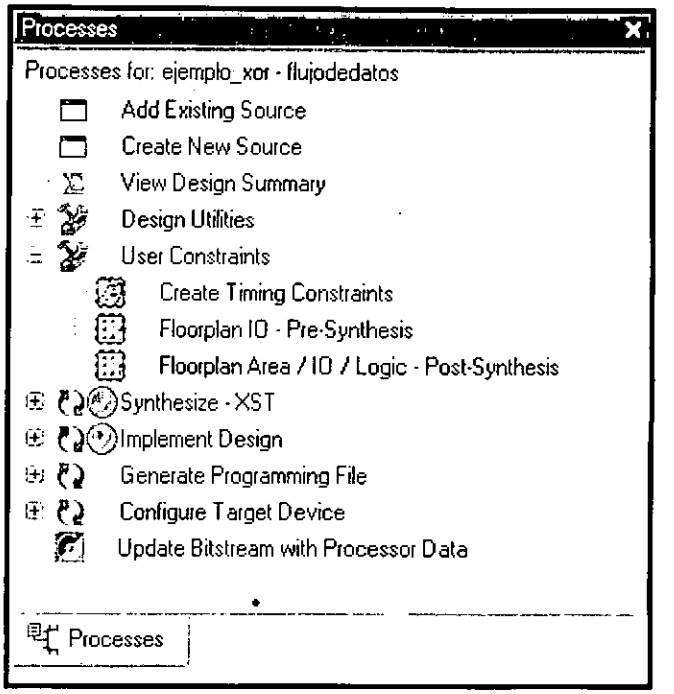


Fig. 33. Inicio del proceso de declaración de terminales.

Hacer clic sobre el botón OK en la ventana emergente y luego aparecerá la ventana principal de la herramienta Xilinx PACE (Pinout and Area Constraints Editor), ver la Fig. 34. Si en caso no pudiera ver alguna de las ventanas, puede activarlas desde el menú View.

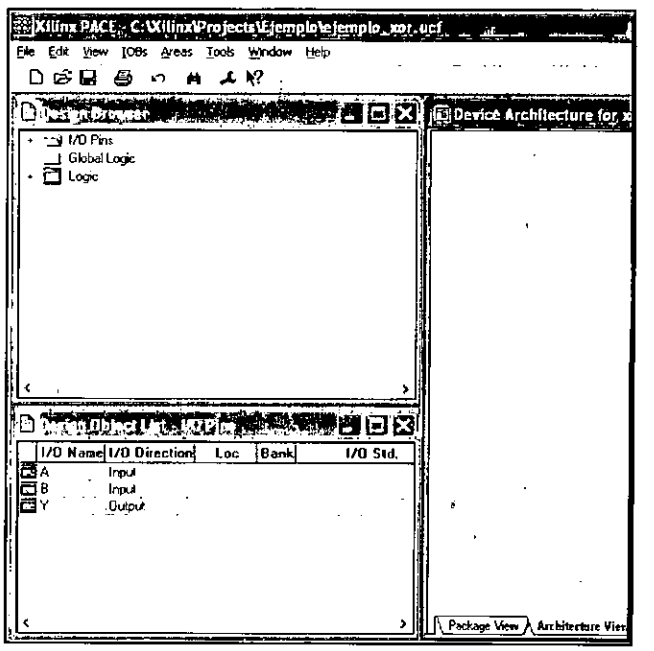


Fig. 34. Ventana principal de la herramienta Xilinx PACE.

En el campo Loc de la ventana Design Object List – I/O Pins, introducir los nombres de los pines que se utilizarán para cada puerto del proyecto. En

este caso se asignará el pin AC11, que corresponde a SW0 para el puerto A, AD11, que corresponde a SW1 para el puerto B y AC4, que corresponde al LED0 para el puerto Y, como se muestra en la Fig. 35. Para mayor información de la localización de los conmutadores y LEDs revisar el manual de referencia de hardware [3].

Cerrar Xilinx PACE y guardar los cambios, luego en Project Navigator volver a realizar los pasos 2 y 3, y comprobar que los pines asignados no han sido alterados. Si no hay error cerrar Xilinx PACE, caso contrario repetir los pasos anteriores.

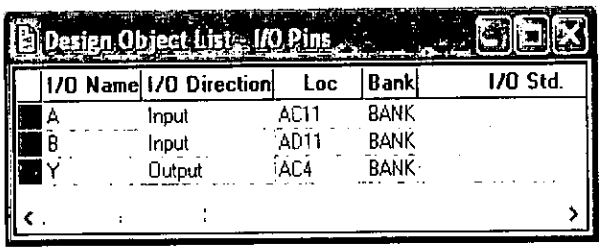


Fig. 35. Asignación de pines del dispositivo.

G. Programación del dispositivo

Conectar el sistema de desarrollo XUPV2P y encenderlo. Luego de que la computadora lo reconozca, hacer doble clic sobre la opción Configure Target Device de la ventana Sources. Configure las opciones de la ventana emergente, correspondiente a la herramienta iMPACT, como se muestra en la Fig. 36.

Hacer clic en el botón Bypass de las dos primeras ventanas emergentes denominadas Assign New Configuration File, ver Fig. 37(a), porque estas ventanas corresponden a la PROM y a SystemACE del sistema de desarrollo. En la tercera ventana, que corresponde al XC2VP30, seleccionar ejemplo_xor.bit y hacer clic en Open, ver Fig. 37(b).

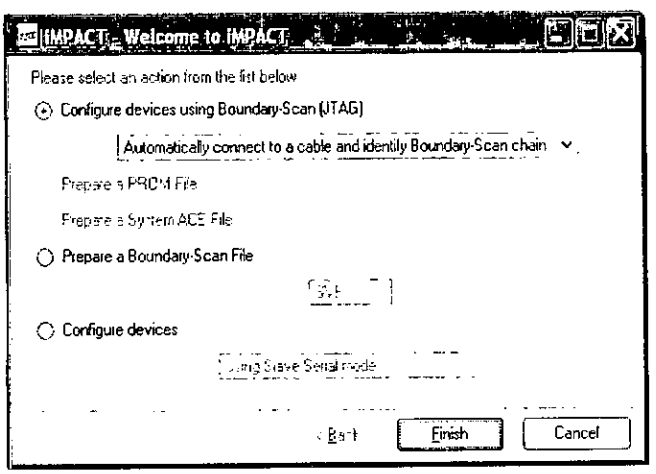
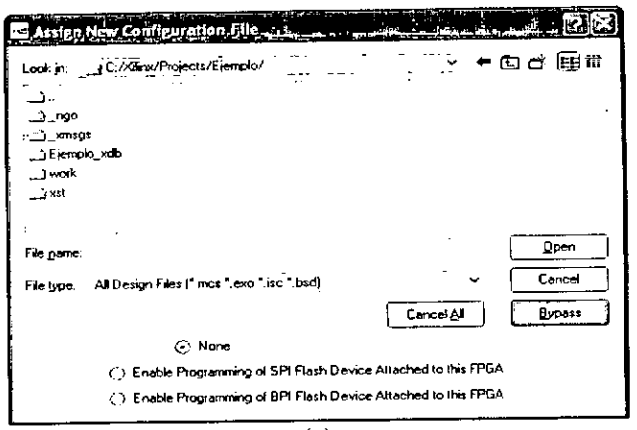
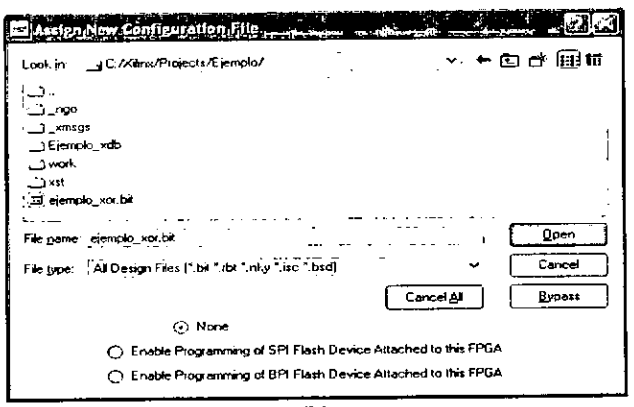


Fig. 36. Configuración de iMPACT.



(a)



(b)

Fig. 37. Asignación de archivo de configuración.

Hacer clic en OK en las dos ventanas emergentes, luego seleccionar el dispositivo XC2VP30, hacer clic derecho y seleccionar la opción Program, tal como se indica en la Fig. 38.

Hacer clic en el botón OK en la ventana emergente y se iniciará la programación del sistema de desarrollo XUPV2P. Aparecerá el mensaje Program Succeeded si el proceso fue exitoso, como se muestra en la Fig. 39.

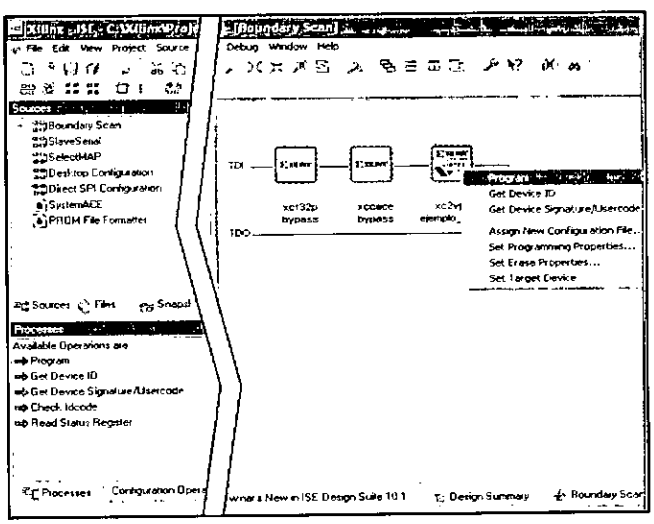


Fig. 38. Programación del dispositivo XC2VP30.

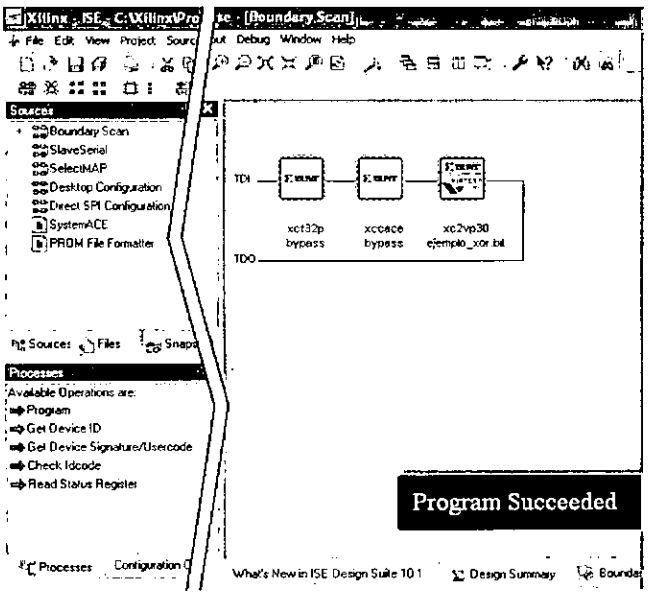
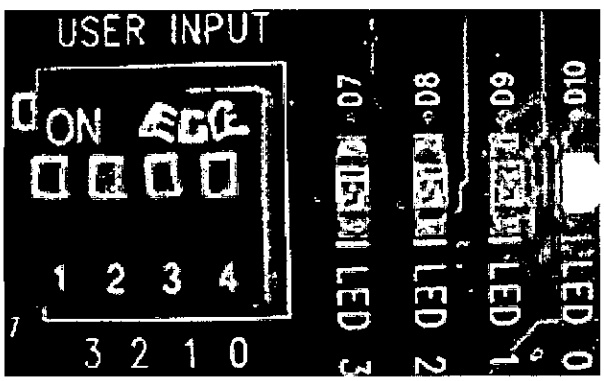


Fig. 39. Programación exitosa del sistema de desarrollo XUPV2P.

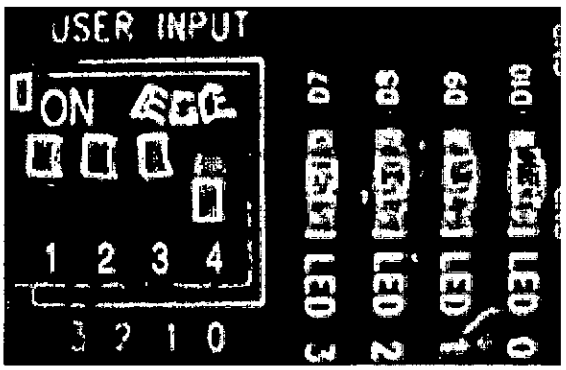
III. RESULTADOS

Luego de la programación exitosa del dispositivo, podemos verificar su funcionamiento, recordemos que se ha realizado una puerta lógica XOR en el FPGA, tal como inicialmente se describió en lenguaje VHDL, ver Figura 7, la puerta posee dos puertos de entradas (A, B) y uno de salida (Y).

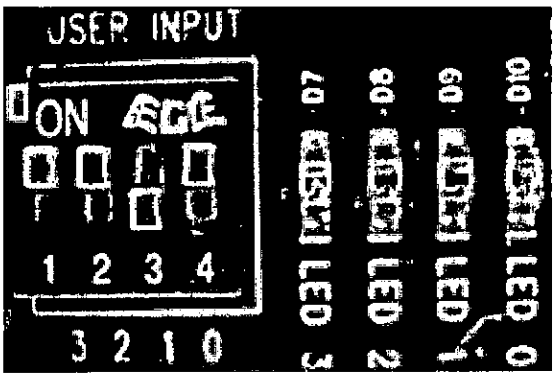
En las fotografías de la Figura 40, los conmutadores SW0 y SW1 representan respectivamente a los puertos A y B de la XOR, mientras que el LED0 representa al puerto Y (tal como fue especificado en el paso F de la Metodología).



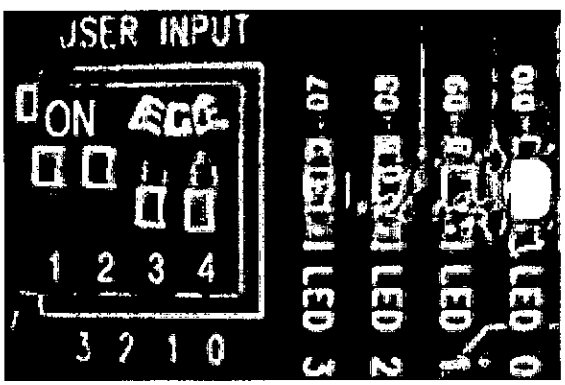
(a)



(b)



(c)



(d)

Fig. 40. Resultados experimentales.

Para verificar el correcto funcionamiento de la puerta lógica, hay que tener en cuenta que el LED0 está polarizado inversamente, por este motivo cuando la salida es 0 lógico el LED0 se ilumina, mientas que cuando es 1 lógico el LED0 se apaga. La Tabla II explica los resultados mostrados en las fotografías de la Figuras: 40(a), 40(b), 40(c) y 40(d).

TABLA II
RESULTADOS

Figura	SW1	SW0	LED0
40 (a)	ON (1 lógico)	ON (1 lógico)	Encendido (0 lógico)
40 (b)	ON (1 lógico)	OFF (0 lógico)	Apagado (1 lógico)
40 (c)	OFF (0 lógico)	ON (1 lógico)	Apagado (1 lógico)
40 (d)	OFF (0 lógico)	OFF (0 lógico)	Encendido (0 lógico)

Verificándose el correcto funcionamiento de la realización en el FPGA de la puerta lógica XOR.

V. CONCLUSIONES

Se ha realizado un circuito lógico dentro de un FPGA teniendo la descripción del circuito lógico en código VHDL, se ha pasado por todas las etapas del diseño, tales como: sintetizar, simular, implementar y programar el CI XC2VP30 del sistema de desarrollo XUPV2P utilizando el programa Xilinx WebPack ISE. La simulación ha sido lograda mediante dos herramientas: el simulador de ISE y el ModelSim XE.

Se ha descrito un proceso para un FPGA inserto en el sistema de desarrollo XUP Virtex-II Pro, sin embargo, los pasos descritos se ajustan para programar a cualquier otro FPGA e inclusive a CPLDs.

REFERENCIAS

[1] ISE WebPACK Design Software. Disponible en: http://www.xilinx.com/tools/webpack.htm. Acceso: Julio 2009.

[2] Tutorial de descarga e instalación de XILINX ISE Y MODELSIM. Disponible en: http://arantxa.ii.uam.es/~etc1lab/Archivos_y_pdf/tutorial_descarga_xilinx.pdf. Acceso: Julio 2009

[3] Xilinx University Program Virtex-II Pro Development System - Hardware Reference Manual. Disponible en: http://www.xilinx.com/univ/XUPV2P/Documentation/ug069.pdf. Acceso: julio 2009.