

P3

Introducción a Vivado IDE, parte I: Simulación

Objetivos

En la tercera práctica, el alumno entenderá las funcionalidades básicas del ambiente de desarrollo Vivado, para el desarrollo de firmware para FPGAs en el lenguaje VHDL, incluyendo la captura del código fuente, simulación, síntesis, verificación y programación del FPGA.

Planteamiento

En un programa de concursos de televisión, hay un juego en el que participan 3 personas. Cada participante cuenta con un interruptor; los interruptores cerrados (posición "ON") generan un "uno" lógico y los interruptores abiertos (posición "OFF") generan un "cero" lógico. Por otra parte, hay 4 indicadores (LEDs) en un tablero: A, B, C y D. Un indicador enciende si recibe un "uno" lógico. Tomar en cuenta las siguientes consideraciones:

- El indicador A debe encender sólo cuando todos los jugadores posicionen en "ON" sus interruptores.
- El indicador B debe encender si y sólo si A está en "OFF".
- El indicador C debe encender si y sólo si dos o más jugadores posicionan en "ON" sus interruptores.
- El indicador D debe encender si y sólo si uno o ninguno de los jugadores posicionan en ON sus interruptores.

Se generó la siguiente tabla de verdad y se encontraron las ecuaciones booleanas correspondientes:

P1	P2	P3	A	B	C	D
0	0	0	0	1	0	1
0	0	1	0	1	0	1
0	1	0	0	1	0	1
0	1	1	0	1	1	0
1	0	0	0	1	0	1
1	0	1	0	1	1	0
1	1	0	0	1	1	0
1	1	1	1	0	1	0

- $A = P1 \cdot P2 \cdot P3$
- $B = P1' + P2' + P3'$
- $C = P2 \cdot P3 + P1 \cdot P3 + P1 \cdot P2$
- $D = P1' \cdot P2' + P1' \cdot P3' + P2' \cdot P3'$

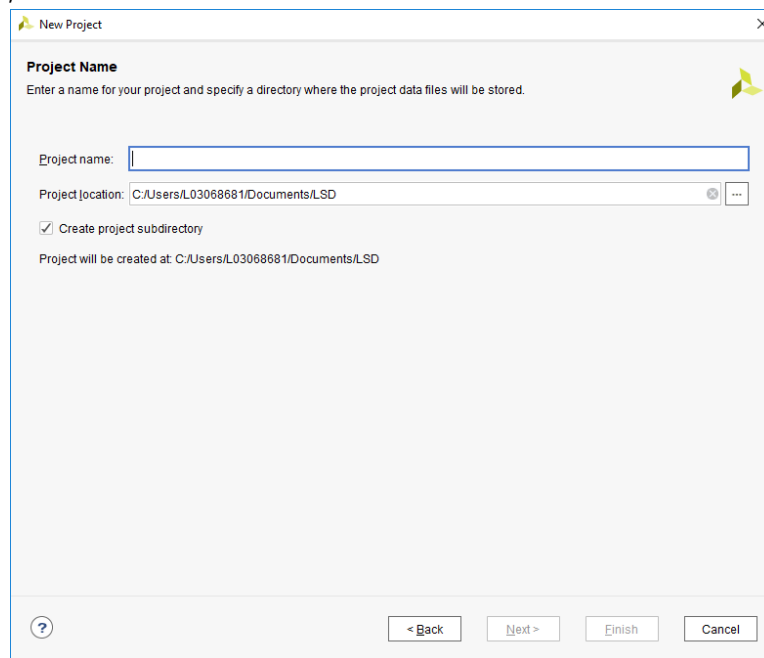
Resolver el problema implementando un programa en VHDL que resuelva la lógica necesaria para obtener las respuestas en las cuatro salidas a partir de las posibles combinaciones de entradas.

Procedimiento

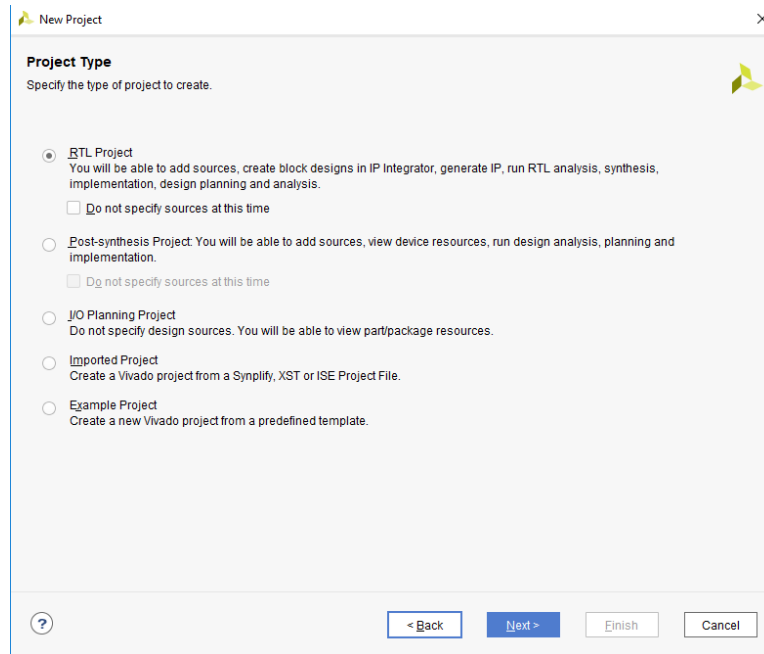
1. Ejecutar Vivado 2018.2 (o versión más reciente)
2. En la pantalla de inicio, seleccionar "Create Project" para crear un proyecto nuevo.



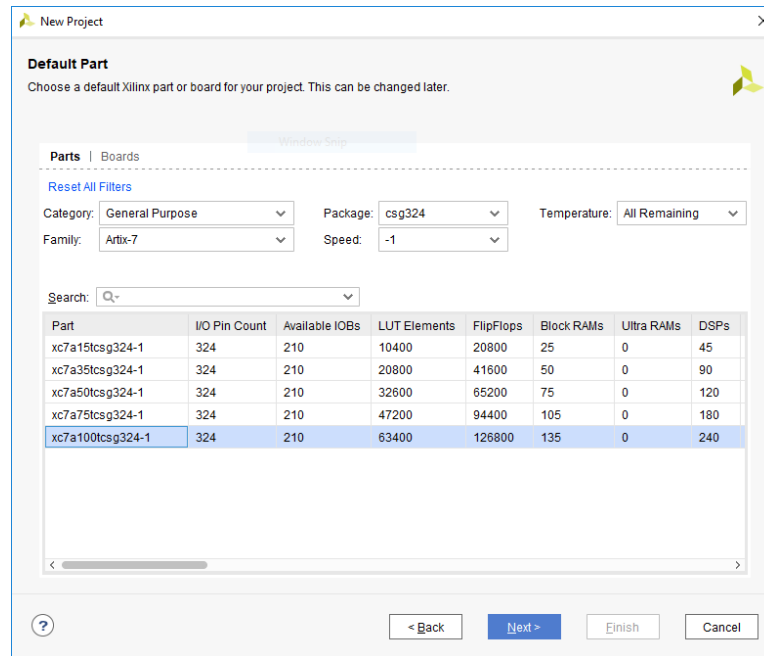
3. Dar un nombre al proyecto y seleccionar el lugar apropiado para guardar el proyecto. Se recomienda que para este momento ya tenga una carpeta creada especialmente para el laboratorio, con la finalidad de mantener ordenados sus archivos. Hacer clic en "Next".



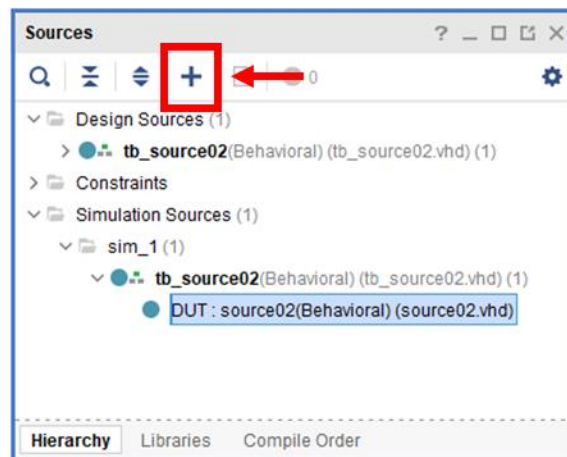
4. Seleccione el tipo de proyecto RTL Project, que nos permitirá crear los códigos fuente en VHDL, además de realizar simulaciones, síntesis e implementación en el FPGA.



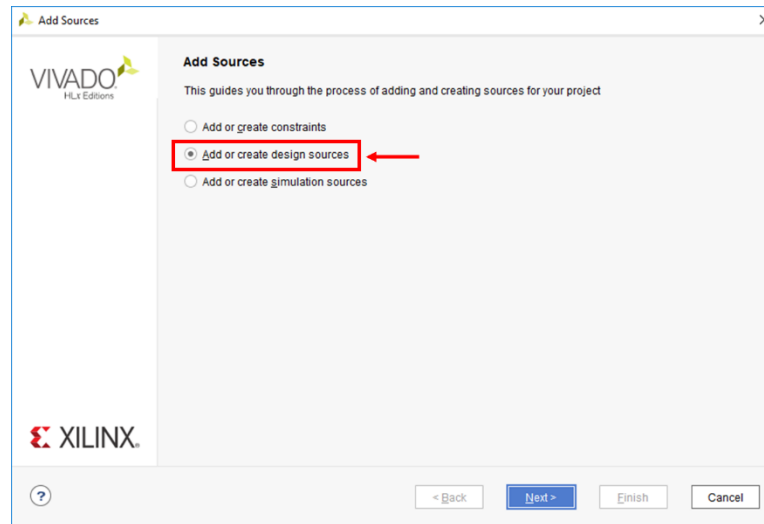
5. Por el momento, no tenemos ningún archivo de código fuente ni de restricciones para agregar al proyecto, por lo que a las ventanas Add Sources y Add Constraints, debemos solo hacer click en "Next".
6. En este laboratorio estaremos trabajando con la tarjeta "Nexys 4", que cuenta con un FPGA Artix XC7A100T-CSG324, para esto debemos configurar el proyecto para que funcione con este FPGA en particular. Después de completar la siguiente configuración hacer clic en "Next" y luego en "Finish".



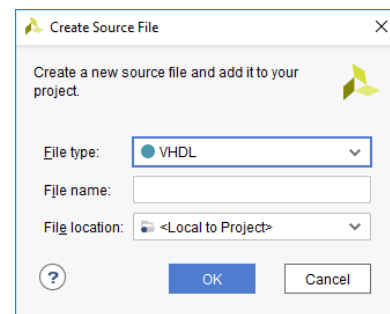
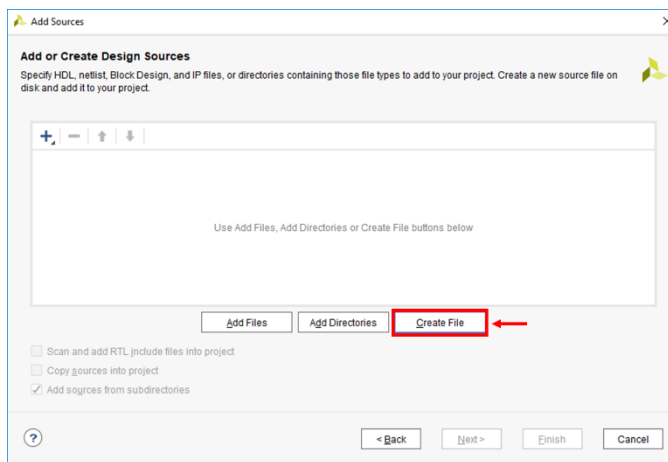
- Con esto queda terminada la configuración del proyecto. Ahora es necesario comenzar a escribir los programas que representarán la lógica combinacional y/o secuencial para resolver el problema pedido.
- En el menú "File", dar click en la opción "Add Sources..." para crear un nuevo archivo de código fuente. También puedes hacer click en el signo "+" de la ventana "Sources":



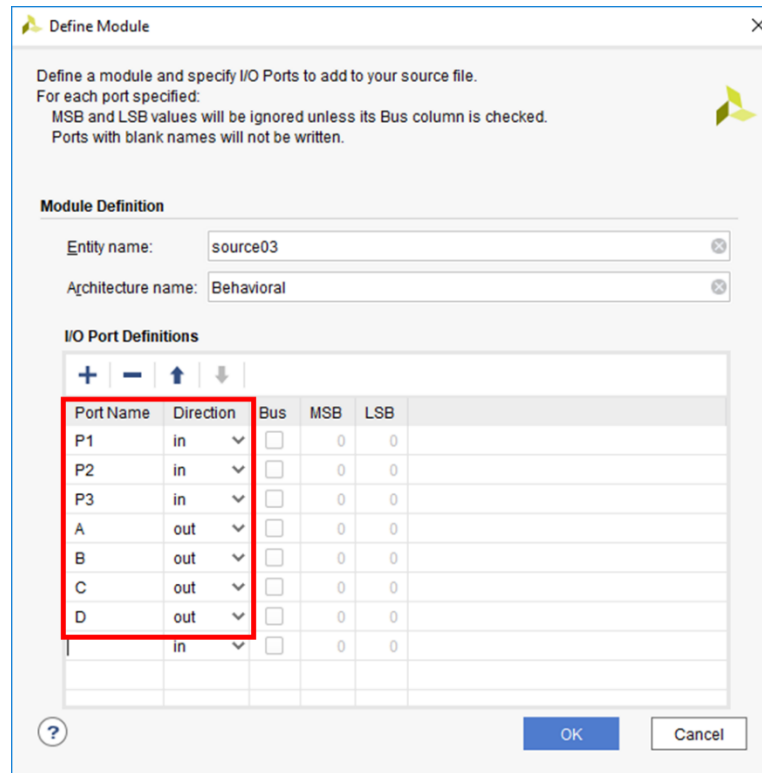
- Selecciona la opción "Add or create design sources" para indicar que deseamos crear un archivo de código fuente.



10. Dado que no contamos con un archivo con código fuente previamente escrito, en la siguiente ventana seleccionamos la opción "Create File". En la ventana emergente, seleccionamos el tipo de archivo como "VHDL", asigna un nombre a tu nuevo archivo y da click en "Ok". Para finalizar, haz click en "Finish".



11. Una nueva ventana nos pedirá definir los puertos de entrada y salida que tendrá el diseño del módulo. La definición de estos puertos en esta etapa ayuda agilizar la escritura de código fuente correspondiente a esta sección y a evitar errores. Como hemos visto, el módulo que vamos a diseñar consta de tres entradas: P1, P2 y P3; y de cuatro salidas: A, B, C y D. Define las entradas y salidas como se muestra a continuación:



Define a module and specify I/O Ports to add to your source file.
For each port specified:
MSB and LSB values will be ignored unless its Bus column is checked.
Ports with blank names will not be written.

Module Definition

Entity name:

Architecture name:

I/O Port Definitions

Port Name	Direction	Bus	MSB	LSB
P1	in	<input type="checkbox"/>	0	0
P2	in	<input type="checkbox"/>	0	0
P3	in	<input type="checkbox"/>	0	0
A	out	<input type="checkbox"/>	0	0
B	out	<input type="checkbox"/>	0	0
C	out	<input type="checkbox"/>	0	0
D	out	<input type="checkbox"/>	0	0
	in	<input type="checkbox"/>	0	0

OK Cancel

12. El archivo nuevo deberá ser agregado a la ventana "Sources" y haciendo doble click sobre el nombre del mismo, se abrirá el editor con un código fuente predefinido, incluyendo la declaración de los puertos de entrada y salida realizada en el paso anterior.
13. Es necesario agregar líneas de código que definan cómo están interconectadas las entradas y las salidas, a través de compuertas lógicas. Esto se realiza dentro de la función architecture Behavioral; específicamente, entre las líneas begin y end behavioral. En el código mostrado puedes ver la implementación de las ecuaciones booleanas definidas previamente a partir del comportamiento lógico deseado, utilizando VHDL. Completa el código faltante en tu programa basándote en estas líneas de código.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

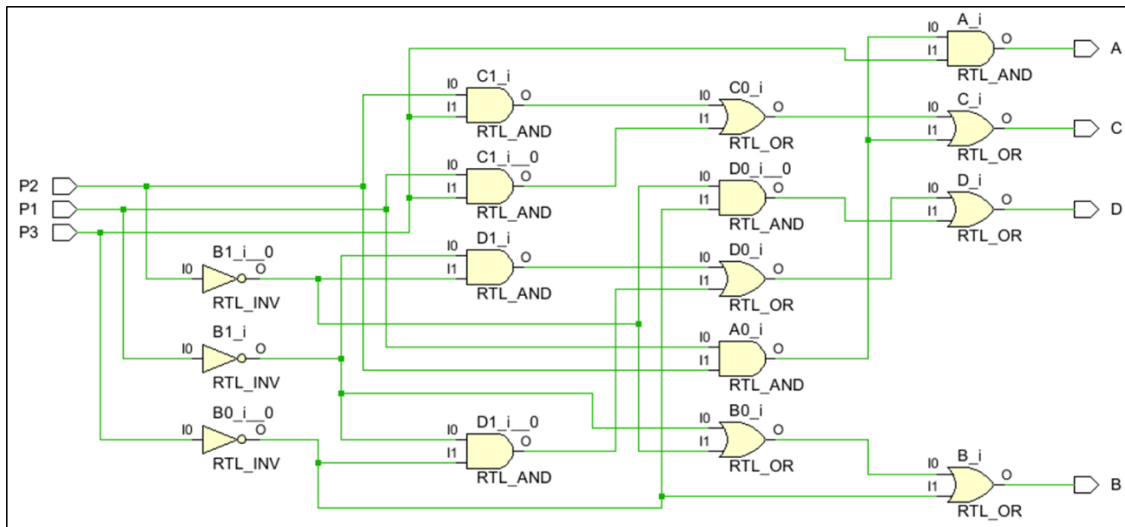
entity source02 is
    Port ( P1 : in STD_LOGIC;
          P2 : in STD_LOGIC;
          P3 : in STD_LOGIC;
          A  : out STD_LOGIC;
          B  : out STD_LOGIC;
          C  : out STD_LOGIC;
          D  : out STD_LOGIC);
end source02;

architecture Behavioral of source02 is
```



```
begin
A <= (P1 AND P2 AND P3);
B <= (NOT P1 OR NOT P2 OR NOT P3);
C <= (P2 AND P3) OR (P1 AND P3) OR (P1 AND P2);
D <= (NOT P1 AND NOT P2) OR (NOT P1 AND NOT P3) OR (NOT P2 AND
NOT P3);
end Behavioral;
```

14. Guarda tu archivo fuente una vez que hayas capturado el programa completo.
15. El ambiente de desarrollo puede generar automáticamente el diagrama esquemático a partir de las ecuaciones booleanas capturadas en el programa. Para esto, dentro de la ventana "Flow Navigator", haz click en la sección "RTL Analysis" y selecciona "Ok" en la ventana emergente que aparecerá. En el diagrama esquemático generado, podrás ver las tres entradas, P1, P2 y P3; las cuatro salidas, A, B, C y D, así como las compuertas lógicas descritas con VHDL en tu código fuente. En el diagrama siguiente puedes ver un ejemplo de cómo podría lucir tu esquemático:



16. Para proceder a la simulación es común y deseable crear un testbench, o banco de pruebas, que consiste en un archivo con código fuente adicional que le dice al ambiente de desarrollo cuáles son las entradas en nuestro diseño que queremos estimular, y nos permite monitorear las salidas de este. Para crear un testbench, crea un archivo nuevo de código fuente en VHDL (extensión .vhd) de la misma forma que se realizó en los pasos 8 a 11 de este procedimiento. Es recomendable anteponer las letras "tb_" y utilizar el nombre del archivo de código fuente para el que se está preparando el testbench, para mantener ordenado el árbol de archivos.
17. Para el caso de un testbench, no es necesario definir ningún puerto de entrada o de salida en la ventana de definición del módulo.



18. El código fuente del testbench se muestra a continuación. Copia este código a tu propio testbench, poniendo especial atención a los comentarios realizados en cada sección para que tengas claro el propósito de cada una.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity tb_source02 is
-- Port ( );
end tb_source02;

-- Conexión del testbench al diseño deseado. En este caso
"source02"
architecture Behavioral of tb_source02 is
component source02 is
    Port ( P1 : in STD_LOGIC;
           P2 : in STD_LOGIC;
           P3 : in STD_LOGIC;
           A : out STD_LOGIC;
           B : out STD_LOGIC;
           C : out STD_LOGIC;
           D : out STD_LOGIC);
end component;

-- Creación de señales de estimulación y monitoreo
signal P1_s : STD_LOGIC;
signal P2_s : STD_LOGIC;
signal P3_s : STD_LOGIC;
signal A_s : STD_LOGIC;
signal B_s : STD_LOGIC;
signal C_s : STD_LOGIC;
signal D_s : STD_LOGIC;

begin

-- Mapeo de entradas y salidas a señales del testbench
DUT: source02 port map(
    P1 => P1_s,
    P2 => P2_s,
    P3 => P3_s,
    A => A_s,
    B => B_s,
    C => C_s,
    D => D_s);

-- Estimulación de entradas mediante señales de testbench
process
begin
    P1_s <= '0';
    P2_s <= '0';
```




```
P3_s <= '0';

wait for 10 ns;

P1_s <= '0';
P2_s <= '0';
P3_s <= '1';

wait for 10 ns;

P1_s <= '0';
P2_s <= '1';
P3_s <= '0';

wait for 10 ns;

P1_s <= '0';
P2_s <= '1';
P3_s <= '1';

wait for 10 ns;

P1_s <= '1';
P2_s <= '0';
P3_s <= '0';

wait for 10 ns;

P1_s <= '1';
P2_s <= '0';
P3_s <= '1';

wait for 10 ns;

P1_s <= '1';
P2_s <= '1';
P3_s <= '0';

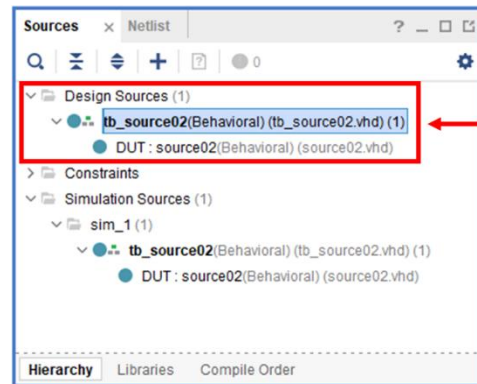
wait for 10 ns;

P1_s <= '1';
P2_s <= '1';
P3_s <= '1';

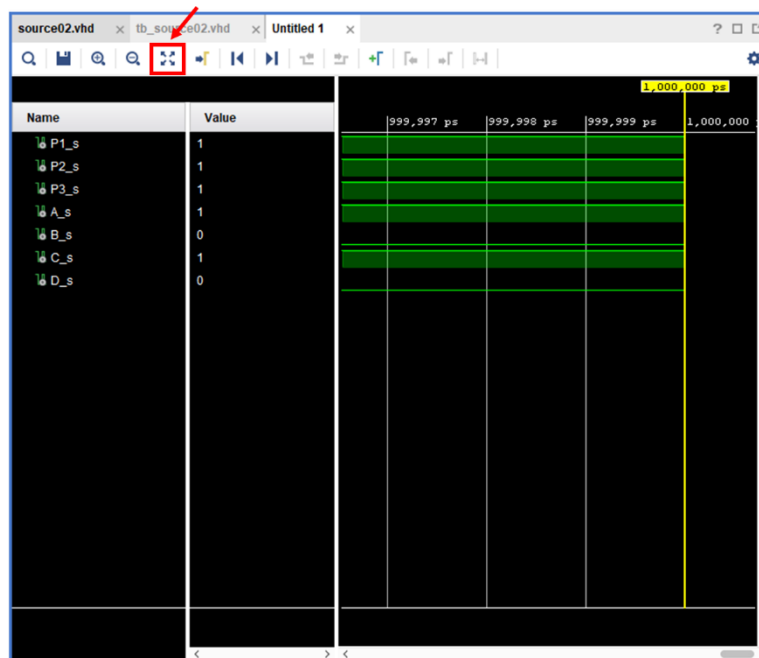
wait;
end process;
end Behavioral;
```

19. En el test bench definimos el Design Under Test (DUT) que es diseño que estamos probando ("source02" en este caso). El ambiente de desarrollo reconoce esto y muestra,

en la ventana "Sources", el DUT como un diseño dependiente jerárquicamente del test bench creado:



20. Para ejecutar la simulación, basta con hacer click en "Run Simulation" bajo la pestaña "SIMULATION" en la ventana "Flow Navigator", y seleccionar "Run Behavioral Simulation". La ventana de simulación se verá similar a la imagen mostrada abajo. Haz click en el botón "Zoom Fit" para visualizar la línea de tiempos completa de la simulación.



21. El simulador ejecuta por defecto una simulación de 1 microsegundo; aunque en este caso, la prueba de estimulación de entradas tomó solamente 70 nanosegundos. Para ajustar la línea de tiempo y visualizar mejor los resultados, en la barra de simulación, (1) introduce un tiempo de 100 ns, luego haz click en el botón "Restart" (2), y, por último, "Run for 100 ns" (3). Finalmente, presiona nuevamente el botón "Zoom Fit".



22. Comprueba que las salidas correspondan a la tabla de verdad propuesta al inicio del ejercicio. Si no es así, modifica tu código fuente para hacer las correcciones necesarias.
23. Guarda tu proyecto.

Entregables

Los entregables deben ser enviados al final de la parte 2 de la práctica. Se entrega **un solo reporte** en incluya tu trabajo tanto de la parte 1 como de la parte 2 de la práctica. Los entregables son:

- Código/ fuente en VHDL (GitHub)
- Demostración de implementación en FPGA (Video)
- Reporte técnico (PDF en Canvas)
 - Esquemático RTL (paso 15 de la parte 1)
 - Evidencias de simulación con capturas de pantalla para las diferentes combinaciones de entradas.
 - Enlaces a repositorio en GitHub y video demostrativo