Introducción al ISE WebPACK 14.7

Fernando Guiraud

Laboratorio I de Microprocesadores
Grupo 1EE141
Profesor: Elias Mendoza

1. Introducción

El Xilinx ISE WebPACK es un conjunto completo de diseño lógico programable FPGA/CPLD que proporciona:

- ✓ Especificación de lógica programable mediante captura esquemática o Verilog/VHDL
- ✓ Síntesis y "Place Route" de la lógica especificada para varios FPGA y CPLD de Xilinx
- ✓ Simulación funcional (comportamental) y de sincronización ("post-Place Route")
- ✓ Descarga de datos de configuración en el dispositivo de destino a través del cable de comunicaciones

El desarrollo de ISE WebPACK se ha detenido a favor de Vivado Suite, pero sigue siendo útil desarrollarlo para dispositivos más antiguos que no son compatibles con la nueva suite.

2. Procedimiento

En este informe de laboratorio usamos el ISE WebPACK 14.7. El primer paso consistió en crear un nuevo proyecto y definir los parámetros de la tarjeta FPGA a ser utilizada, en este caso una Spartan 3A de Xilin, llamada ElbertV2 y elaborada por Numato.

Definimos las entradas y salidas a ser utilizadas en el algoritmo, las cuales corresponden dos entradas y cinco salidas lógicas. Mas adelante estas entradas y salidas serán mapeadas con las rutas físicas de la tarjeta usando los datos del archivo ucf de la tarjeta.

Para cada entrada definimos un switch de dos estados y para cada salida representamos con un led.

Figura 1. Configuración de la tarjeta Spartan 3A.

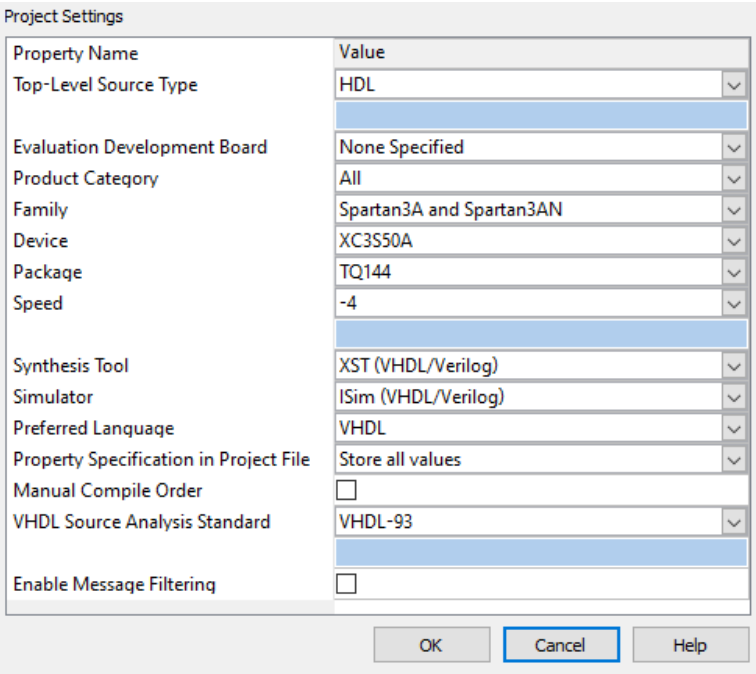


Figura 2. Ucf correspondiente de cada pin de la tarjeta.

```
1
2     NET "a"        LOC = P70   | PULLUP  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
3     NET "b"        LOC = P69   | PULLUP  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
4
5     NET "f_not"            LOC = P46   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
6     NET "f_and2"            LOC = P47   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
7     NET "f_or2"            LOC = P48   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
8     NET "f_nand2"            LOC = P49   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
9     NET "f_nor2"            LOC = P50   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
```

Escribimos un código encargado de hacer operaciones lógicas con las dos entradas proporcionadas:

Como podemos ver en la **figura 3**, se incluyeron las librerías IEEE y IEEE.STD LOGIC 1164. Seguidamente podemos ver las variables de entrada y salida. Por ultimo vemos las operaciones entre las entradas, la primera consiste en la inversión de la entrada a, la segunda una compuerta and entre las dos entradas, la tercera una compuerta or, la cuarta una compuerta nand y por ultimo una compuerta nor.

Figura 3. Código VHDL.

```
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22
23  entity misCompuertas is
24      Port ( a : in  STD_LOGIC;
25             b : in  STD_LOGIC;
26             f_not : out  STD_LOGIC;
27             f_and2 : out  STD_LOGIC;
28             f_or2 : out  STD_LOGIC;
29             f_nand2 : out  STD_LOGIC;
30             f_nor2 : out  STD_LOGIC);
31  end misCompuertas;
32
33  architecture Behavioral of misCompuertas is
34
35  begin
36
37   f_not <= not a;
38   f_and2 <= (a and b);
39   f_or2 <= (a or b);
40   f_nand2 <= (a nand b);
41   f_nor2 <= (a nor b);
42
43  end Behavioral;
```

3. Resultados

Para obtener los resultados de esta experiencia de laboratorio, nos limitamos a simular las operaciones lógicas dentro del entorno de Xilin. Para esto generamos el siguiente código de simulación:

Como se puede ver en la **figura 4**, el código se encarga de darle todas las posibles combinaciones de valores a las dos entradas para probar las compuertas lógicas definidas en la sección anterior. Cada prueba tiene una duración de 10 nanosegundos.

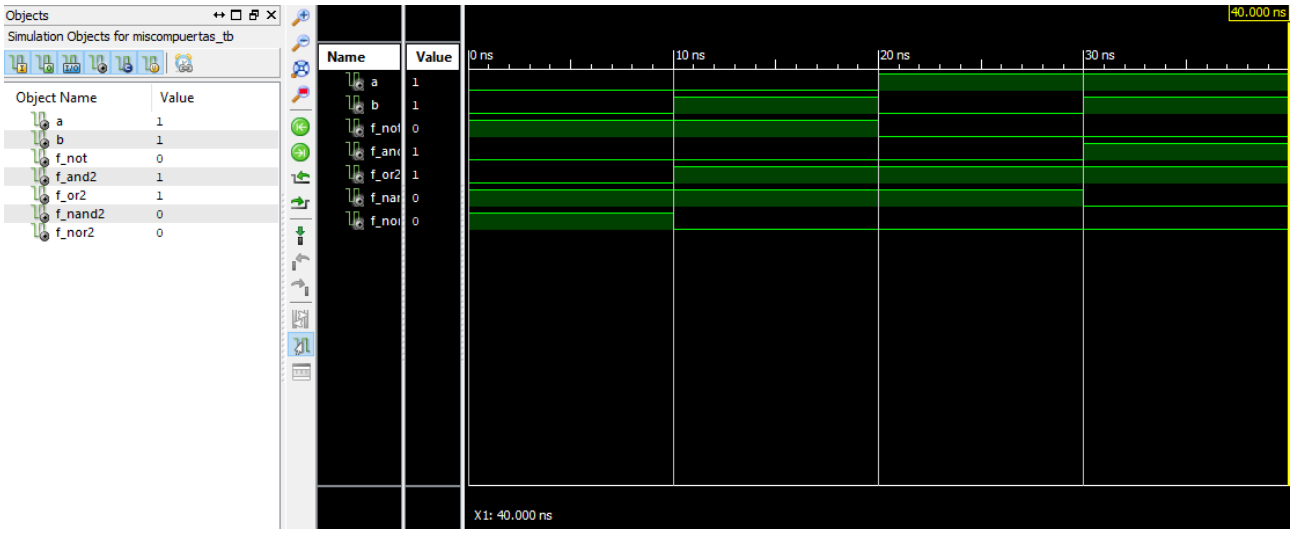
Esto nos da como resultado la simulación de la **figura 5**.

En esta figura, podemos ver como los valores de a y b van cambiando en el tiempo estipulado y como estas entradas van variando los resultados de las operaciones lógicas de cada compuerta.

Figura 4. Código de simulación.

```
93      -- Stimulus process
94      stim_proc: process
95      begin
96          -- hold reset state for 100 ns.
97
98          -- wait for <clock>_period*10;
99
100         -- insert stimulus here
101   a<='0';
102   b<='0';
103
104   wait for 10 ns;
105   a<='0';
106   b<='1';
107
108   wait for 10 ns;
109   a<='1';
110   b<='0';
111
112   wait for 10 ns;
113   a<='1';
114   b<='1';
115         wait;
116     end process;
117
118   END;
119
```

Figura 5. Simulación.



4. Conclusiones

Dentro del entorno de programación de Xilin podemos definir entradas y salidas las cuales podemos manejar con un sinnúmero de operaciones lógicas con un fin especifico, el cual puede ser verificado mediante simulaciones sin tener que utilizar una tarjeta físicamente. Esto nos puede ayudar a asegurarnos que el algoritmo creado cumple su función antes de ser llevado a la tarjeta y evitar posibles errores que pueden ser dañinos para los sistemas diseñados dependiendo de las funciones que cumplen los mismos.