

CIRCUITOS ELECTRÓNICOS DIGITALES (CED-ISW)

Práctica 4: Descripción de funciones combinacionales con Verilog e implementación en FPGA)

Objetivos de la práctica

- Introducción al entorno de diseño lógico Xilinx ISE webpack 12.4, con sus funciones básicas
- especificación de circuitos combinacionales básicos en Verilog
- Simulación lógica de circuitos combinacionales
- Implementación en un dispositivo programable FPGA

Material:

- Ordenador con Xilinx ISE Design Suite 12.4 instalado y placa de entrenamiento de FPGA Basys2
- Tutorial de Xilinx ISE

1. ESTUDIO TEÓRICO (Debe presentarse antes de empezar la práctica)

- a) Analice el circuito de la figura y obtenga su expresión algebraica y su mapa de Karnaugh:

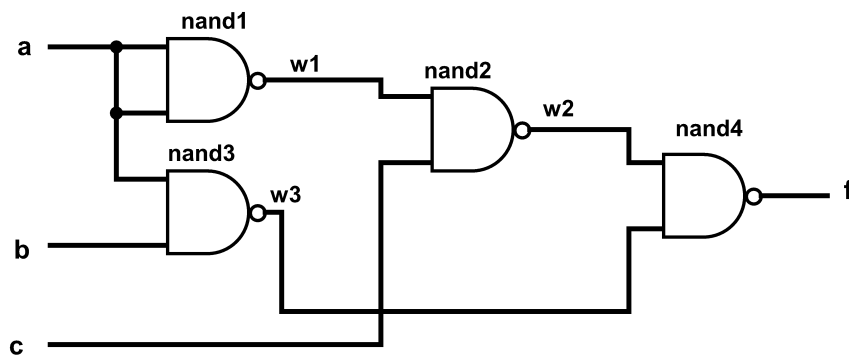
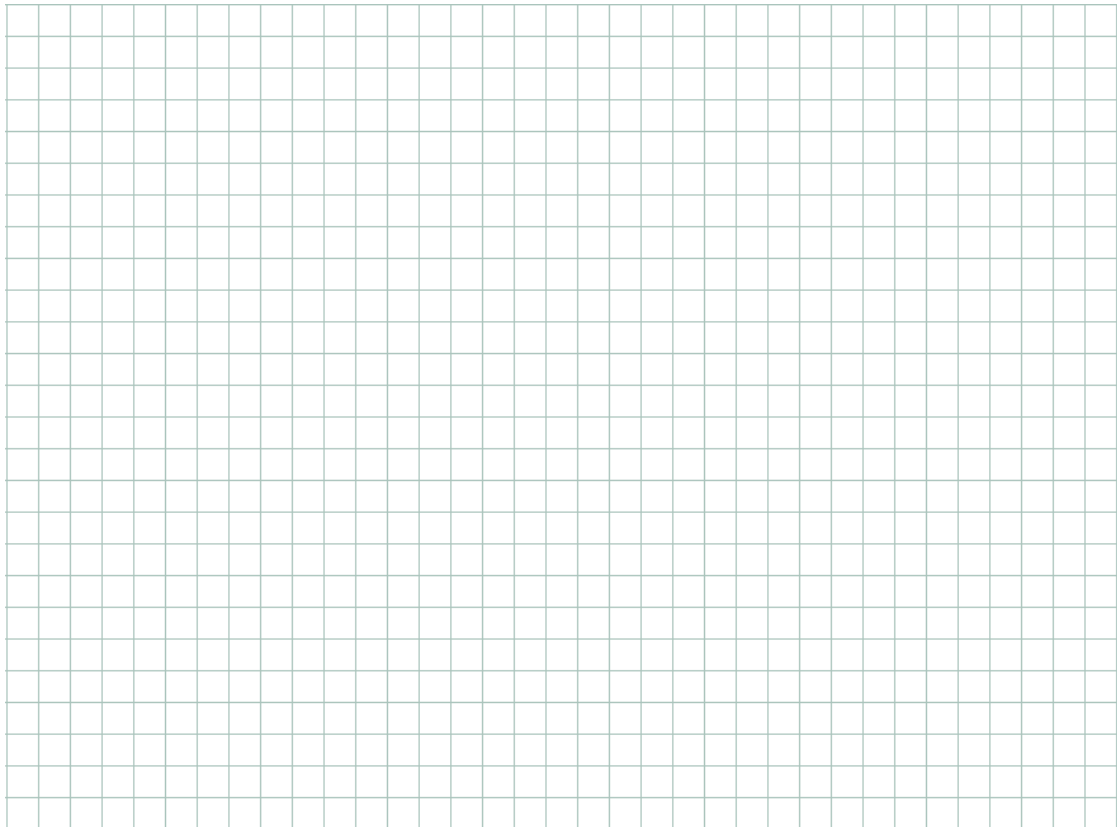
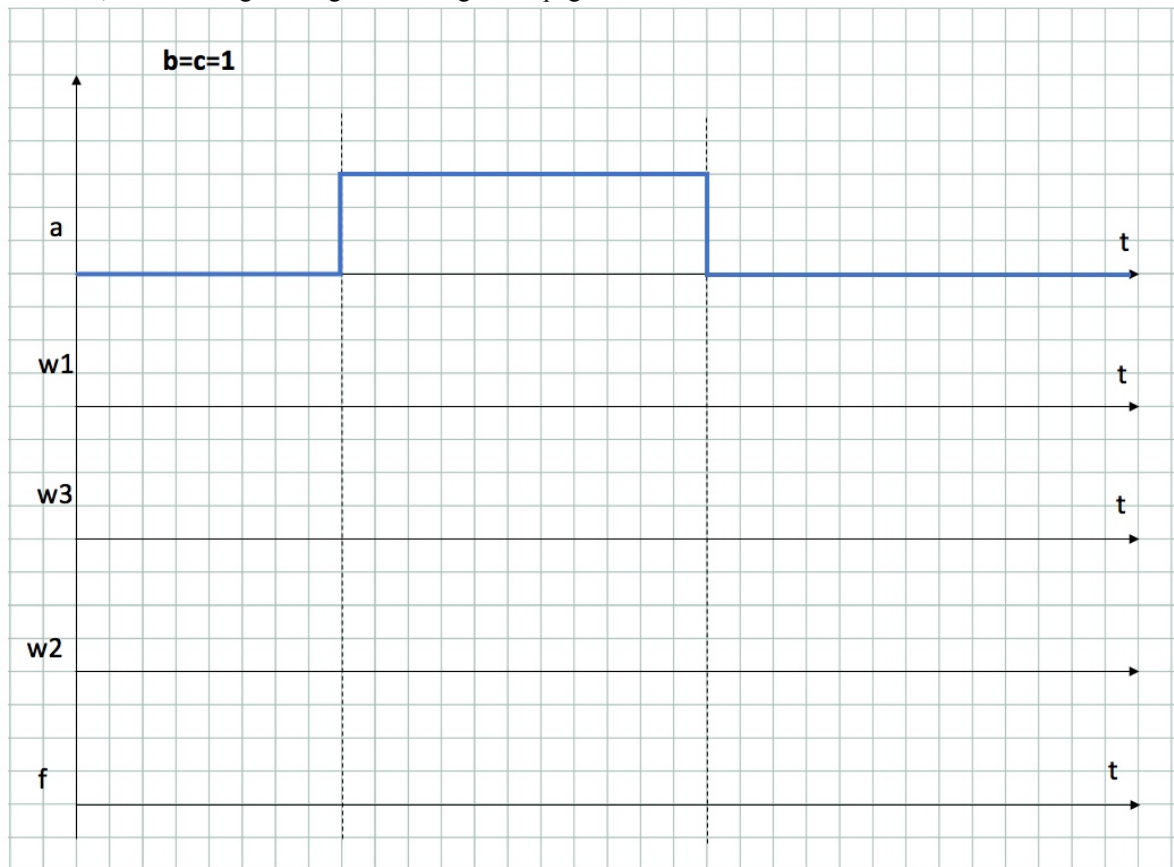


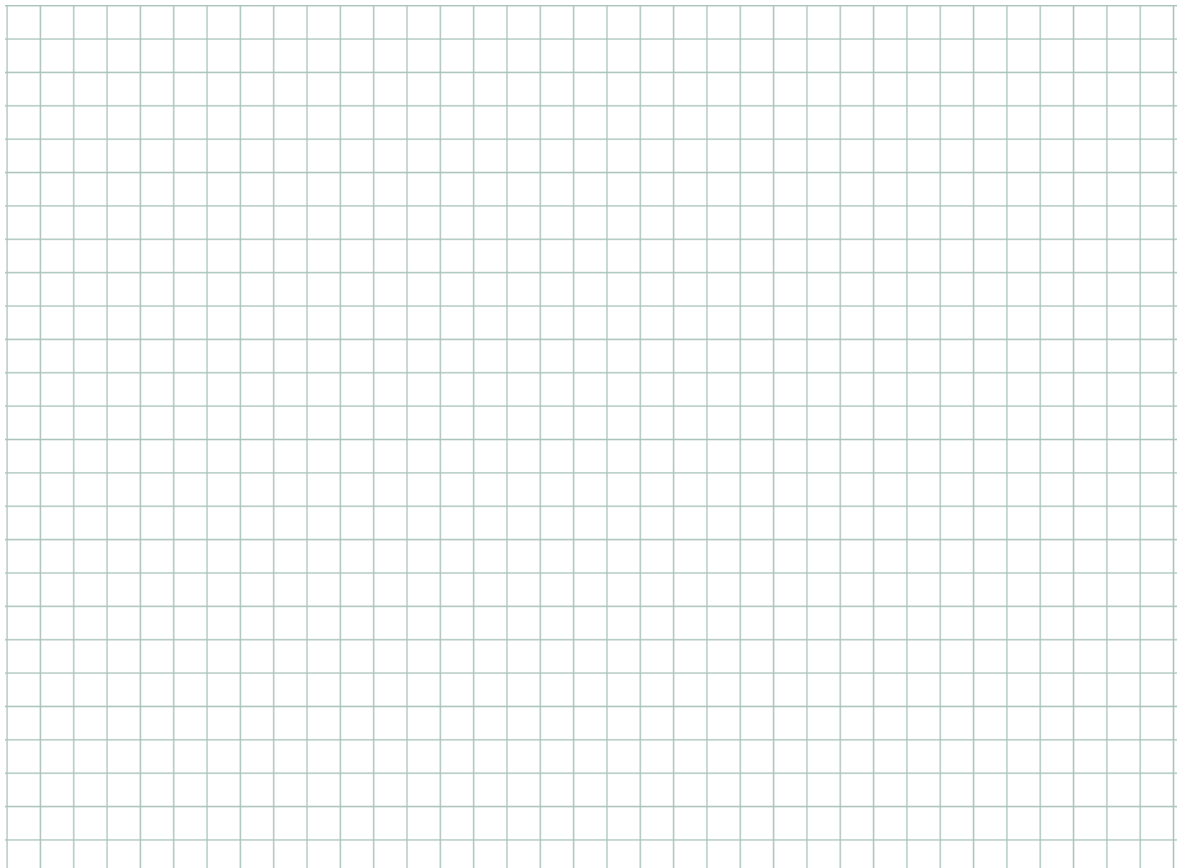
Figura 1



- b) Considerando que todas las puertas introducen un retraso Δ , obtenga las formas de onda de $w1$, $w2$, $w3$ y f para la siguiente situación: $b=c=1$; a cambia según la figura de la siguiente página:



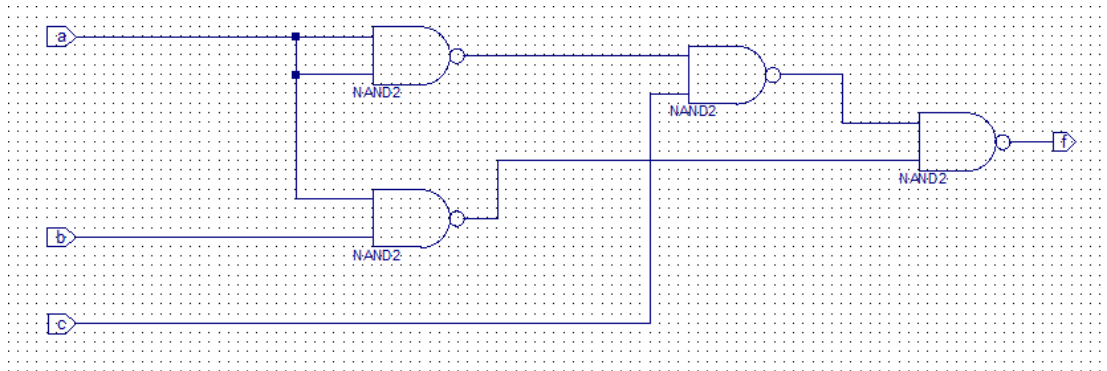
- c) Diseñe un circuito “votador” o “mayoritario” de tres entradas (a, b, c). La salida f tomará el valor lógico que más se repita en sus entradas.



2. TRABAJO EXPERIMENTAL EN EL LABORATORIO

2.1 SCHEMATICS

- a) Cree un proyecto en blanco en ISE Design Suite y, con ayuda de Schematics, añada un módulo correspondiente al circuito de la figura 1. Debe obtener algo parecido a:

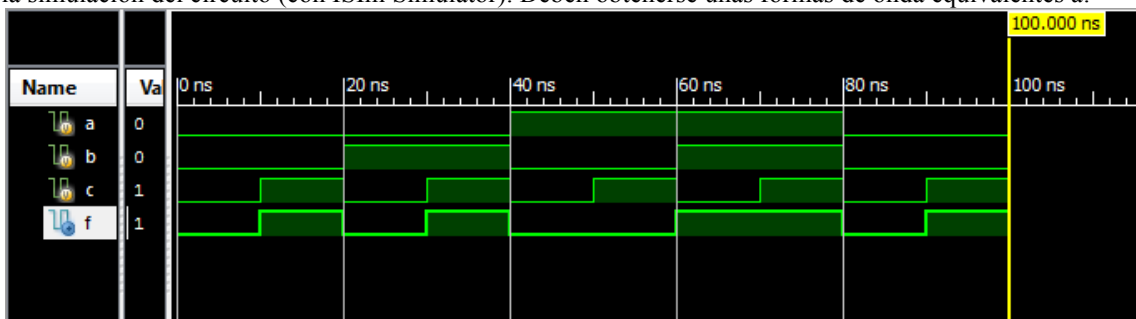


- b) Active “modo simulación” e incorpore ahora un fichero de simulación de este circuito y compruebe que funciona según su Kmapa. Para ello, se propone incluya las siguientes órdenes de simulación:

```
always #10 {a,b,c} = {a,b,c} + 1;

//Initialize Inputs
initial begin
    a=0;
    b=0;
    c=0;
    #100;
    $finish;
end
```

- c) Ejecute la simulación del circuito (con ISIm Simulator). Deben obtenerse unas formas de onda equivalentes a:



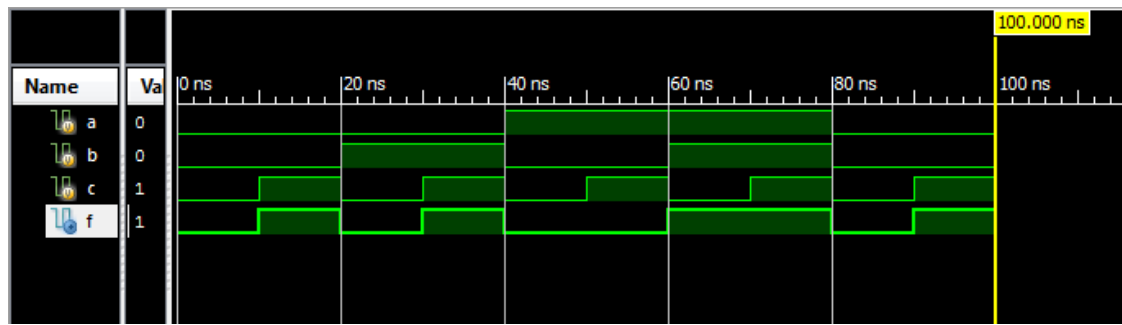
Como puede comprobar, la simulación generó todos los posibles valores binarios de las variables a,b,c y se ha calculado los valores lógicos de f. Compruebe si el funcionamiento del circuito responde al k-mapa obtenido en el análisis del estudio teórico.

2.2. DESCRIPCIÓN VERILOG FUNCIONAL

- a) Cree un proyecto nuevo con un módulo que corresponda a la descripción funcional de la función f(a,b,c) obtenida en el estudio teórico.

```
module f_funcional(
    input a,b,c,
    output f
);
// especifique aquí la función con assign
endmodule
```

- b) Realice la simulación del módulo y compruebe que funciona de acuerdo a su Kmapa. Para ello debe pasar a “modo simulación” y crear un nuevo fichero con las órdenes de simulación correspondientes. Deben generarse unas gráficas similares a las del apartado 2.1.c



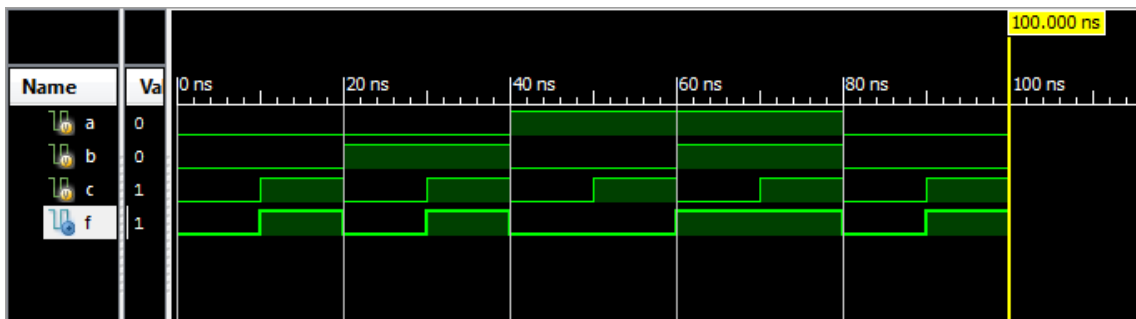
2.3. DESCRIPCIÓN VERILOG ESTRUCTURAL

- a) Cree un nuevo proyecto con un nuevo módulo Verilog, correspondiente a la descripción estructural (a nivel de puertas) del circuito de la figura 1 del estudio teórico.

```
module circuitonand_gates (
    input a,b,c,
    output f
);
    wire w1,w2,w3; //tenemos que declarar los cables de conexión
    nand nand1(w1,a,a);
    //complete la descripción estructural

endmodule
```

- b) Realice la simulación del circuito y compruebe que funciona de acuerdo a su kmapa. Para ello debe pasar a “modo simulación” y crear un nuevo fichero con las órdenes de simulación correspondientes. Deben generarse unas gráficas similares a las del apartado 2.1.c



- c) Con objeto de visualizar las formas de onda de w1, w2 y w3, debemos modificar la especificación Verilog del módulo anterior para convertir **w1, w2 y w3 en salidas del circuito** (al mismo nivel que f); incluya también retardos de 1ns en cada una de las puertas:

```
module circuitonand_gates (
    input a,b,c,
    output          //aquí vienen las salidas
);
    nand #1 nand1(w1,a,a);
    //complete la descripción estructural

endmodule
```

- d) Cree un nuevo fichero de pruebas (test bench) para intentar reproducir la situación planteada en el apartado b del estudio teórico (esto es, b=c=1, y a cambia de 0 a 1 y vuelve a 0):

```
//Initialize Inputs
initial begin
    a=0;
    b=1;
    c=1;
    //. . .

    $finish;
end
```

2.4. IMPLEMENTACIÓN EN FPGA DEL CIRCUITO VOTADOR DE 3 ENTRADAS

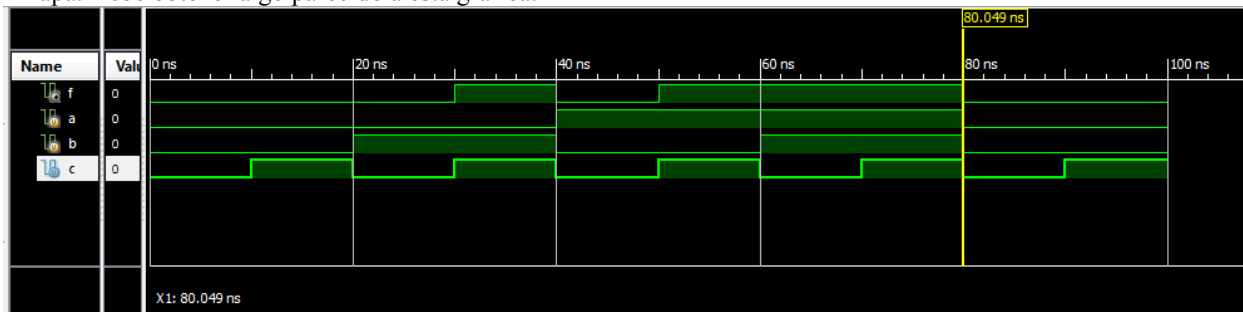
Se pretende realizar ahora el ciclo de vida completo de un proyecto: especificación, diseño, simulación e implementación física. Para ello, vamos a partir del circuito votador diseñado en el apartado c del estudio teórico. Utilizaremos la placa Basys 2 del fabricante Digilent. Los valores lógicos de las variables a,b y c (votaciones) se realizarán con 3 de los 8 switches de la placa; el resultado de la votación se visualizará en uno de los 8 LEDs disponibles.

- a) Cree un proyecto nuevo en ISE Design Suite y especifique en Verilog el circuito votador de tres entradas.

```
module votador (
    input a,b,c,
    output f
);
//especifique la función con assign

endmodule
```

- b) Incluya un fichero testbench para este módulo y realice una simulación lógica del circuito para comprobar que funciona según su kmapa. Debe obtener algo parecido a esta gráfica:



- c) Pasamos ahora a implementar el circuito en la placa BASYS2, que mostramos en la siguiente figura:

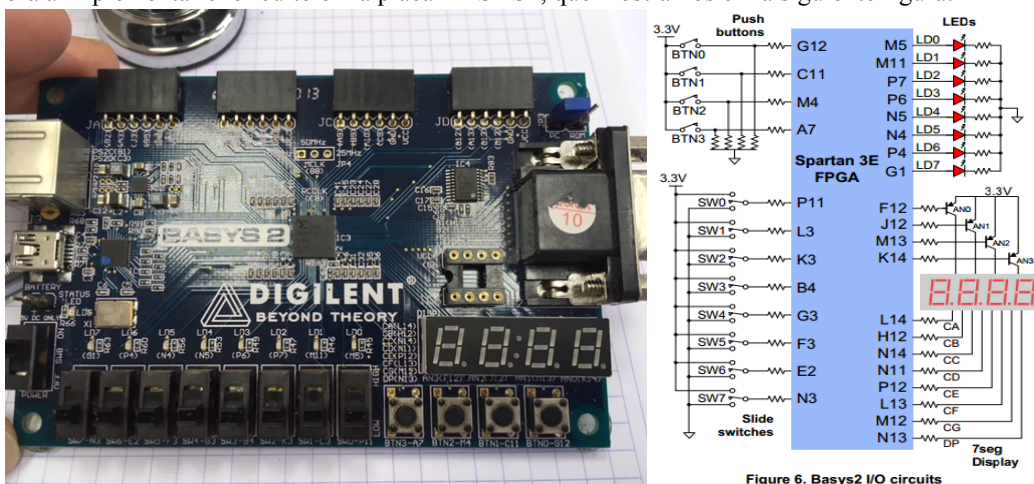


Figure 6. Basys2 I/O circuits

El fabricante de la placa, suministra un fichero de asociaciones “basys2.ucf” que tendremos que modificar para asignar las entradas y salida de nuestro módulo verilog, a pines (patitas o terminales) de la FPGA Spartan 3E que, a su vez, están conectados a elementos físicos de la placa (LEDs, switches, pulsadores, displays 7 segmentos, etc).

Pase a modo “**Implementation**”, añada al proyecto el fichero de asociaciones “**basys2.ucf**” y edítelo. Todas las líneas deben empezar por // (comentarios), salvo las líneas correspondientes a los dispositivos físicos que vamos a utilizar. Podemos asociar, por ejemplo, las entradas (variables a,b,c) a los switches SW7,SW6, SW5 y la salida (función f) al LED LD7.

```
# This file is a general .ucf for Basys2 rev C board
# To use it in a project:
# - remove or comment the lines corresponding to unused pins
// . . .

# Pin assignment for LEDs
NET "f" LOC = "G1" ; # Bank = 3, Signal name = LD7
//NET "g" LOC = "P4" ; # Bank = 2, Signal name = LD6
//NET "ld<5>" LOC = "N4" ; # Bank = 2, Signal name = LD5
//NET "ld<4>" LOC = "N5" ; # Bank = 2, Signal name = LD4
// . . .

# Pin assignment for Sws
NET "a" LOC = "N3"; # Bank = 2, Signal name = SW7
NET "b" LOC = "E2"; # Bank = 3, Signal name = SW6
NET "c" LOC = "F3"; # Bank = 3, Signal name = SW5
//NET "d" LOC = "G3"; # Bank = 3, Signal name = SW4
//NET "datb<3>" LOC = "B4"; # Bank = 3, Signal name = SW3
// . . .
```

El formato es:

NET “nombre_entrada_o_salida_modulo” **LOC** = “Nombre pin FPGA”;

Genere el fichero de programación del proyecto depurando errores si fuera necesario; para ello, marque el fichero Verilog raíz (de mayor jerarquía o “top module”), y en la ventana de herramientas seleccione “**Generate Programm File**”. Con ello se generará un fichero de programación con extensión .bit que utilizaremos para programar la FPGA con ayuda de la aplicación ADEPT, suministrada por Digilent, fabricante de la placa Spartan 3E .

Consulte el documento “**Tutorial de Xilinx ISE**” para seguir con detalle todo el proceso.

2.5. IMPLEMENTACIÓN DE OTRAS FUNCIONES COMBINACIONALES

El profesor puede proponer la realización de otros diseños de funciones combinacionales (por ejemplo, algún problema del boletín 2), para comprobar que el estudiante ha asimilado los contenidos de la práctica.