Diseño Digital con Lógica Programable y Lenguajes de Descripción de Hardware

Ing. Arturo J. Miguel de Priego Paz Soldán

www.tourdigital.net
amiguel@pucp.edu.pe

Chincha, Perú
8 de mayo de 2010

Primer Laboratorio

INTRODUCCIÓN A QUARTUS II
DESCRIPCIÓN VHDL, COMPILACIÓN Y SIMULACIÓN DE CIRCUITOS

Objetivos

Al finalizar este módulo el estudiante estará en capacidad de:

- ✓ Utilizar Quartus II Web Edition 9.1 como herramienta CAD para crear proyectos y editar códigos VHDL de circuitos lógicos simples, y luego compilarlos y simularlos para comprobar su funcionamiento.

1. INTRODUCCIÓN

El sistema de desarrollo Quartus II es una plataforma de herramientas para el diseño de circuitos digitales sobre dispositivos FPGA y CPLD de Altera. Quartus II provee aplicaciones para la entrada de diseño, síntesis lógica, simulación lógica, ubicación y conexionado, análisis temporal, administración de potencia y programación de dispositivos, junto con una variedad de utilitarios y aplicaciones adicionales para el diseño lógico programable.

Este laboratorio sirve como una guía de introducción al programa **Quartus II Web Edition v9.1** de Altera Corp. y está lejos de ser una guía completa. Consulta en www.altera.com por más información sobre este programa y sobre otros productos y recursos de Altera.

El propósito de este laboratorio es presentar la interfaz de usuario de Quartus II, y sus características y metodologías de diseño básicas para la descripción VHDL, síntesis y simulación de circuitos digitales.

> *NOTA: Altera Corp. y Xilinx Inc. actualmente son las compañías dominantes en la industria de la lógica programable. Consulta www.xilinx.com para saber más de los productos y servicios de Xilinx.*

El programa para este laboratorio puede descargarse desde el sitio
https://www.altera.com/support/software/download/altera_design/quartus_we/dnl-quartus_we.jsp

Una guía rápida de Quartus II se puede obtener en
www.altera.com/literature/manual/mnl_qts_quick_start.pdf

El manual completo del programa se encuentra en
www.altera.com/literature/hb/qts/quartusii_handbook.pdf

Para aprender interactivamente, revisa las demostraciones en línea de Quartus II en la dirección
http://www.altera.com/education/demonstrations/online/design-software/onl-design-software-demos.html

NOTA: Algunas imágenes de este documento serán distintas a las que aparecerán en el programa. Esto es porque las imágenes se tomaron de una versión de software anterior. Solamente este aspecto visual es diferente. En caso de encontrar otros problemas envía un mensaje electrónico para reportarlos.

El propósito de un diseño en VHDL para síntesis con lógica programable es obtener un circuito lógico sobre un dispositivo de lógica programable a partir de una descripción VHDL. Para ello, se sigue un proceso determinado por un flujo de diseño, donde se indica cada fase del desarrollo del circuito digital.

Repasemos primero los conceptos de VHDL y lógica programable.

Con VHDL podemos definir el funcionamiento de un circuito lógico textualmente con sentencias que definen su interfaz y comportamiento. La descripción del circuito se realiza principalmente con expresiones lógicas, componentes y procesos de señales. El lenguaje permite abstraer las funciones lógicas y codificar en alto nivel los algoritmos de funcionamiento del circuito.

Para conocer más ejemplos de diseño en VHDL, recomendados para Quartus II, revisa www.altera.com/support/examples/vhdl/vhdl.html

Los dispositivos de lógica programable son circuitos integrados que pueden ser configurados o programados para que realicen funciones lógicas especiales, de acuerdo a los requerimientos del diseñador. Estos circuitos integrados contienen puertas lógicas y registros (y en muchos casos memorias) que pueden interconectarse de varias maneras, según lo que el diseñador necesite. En un FPLD el diseñador establece las interconexiones entre circuitos básicos, ya presentes en el circuito integrado del FPLD, para formar circuitos más complejos eligiendo cómo interconectar estos circuitos básicos entre sí. Las interconexiones se realizan haciendo conducir o deshabilitando interruptores electrónicos, que actualmente suelen ser memorias EEPROM o SRAM.

En este laboratorio realizaremos la síntesis del circuito lógico sobre un EPM7128SLC84-15, de la familia MAX7000S. Las hojas de datos de este chip se encuentran en www.altera.com/literature/ds/m7000.pdf

Las herramientas CAD se encargan de realizar la transformación de las descripciones VHDL en los patrones de programación de los interruptores del FPLD, y también de las programaciones de las memorias de datos si fuera el caso.

2. FLUJO DE DISEÑO

Cuando se diseñan circuitos digitales con VHDL como entrada de diseño y con dispositivos de lógica programable como objetivo, el flujo típico de diseño, utilizando herramientas CAD, cubre las siguientes tareas:

1. Primero se ingresa el código VHDL que representa la descripción del funcionamiento del circuito. El programa provee plantillas de códigos comunes que ayudan a reducir el número de errores de sintaxis.
2. Luego se compila la descripción para llegar a obtener un archivo de configuración o programación del FPLD.
3. Usualmente, se simula la lógica y la temporización del circuito, así como se realizan varios tipos de análisis incluyendo potencia eléctrica y riesgos lógicos.
4. Finalmente, se programa y verifica eléctricamente que el FPLD funcione de acuerdo a las especificaciones.

En esta guía veremos compilación a partir de código VHDL y simulación de los circuitos lógicos obtenidos.

El compilador, **Compiler**, consta de varios módulos, entre los que se encuentran:

1. **Analysis and Synthesis** (análisis y síntesis). Analiza el diseño, construye una base de datos, optimiza el diseño para el dispositivo elegido, y mapea la tecnología para el diseño lógico.
2. **Fitter** (organizador y armador). Construye el circuito lógico sobre el dispositivo seleccionado para el proyecto. Coloca y conecta el diseño.
3. **Assembler** (ensamblador). Crea una imagen de programación del dispositivo para configura el dispositivo.
4. **Timing Analyzer** (analizador de tiempos). Analiza los retardos y temporizaciones del circuito.
5. **Design Assistant** (asistente de diseño). Revisa la confiabilidad del diseño basado sobre un conjunto de reglas de diseño. Es un módulo opcional.

Puedes ver el flujo de diseño en operación en la ventana **Tasks** del programa. Cada módulo puede ser configurado de acuerdo a los objetivos de diseño. También, algunos módulos pueden ser deshabilitados si no se requieren por el momento.

El simulador, **Simulator**, determina los valores de las funciones de salida de acuerdo a los valores de las señales de entrada y estados de las señales internas del circuito. Permite depurar la lógica del circuito así como conocer los retardos de propagación de las señales por el circuito. Esta herramienta ayuda a reducir el tiempo de diseño.

En las secciones 3 a 8 de esta guía realizaremos un flujo de diseño básico que comprende las siguientes fases:

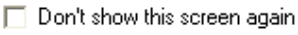
a) Inicio del programa
b) Proyecto de diseño
c) Descripción del circuito
d) Compilación de la descripción VHDL
e) Edición de ondas para la simulación
f) Simulación lógica y temporal

La guía finaliza con un ejercicio de diseño con el propósito de validar el aprendizaje de esta práctica de laboratorio.

3. INICIO DEL PROGRAMA

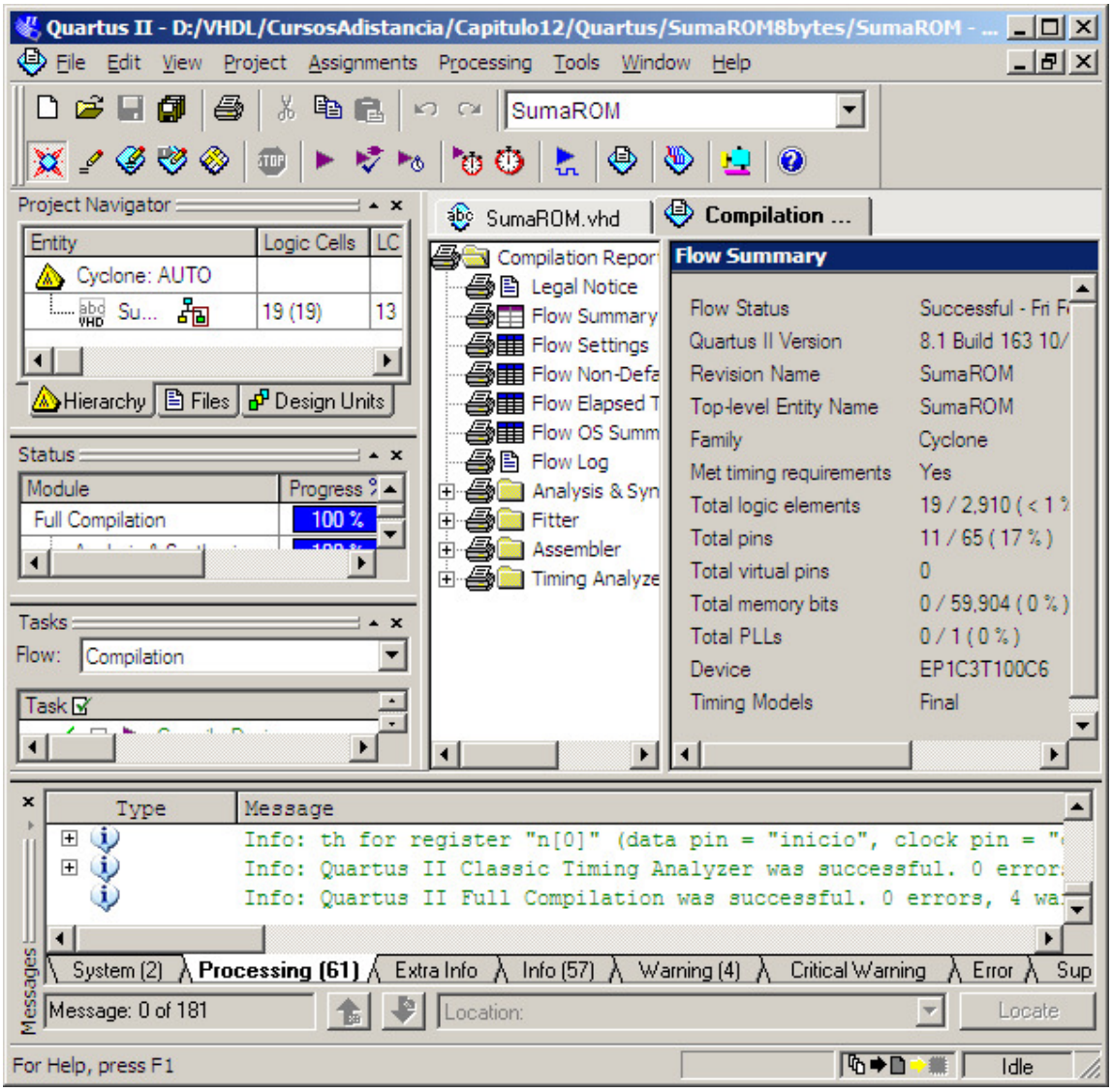
La primera vez que se inicia el programa aparece la ventana siguiente:

Al pulsar el botón  se ejecuta un tutorial interactivo de introducción a Quartus II, el cual presenta en primer lugar una revisión del software, y luego enseña cómo crear y compilar un diseño, ejecutar un análisis y una simulación temporal, y configurar un dispositivo; culmina presentando un par de metodologías de diseño. Además del tutorial puedes usar los enlaces en **Web links:** para conocer más de los recursos de Altera para el diseño con lógica programable.

La ventana anterior aparece la primera vez que el programa se ejecuta. Es posible desactivarla en cada inicio colocando una marca en 

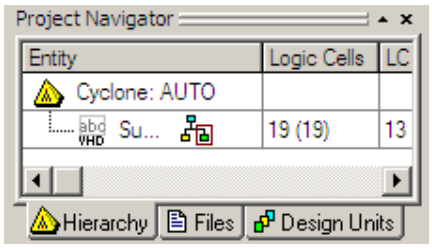
Para continuar, pulsa el botón izquierdo del ratón sobre el aspa en la parte superior derecha de la ventana. Verás el programa de múltiples ventanas de Quartus II.

La interfaz del programa está compuesta por un conjunto de ventanas dedicadas a tareas específicas. La figura siguiente las muestra para un proyecto tomado como ejemplo.

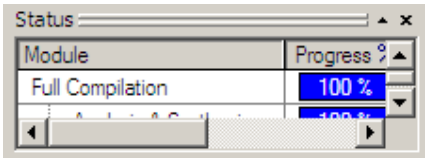


Los menús y barras de herramientas cambian con la aplicación que se está ejecutando.

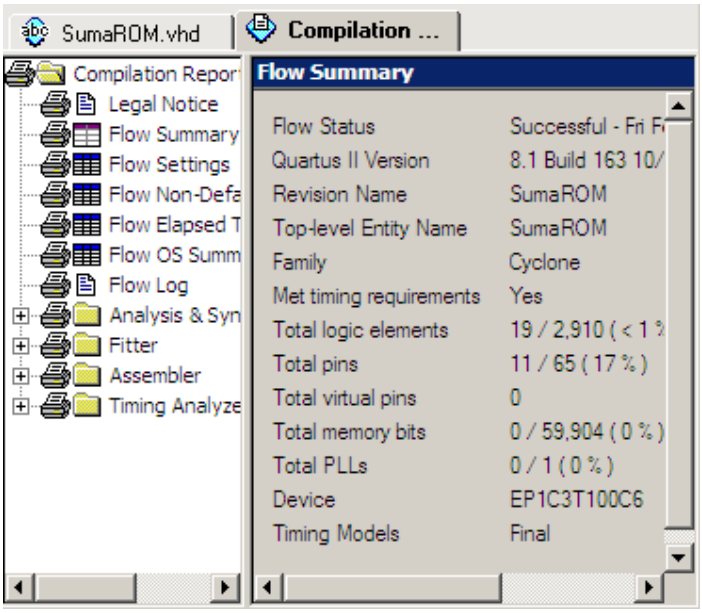
El navegador de proyecto, **Project Navigator**, visualiza la jerarquía del proyecto, los archivos de diseño y comandos para tareas comunes del proyecto. Aquí se colocan también las entidades que el diseñador ingresa o que se encuentran durante la compilación y simulación.



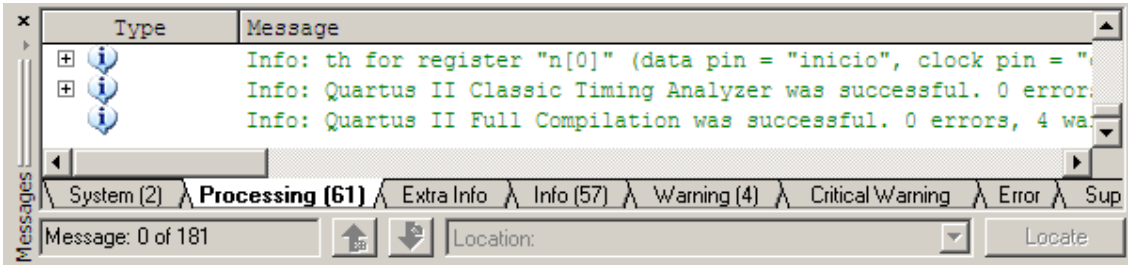
La ventana de estado, **Status**, visualiza el estado de las tareas de procesamiento. Cuando compilas y simulas un diseño, esta ventana visualiza el progreso (como un porcentaje) y el tiempo que toma cada tarea.



En la ventana principal de la aplicación puedes editar archivos y ver los reportes de diseño y otras ventanas. El compilador y el simulador colocan sus reportes en esta sección en ventanas independientes.

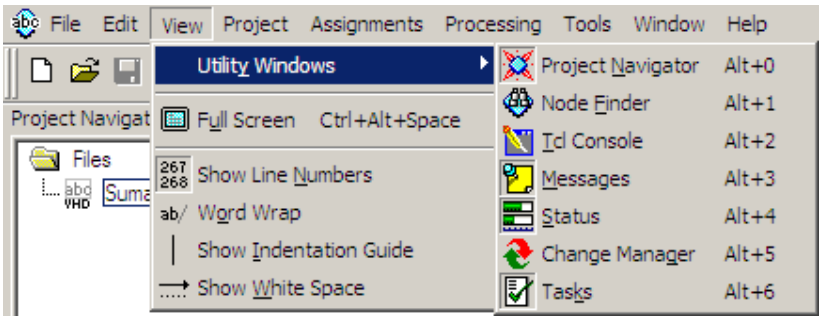


La ventana de mensajes, **Messages**, visualiza mensajes de información, advertencias y errores.



La ventana de tareas, **Tasks,** permite seguir el proceso de diseño. Puedes elegir dos vistas de flujo: **Full Design** (diseño completo) y **Compilation** (solamente compilación). En ellas puedes ver qué actividades se encuentran en curso y cuáles han concluido exitosamente.

Estas ventanas pueden mostrarse u ocultarse según la opción que se seleccione del submenú **Utility Windows** del menú **View**.

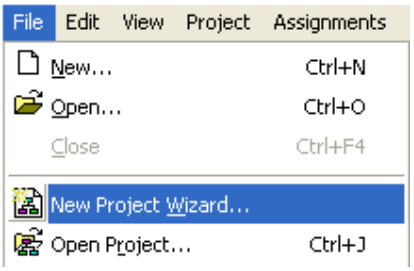


4. PROYECTO DE DISEÑO

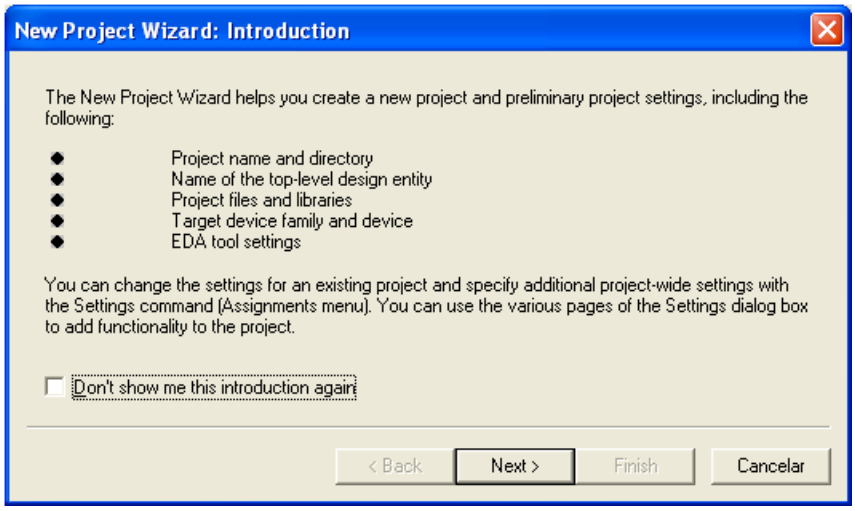
Todos los diseños de circuitos en Quartus II se administran como proyectos. Los proyectos incluyen los archivos de diseño y de configuración junto con otros que Quartus II crea para llevar a cabo sus tareas.

En los pasos siguientes vamos a establecer el nombre del proyecto, el directorio de trabajo, bibliotecas y archivos de proyectos, el dispositivo donde se construirá el circuito y las herramientas EDA adicionales para el proyecto. Algunas opciones disponibles no serán utilizadas por ahora.

1. Para crear un nuevo proyecto elije del menú **File** la opción **New Project Wizard…** Esta aplicación guía paso a paso a configurar un nuevo proyecto.

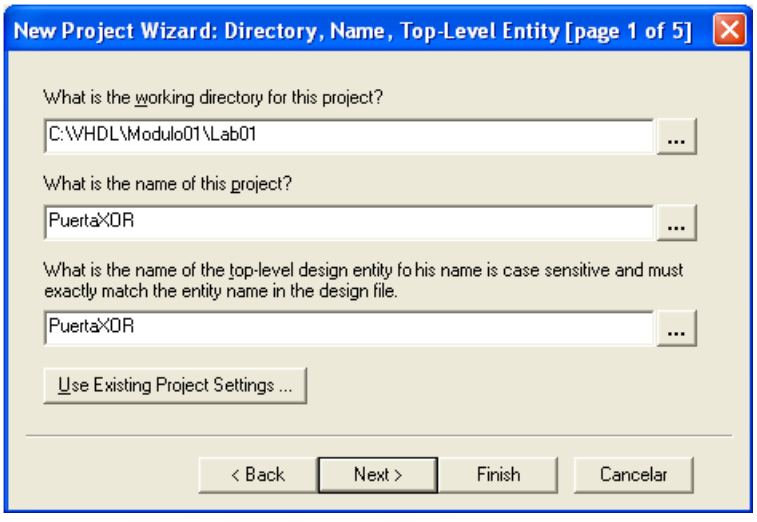


2. La primera vez que ejecute esta aplicación aparece una ventana de introducción. Pulsa en el botón Next > para continuar.

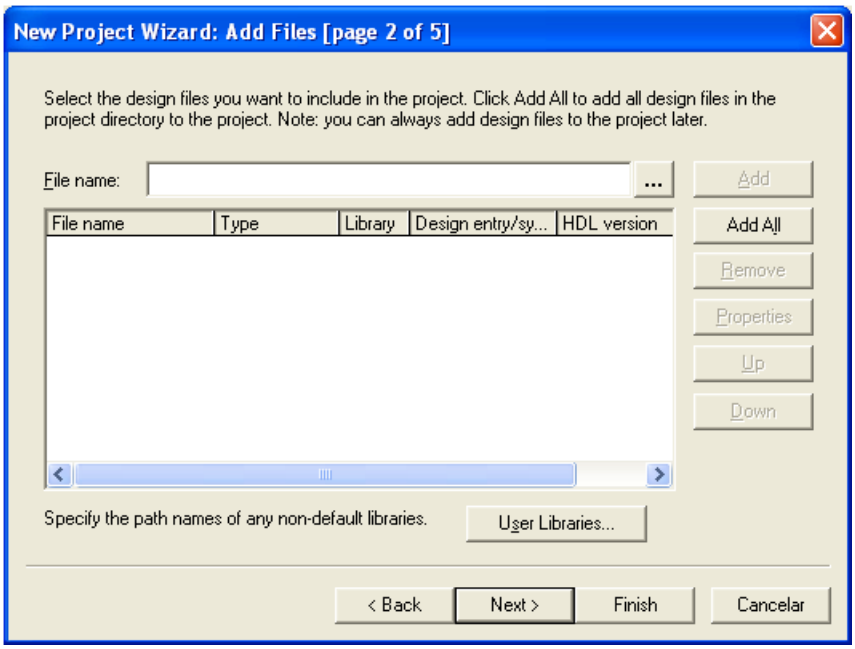


Si colocas una marca en la casilla Don't show me this introduction again *esta ventana de introducción no aparecerá en las siguientes ejecuciones.*

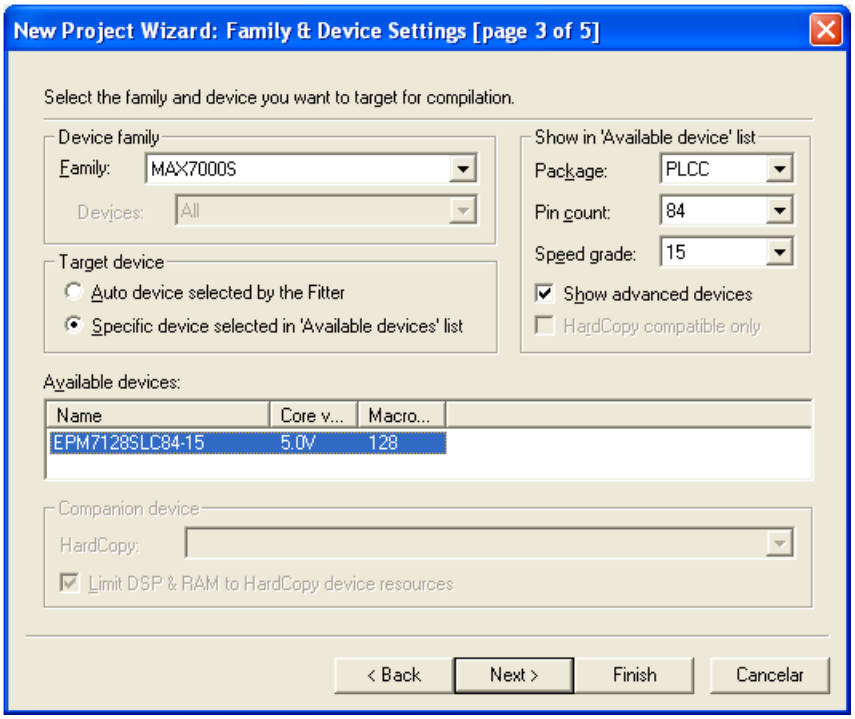
3. Aparecerá una ventana para ubicar el nombre del proyecto y directorios de trabajo. Selecciona un directorio para el proyecto y escribe los nombres del proyecto y del archivo de mayor jerarquía en el proyecto. Como se trata de un proyecto de un archivo, solamente escribe PuertaXOR. Luego pulsa en Next >.



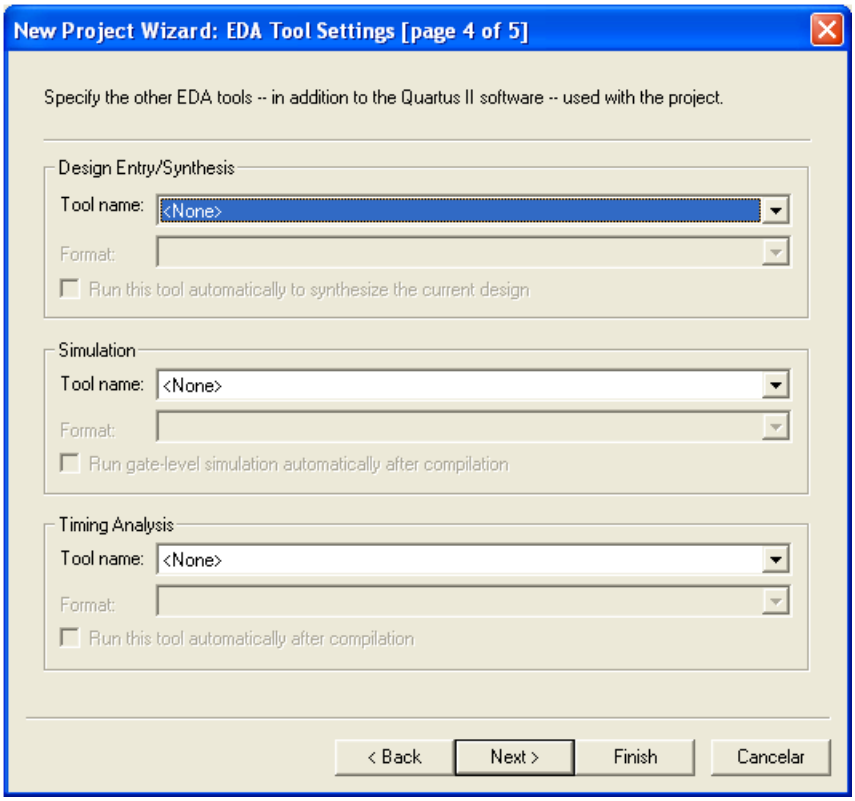
4. En la nueva ventana pueden ingresarse archivos en el proyecto que ya hayan sido creados. Como recién empezamos no tenemos archivos previos, por lo tanto, pulsa en Next >.



5. En la nueva ventana podemos seleccionar el dispositivo sobre el cual se creará el circuito lógico. Usaremos el EPM7128SLC84-15 que pertenece a la familia MAX7000S. Coloca los valores mostrados en la figura y luego pulsa en Next >.

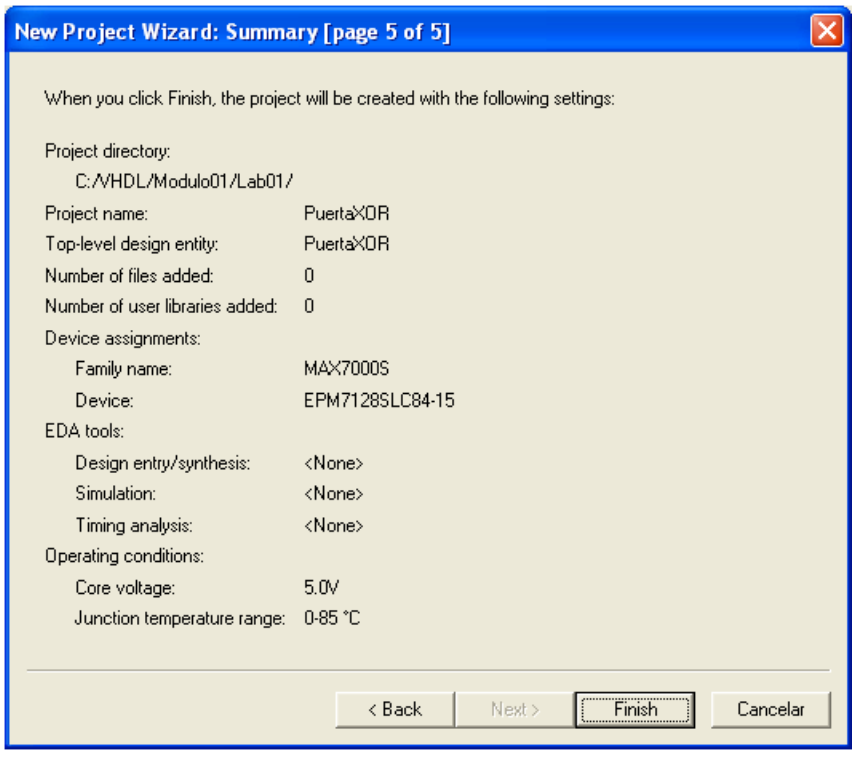


6. En esta ventana se seleccionan las herramientas EDA de otros programas. Utilizaremos las propias herramientas de Quartus por el momento. Pulsa en Next >.



7. Finalmente, aparece el resumen del proyecto creado. Pulsa en Finish.

   El proyecto está listo y ahora crearemos el archivo de entrada de diseño con VHDL.

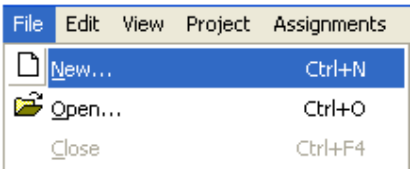


Después de crear un proyecto puedes modificar los parámetros previos y establecer más detalles de configuración para el proyecto. Para ello, elige ***Settings…*** *del menú* ***Assignments*** *y establece los nuevos parámetros y configuraciones navegando sobre las opciones de la ventana.*

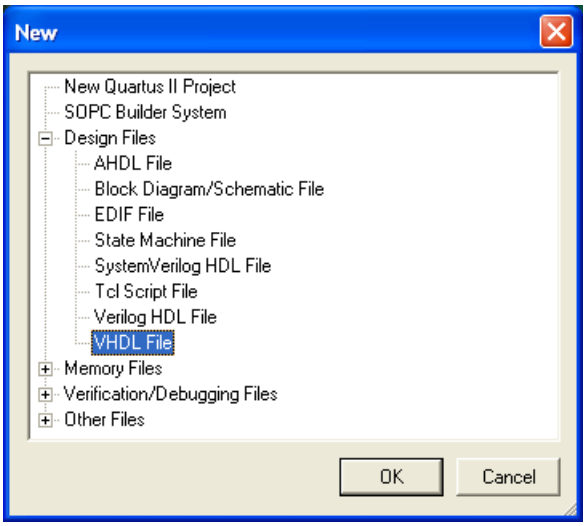
5. DESCRIPCIÓN DEL CIRCUITO

Ahora crearemos un archivo para la descripción del circuito en VHDL.

1. Del menú **File** elige **New..**, o presiona juntas las teclas Ctrl y N (primero presiona la tecla Ctrl), o pulsa sobre el botón de la barra de herramientas. Aparecerá una ventana para elegir el tipo de archivo.



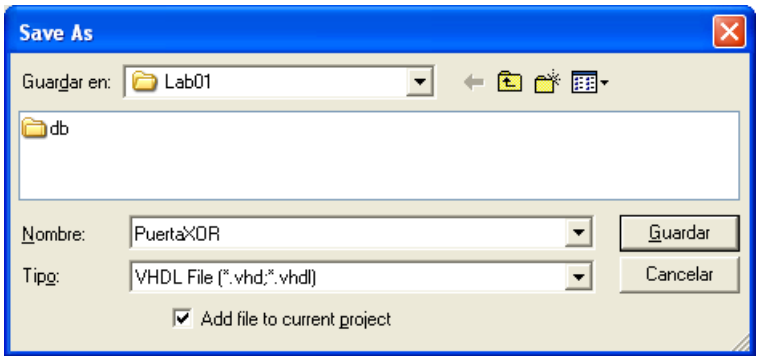
2. En la ventana selecciona **VHDL File** del grupo **Design Files**. Luego pulsa en OK.



3. Del menú **File** elige **Save**, o presiona la tecla Ctrl y con ella la tecla S, o pulsa sobre el botón . Guarda el archivo con el nombre PuertaXOR.

   .

   Mantén activada la opción Add file to current project para que el archivo pase a formar parte del proyecto.



Guarda periódicamente los cambios hechos al archivo.

4. Escribe el texto VHDL que aparece en la página siguiente. Omite o edita los comentarios a voluntad. Recuerda guardar periódicamente los cambios que realices.

```
2  ----------------------------------------
3  -- PuertaXOR.vhd
4  ----------------------------------------
5  -- Modelo de una puerta OR exclusiva
6  -- con asignaciones simples
7  -- f <= a XOR b;
8  ----------------------------------------
9  -- Arturo J. Miguel de Priego
10 -- www.tourdigital.net
11 -- amiguel@pucp.edu.pe
12 -- Chincha, Perú, 24 de febrero de 2009
13 ----------------------------------------
14
15 library ieee;
16 use ieee.std_logic_1164.all;
17
18 entity PuertaXOR is
19    port
20    (
21       a  : in std_logic;
22       b  : in std_logic;
23       f  : out std_logic
24    );
25 end entity;
26
27 architecture rtl of PuertaXOR is
28 begin
29
30    f <= a xor b;
31
32 end rtl;
```

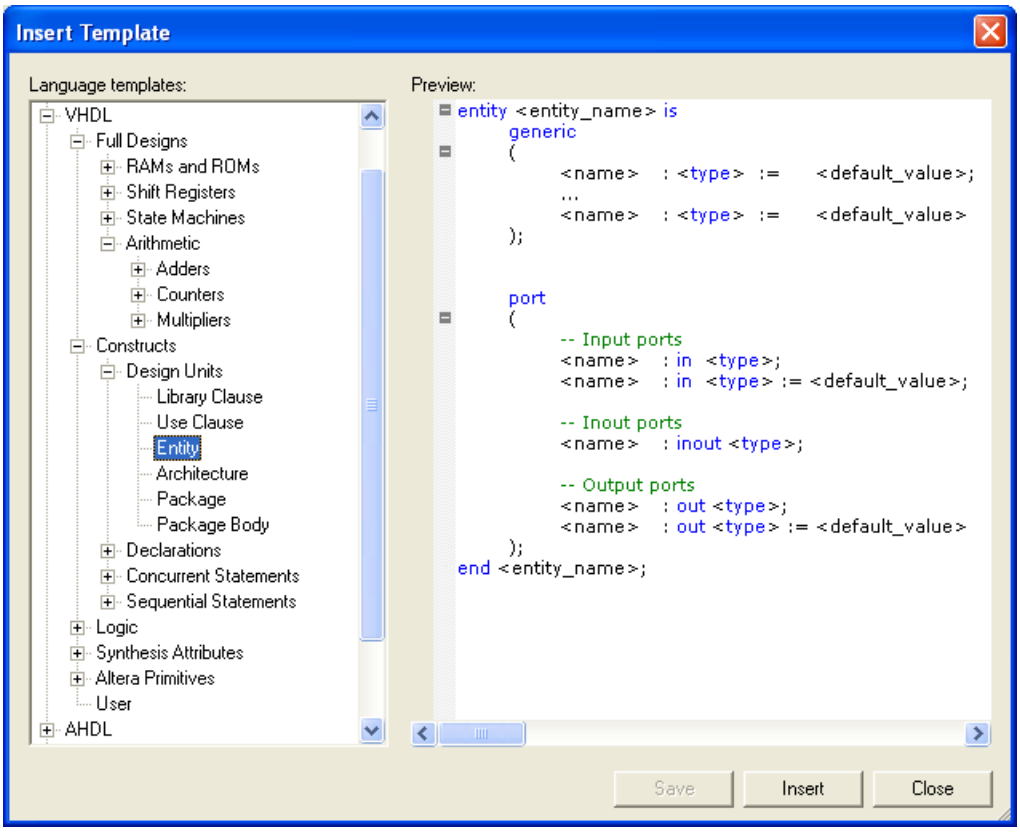
Puedes utilizar plantillas para facilitar la edición del texto y reducir los errores de sintaxis. Del menú **Edit** elige la opción **Insert Template…**

Expande el grupo **VHDL** y explora las plantillas que te ofrece el programa. En la figura de abajo se muestra una plantilla para la entidad.

Una vez que hayas seleccionado una plantilla pulsa sobre el botón Insert para insertar el texto en tu archivo VHDL.

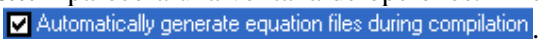
Ubica en tu archivo VHDL el lugar para insertar otra plantilla, vuelve a la ventana de plantillas y selecciona la plantilla para la arquitectura y vuelve a pulsar en Insert.

Cuando hayas terminado de ingresar las plantillas presiona en Close.

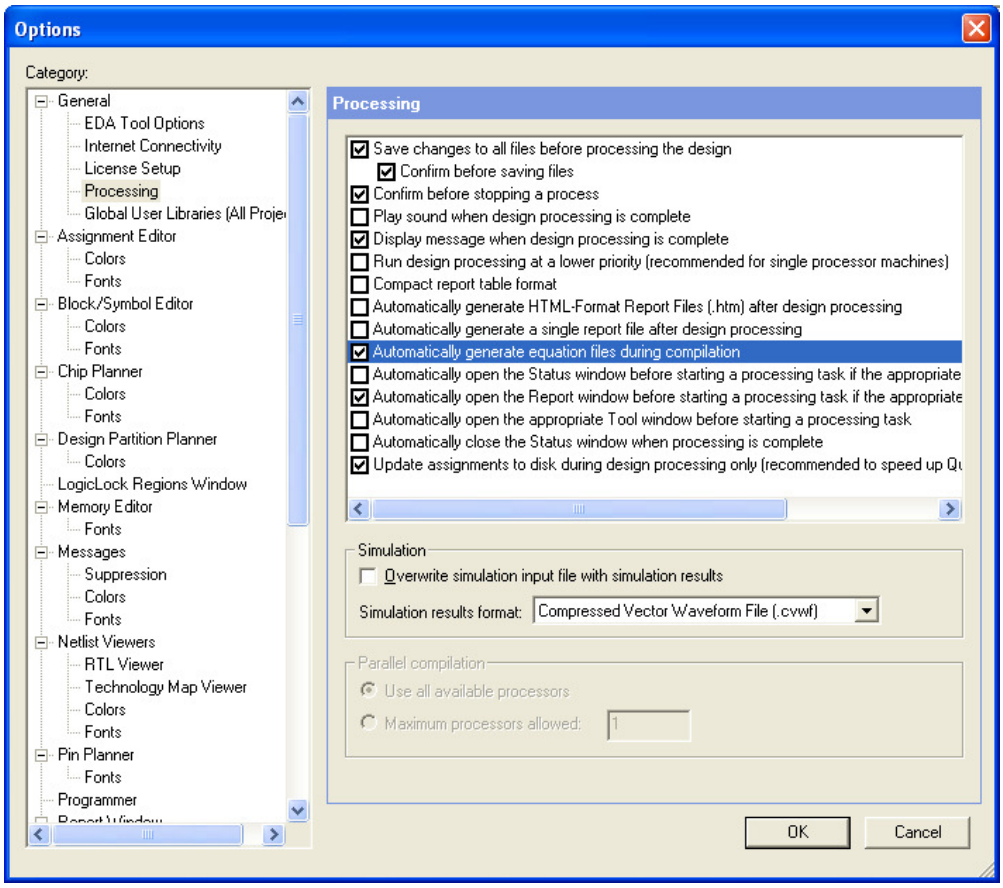


6. COMPILACIÓN DE LA DESCRIPCIÓN VHDL

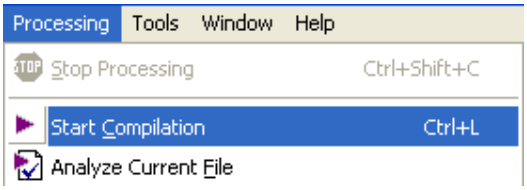
En la fase de compilación se verifica que el texto VHDL cumpla con la sintaxis y semántica del lenguaje. Además, el compilador crea un conjunto de archivos que representan versiones lógicas del texto VHDL.

Las ecuaciones generadas por el compilador pueden verse en el reporte del compilador. Para activar esta opción, del menú **Tools** selecciona **Options…** Aparecerá una ventana de opciones. En el grupo **General**, elige **Processing**. Activa la opción 

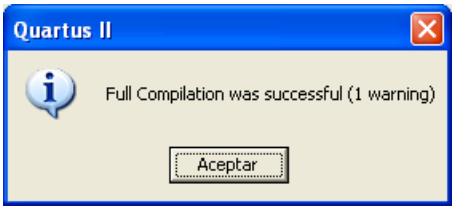
. Luego pulsa en el botón **OK**.



Ahora compilaremos el diseño VHDL. Del menú **Processing** selecciona **Start Compilation**. También puedes presionar simultáneamente las teclas Ctrl y L, o pulsar sobre  .

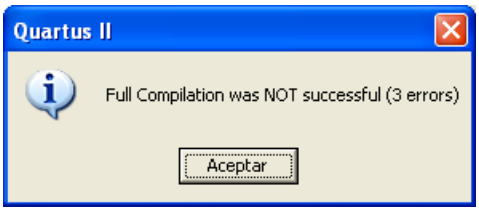


Si el código está correcto aparecerá el siguiente mensaje:

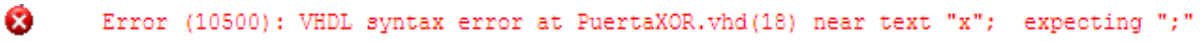


Si ocurren errores, el compilador los menciona en la barra de mensajes. Si al compilar aparecen errores, revisa que hayas escrito el código VHDL como se indicó en la página 8.

Como ilustración, mostraremos un ejemplo de corrección de errores. En la línea 30 del texto VHDL (página 8) separa la palabra XOR en X OR. Luego graba y compila. Aparecerá un mensaje notificando que la compilación encontró errores.

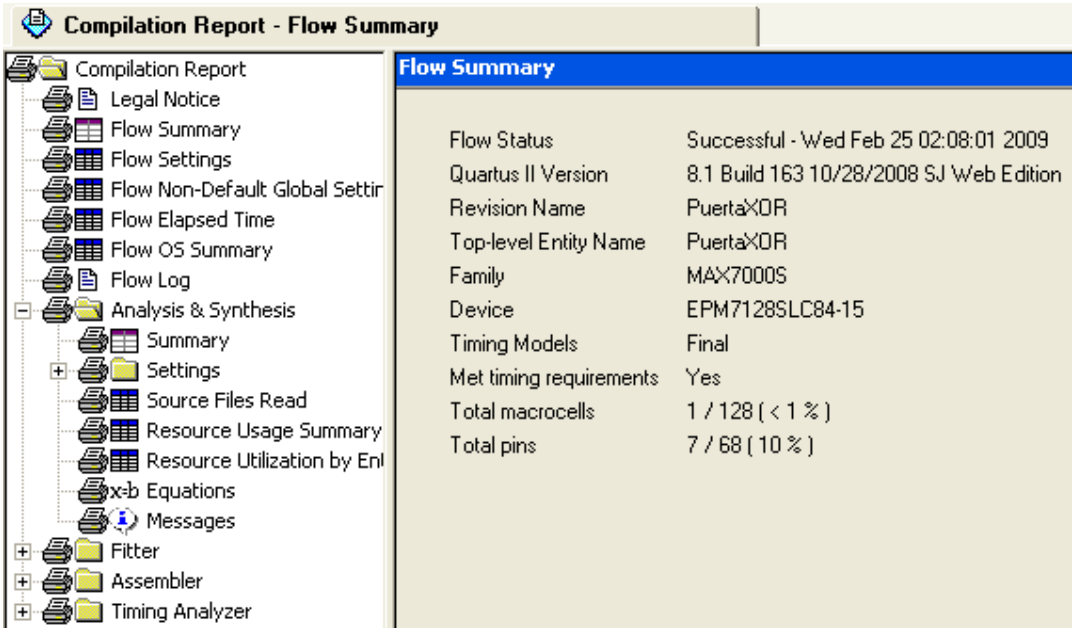


Los errores aparecen descritos en la ventana de mensajes. Con el botón izquierdo, rápidamente pulsa dos veces sobre el primer error.



Aparecerá la ventana del texto VHDL y se resaltará la línea conteniendo el error. Corrígelo y vuelve a compilar.

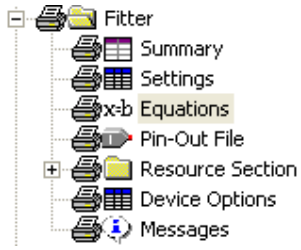
Puedes consultar el reporte cuando la compilación finalice con éxito. El resumen del proceso (**Flow Summary**) del reporte de compilación (**Compilation Report**) se muestra en la figura siguiente:



Nota que todos los pasos que el compilador realiza pueden ejecutarse individualmente dependiendo de la información que se quiera obtener.

Observa que el diseño utiliza siete pines (de 68 en total) y una macrocelda (de 128 en total). El diseño tiene dos entradas y una salida, es decir, tres pines, entonces ¿Por qué el compilador dice que se utilizan siete pines? Sucede que está sumando cuatro pines que se emplean para la programación del circuito a través del estándar JTAG.

Puedes conocer detalles de la compilación seleccionando los temas del reporte.

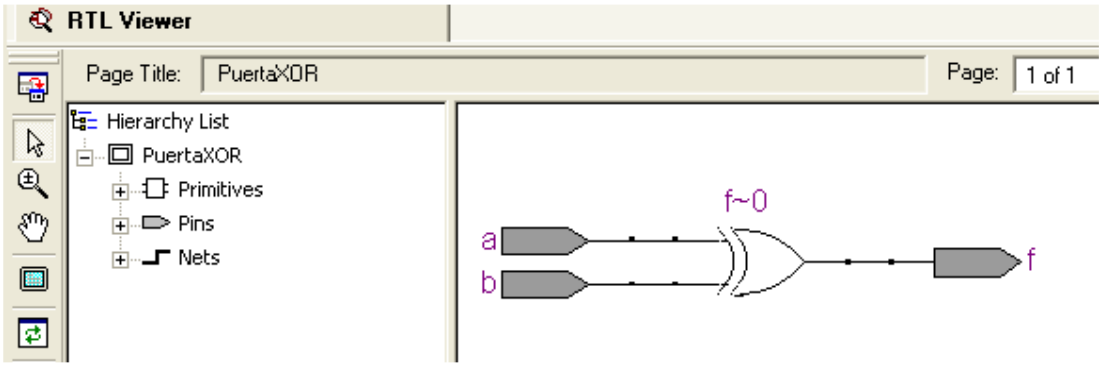


Por ejemplo, las expresiones lógicas finales determinadas por el compilador puedes verlas en la sección **Equations** de **Fitter**.

```
15   A1L4_or_out = b;
16   A1L4 = a $ A1L4_or_out;
17
18
19   --a is a at PIN_52
20   --operation mode is input
21
22   a = INPUT();
23
24
25   --b is b at PIN_33
26   --operation mode is input
27
28   b = INPUT();
29
30
31   --f is f at PIN_12
32   --operation mode is output
33
34   f = OUTPUT(A1L4);
```

En la línea 34 se observa que f es la expresión A1L4 que aparece en la línea 16, que a su vez, remplazando 15 en 16, es a $ b. El símbolo $ indica la operación XOR. Los símbolos &, # y ! representan las funciones AND, OR y NOT respectivamente. Estos símbolos son usados en el reporte del compilador, no forman parte del código VHDL.

Otra aplicación útil es el visualizador del circuito RTL (*Register Transfer Level*), que es un diagrama de circuito esquemático equivalente o muy cercano a la descripción VHDL. Para ver este circuito RTL, del menú **Tools** elige el submenú **Netlist Viewers** y luego la opción **RTL Viewer**.



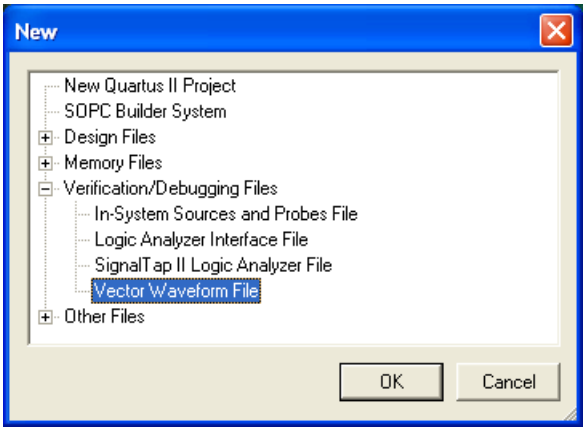
Ten en cuenta que este circuito sirve para comprobar que los circuitos descritos con VHDL sean los esperados por el diseñador. No siempre es el circuito lógico que el compilador determina para construir en el FPLD. Como muestra, modifica la descripción VHDL para la función f escribiendo f <= (a and not b) or (not a and b); y luego de compilar observa el circuito RTL determinado por el compilador.

El circuito que el compilador finalmente obtiene para el FPLD es un circuito optimizado a partir de las restricciones y parámetros del proyecto.

7. EDICION DE ONDAS PARA LA SIMULACION

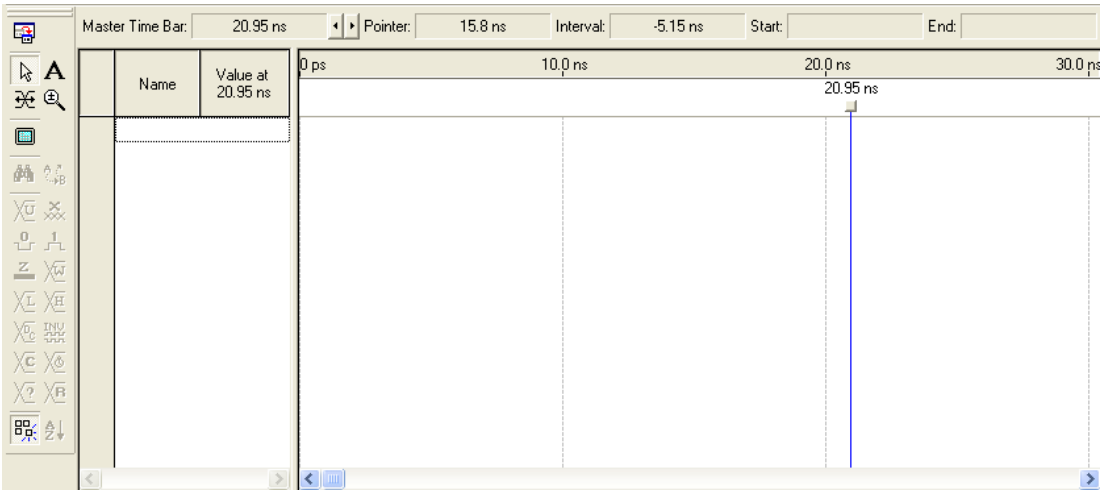
Ahora crearemos un archivo de ondas de simulación para comprobar el funcionamiento del circuito lógico XOR. Indicaremos solamente los vectores de prueba de las entradas. El simulador obtendrá los resultados que podrán observarse igualmente, de manera gráfica, en la ventana de ondas.

1. Del menú **File** elige **New…** o presiona juntas las teclas Ctrl y N o pulsa sobre el botón  . En la ventana que aparece elige **Vector Waveform File** y luego presiona en **OK**.

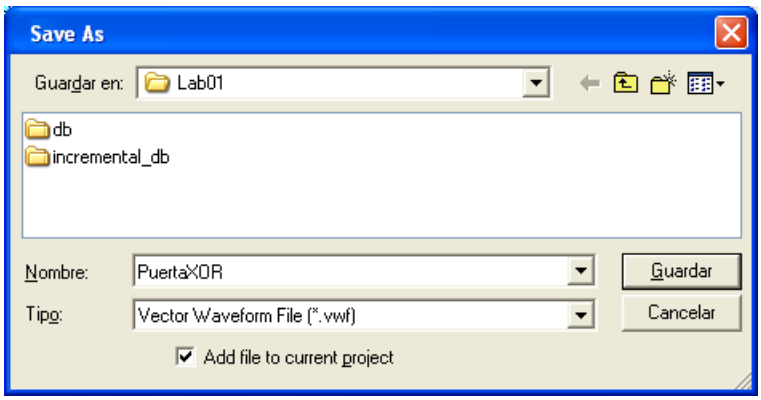
   Aparecerá una ventana para insertar las señales y sus patrones de onda. Al lado izquierdo se encuentra una barra de herramientas para la edición de ondas. En el centro se insertan las señales y en la parte derecha se forman las ondas.

   En la parte superior, los casilleros muestran datos de tiempos e intervalos; ellos cambian automáticamente con la posición del cursor de línea azul y del cursor del ratón.

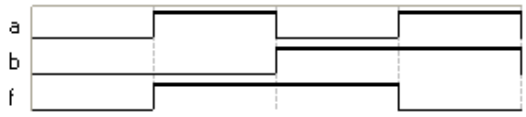


NOTA: *Es posible liberar la ventana de simulación del entorno de ventanas del programa. Si así es más confortable, en el menú* ***Window*** *elige* ***Detach Window*** *o pulsa sobre el botón*  *de la barra de herramientas vertical situada al lado izquierdo de las señales.*

2. Guarda el archivo con el mismo nombre del proyecto. Nota que el tipo de archivo es **vwf** por defecto. Así debe mantenerse.

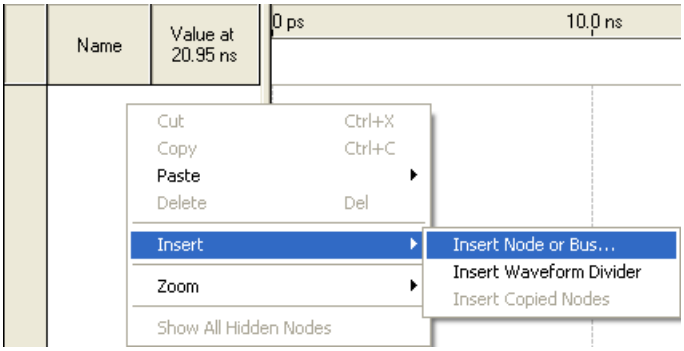


A continuación, formaremos el patrón de ondas para las señales a y b. El simulador obtendrá una respuesta para f, como se muestra en la figura de la derecha.

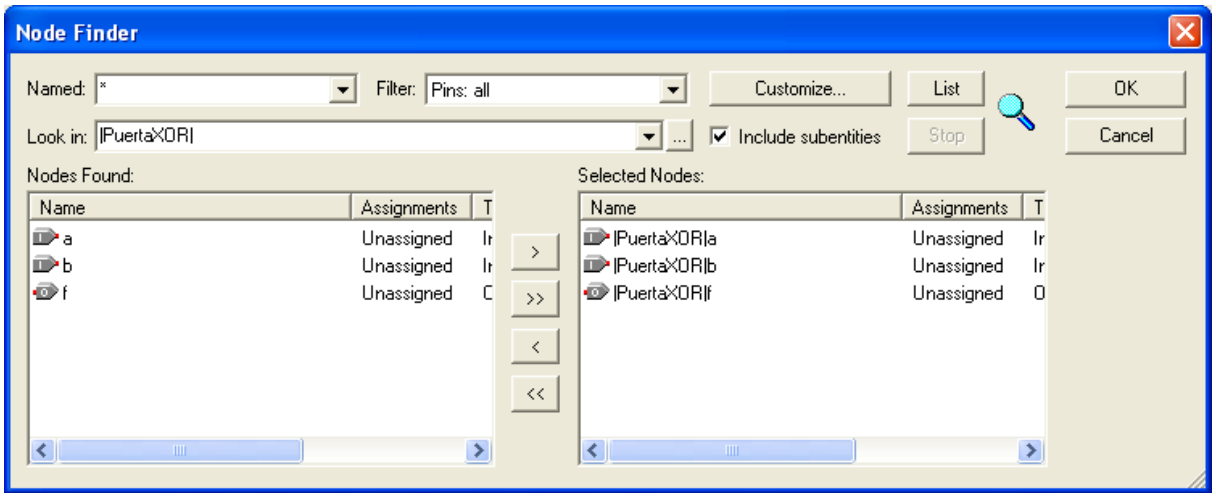


El intervalo mínimo será de 100 ns. De este modo, cuatro combinaciones requerirán de 400 ns de simulación. Estas combinaciones se repetirán una vez, lo cual dará un tiempo total de 800 ns de simulación. Ahora comenzaremos a editar el archivo.

3. Bajo el campo **Name** del editor de ondas, pulsa el botón derecho del ratón. Aparecerá un menú flotante. Selecciona **Insert** y luego **Insert Node o Bus…** Aparecerá una ventana para insertar los nombres de las señales. Pulsa sobre el botón Node Finder... para buscar los nombres.



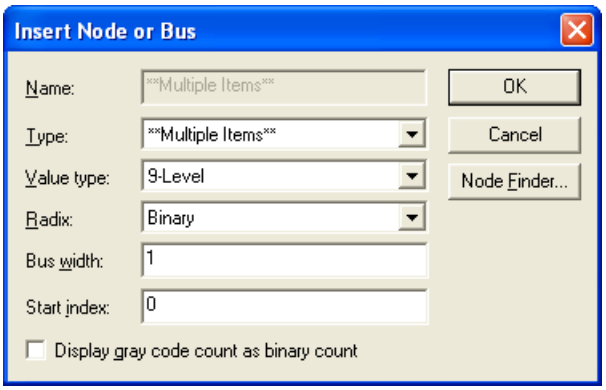
4. Aparecerá una ventana para filtrar y seleccionar las señales que se insertarán en la simulación. En la lista desplegable **Filter** selecciona la opción **Pins: all,** Filter: Pins: all y luego pulsa en el botón List. Selecciona todas las entradas y salidas pulsando en el botón >> y después pulsa en OK.



Nota: *Puedes usar* > *o* < *para transferir señales una por una.*

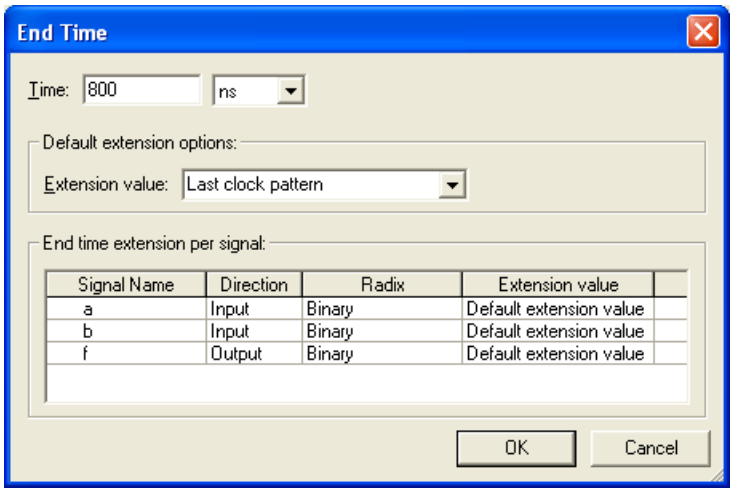
5. En la ventana que aparece, selecciona la representación de valores en binario eligiendo **Binary** en la lista desplegable **Radix**,

Luego pulsa en 



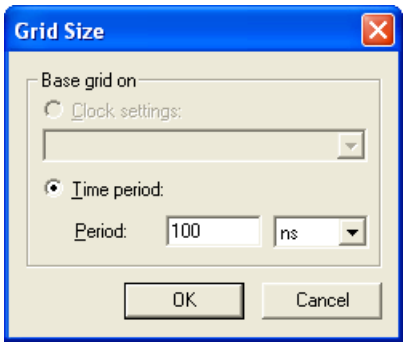
6. Para fijar un tiempo de simulación total de 800 ns, selecciona **End Time…** del menú **Edit**.

Luego, en la casilla **Time** especifica **800 ns**.



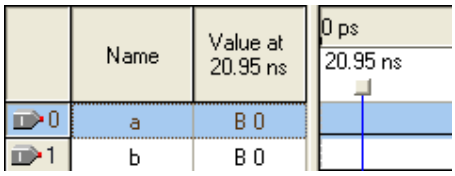
7. Cambiaremos los valores en múltiplos de períodos de 100 ns. Selecciona **Grid Size…** del menú **Edit**.

Luego, especifica el período como **100 ns**.

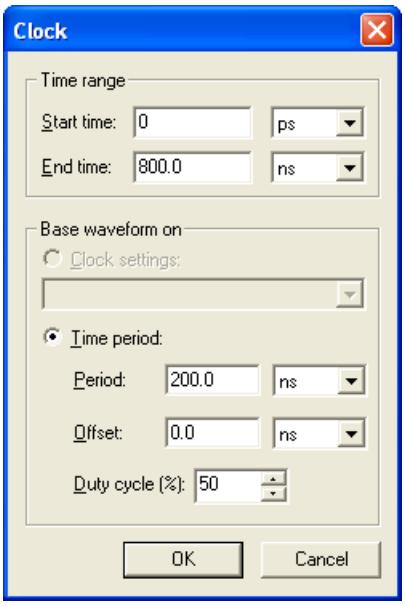


*NOTA: Podemos hacer que el tiempo de simulación se vea completo en la ventana. Selecciona **Fit in Window** del menú **View**, o también mantén presionada la tecla Ctrl mientras pulsas la tecla W.*

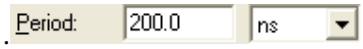
8. Ahora comenzaremos a dar valores a las señales por períodos de tiempo. Formaremos en primer lugar la señal **a** como una señal periódica de 200 ns de período con ciclo de trabajo de 50%, es decir, 100 ns en baja y 100 ns en alta.



Selecciona el canal de la señal **a** pulsando sobre el botón  , al lado izquierdo de su nombre.



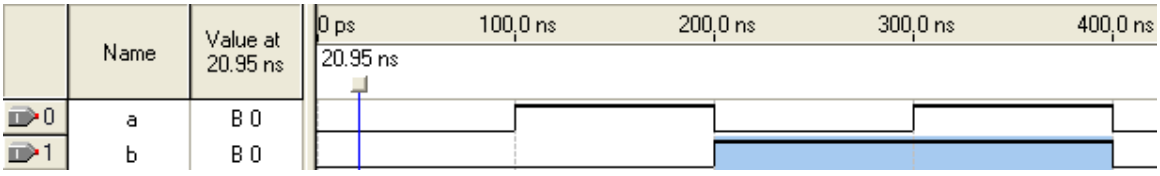
En la barra de herramientas selecciona el botón de reloj,  . Aparecerá una ventana para crear un patrón de onda periódico. Especifica un período de 200 ns.



Luego pulsa en 

El patrón de ondas de la señal **b** puede obtenerse de manera similar, mas usaremos otro método.

Presiona el botón izquierdo del ratón sobre la intersección del canal **b** con la línea de tiempo de 200 ns. Sin dejar de presionar el botón, arrastra el ratón sobre el canal **b** hasta la línea de 400 ns y libera el botón. Pulsa sobre el botón  de la barra de herramientas. El período entre 200 y 400 ns quedará en 1. De igual manera, coloca la señal **b** en alta entre 600 y 800 ns. Recuerda guardar periódicamente los archivos que estés editando.

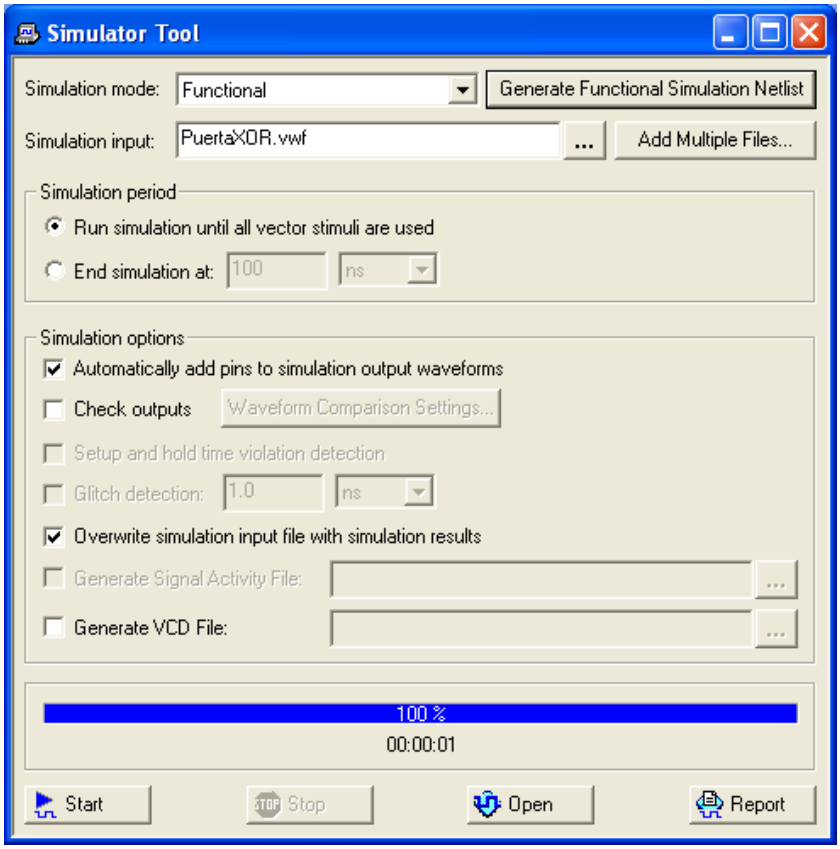


8. SIMULACIÓN LÓGICA Y TEMPORAL

Vamos a simular este circuito de dos maneras diferentes. La primera será una simulación funcional y la segunda una simulación con modelos de retardos llamada simulación temporal, más cercana a la realidad. La simulación funcional sirve para comprobar la lógica del circuito, sin retardos en la propagación de las señales, mientras que la simulación temporal permite conocer los retardos máximos que pueden presentarse cuando se las señales se transmiten de un punto a otro del circuito.

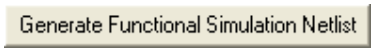
Para realizar una simulación funcional lleva a cabo los siguientes pasos:

1. Del menú **Processing** elige **Simulator Tool**. Aparecerá la ventana de configuración de la simulación.



Activa la actualización automática de resultados en el archivo de entrada.

Fija el modo de simulación funcional en 

y seguidamente presiona en el botón 

para generar los archivos de la simulación funcional.

Cuando el proceso culmine aparecerá el siguiente mensaje:

Pulsa en Aceptar.

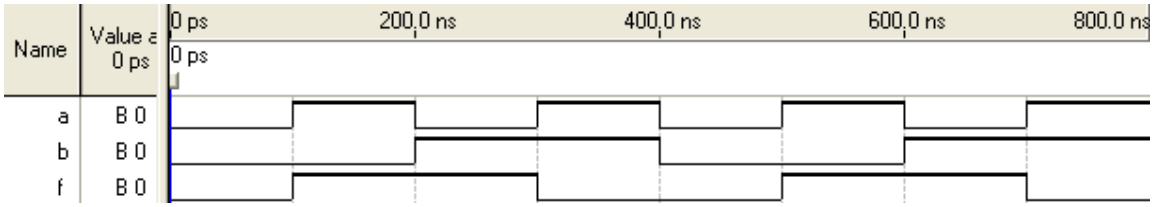
2. Para iniciar la simulación, presiona sobre Start.

   Al culminar el proceso aparecerá el mensaje siguiente:

   Pulsa en Aceptar.

3. Para ver los resultados de la simulación presiona Open. Aparecerá un mensaje para confirmar la actualización del archivo de ondas. Presiona en el botón Sí.

Observa los resultados de la simulación funcional. Nota que la salida cambia inmediatamente con las entradas. Comprueba que la señal f sea la función lógica XOR de las señales a y b en todo momento.



Ahora realizaremos la simulación temporal, que toma en cuenta los modelos de retardos del dispositivo lógico seleccionado para el proyecto.

1. Nuevamente elije **Simulation Tool** del menú **Processing** o trae al frente la ventana del simulador.

   En la ventana del simulador, elige el modo de simulación temporal

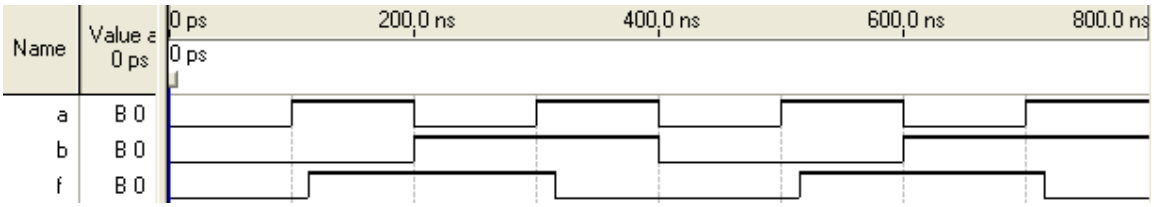
2. Presiona en Start. Comenzará la simulación temporal.

   Al culminar el proceso aparecerá el mensaje siguiente:

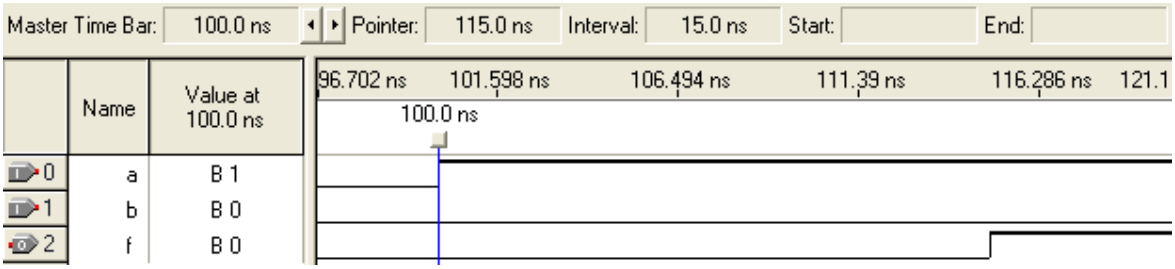
   Pulsa en Aceptar.

3. Para ver los resultados de la simulación presiona Open. Aparecerá un mensaje para confirmar la actualización del archivo de ondas. Presiona en el botón Sí.

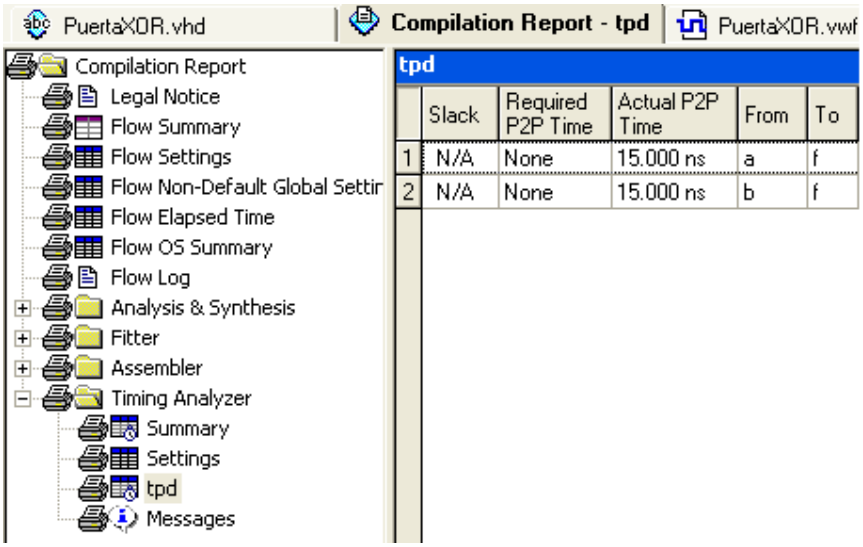
Nota que ahora la señal de salida no cambia inmediatamente; ahora existen retardos.



Presiona el botón  y luego pulsa los botones izquierdo o derecho sobre la zona de ondas para ampliar o reducir la vista de las señales, respectivamente. Ubica los cursores (cursor azul y cursor del ratón) para determinar el retardo, el cual se muestra en la casilla **Interval**. En la figura siguiente se observa que el retardo es de 15 ns:



Este retardo también puede ser conocido sin realizar la simulación temporal. En el reporte de la compilación aparece en la sección **tpd** de **Timing Analizer**.



En sucesivas compilaciones y simulaciones puedes utilizar los botones  *de la barra de herramientas superior para ejecutarlas con las opciones previas.*

Para recuperar proyectos anteriores consulta en el submenú ***Recent Projects*** *del menú* ***File****.*

9. DISEÑO DE UN SUMADOR COMPLETO

En esta parte vas a diseñar un sumador completo con entradas {a, b, cin} y salidas {s, cout}.

1. Nombra tu proyecto como SumadorCompleto. Utiliza un directorio diferente al diseño anterior.

2. Describe el circuito con expresiones lógicas. Incluye comentarios.

3. Compila para el mismo dispositivo del ejemplo anterior. Escribe un resumen de la compilación y agrégalo como comentario en el archivo VHDL.

4. Analiza el circuito RTL. Incluye tus comentarios en el archivo VHDL.

5. Realiza la simulación lógica. Luego haz una copia de SumadorCompleto.vwf como SumadorCompletoLogico.vwf. Agrega tus comentarios sobre la simulación en el archivo VHDL.

6. Haz una simulación temporal. Compárala con la simulación lógica y agrega tus comentarios en el archivo VHDL.

Envía por email los siguientes archivos:

a) Archivo de proyecto : SumadorCompleto.qpf
b) Archivo de opciones del proyecto : SumadorCompleto.qsf
c) Archivo de descripción VHDL : SumadorCompleto.vhd
d) Archivo de simulación lógica : SumadorCompletoLogico.vwf
e) Archivo de simulación temporal : SumadorCompleto.vwf