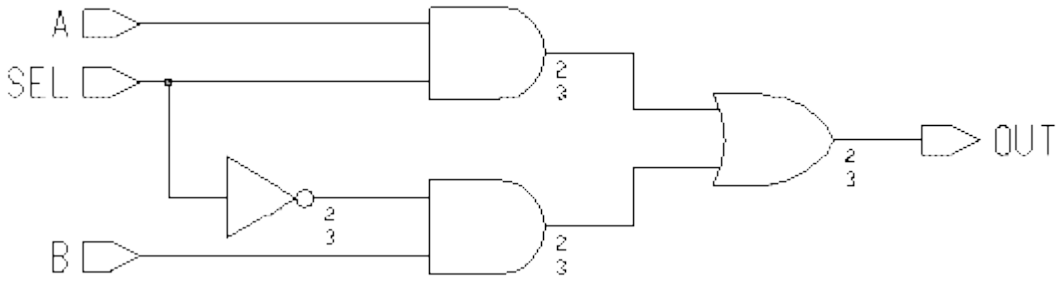
Diseño de un circuito con multiplexores y su implementación en Verilog usando la FPGA Spartan 3A

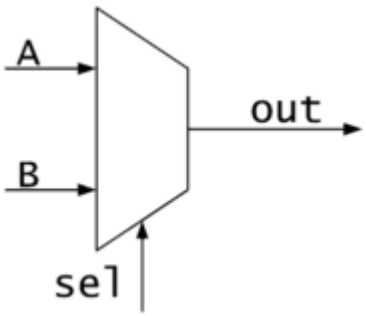
En resumen, en este documento se indica paso a paso la descripción de hardware en la FPGA de un multiplexor 2 a 1, un multiplexor 8 a 1 y posteriormente el uso de estos para diseñar un circuito y su descripción a partir de la siguiente tabla de verdad:

SW [2:0]	Y
000	8'b10101010
001	8'b11001100
010	8'b10011001
011	8´b10100001
. . .	X X x

Para empezar cabe recordar lo que es un multiplexor 2 a 1 de manera lógica:



Y cómo se representa de forma simplificada:



sel	out
0	A
1	B

Viendo así que un multiplexor de este tipo se comporta como un interruptor, siendo la salida igual a A si el selector está en 0, y la salida igual a B si el selector está en 1.

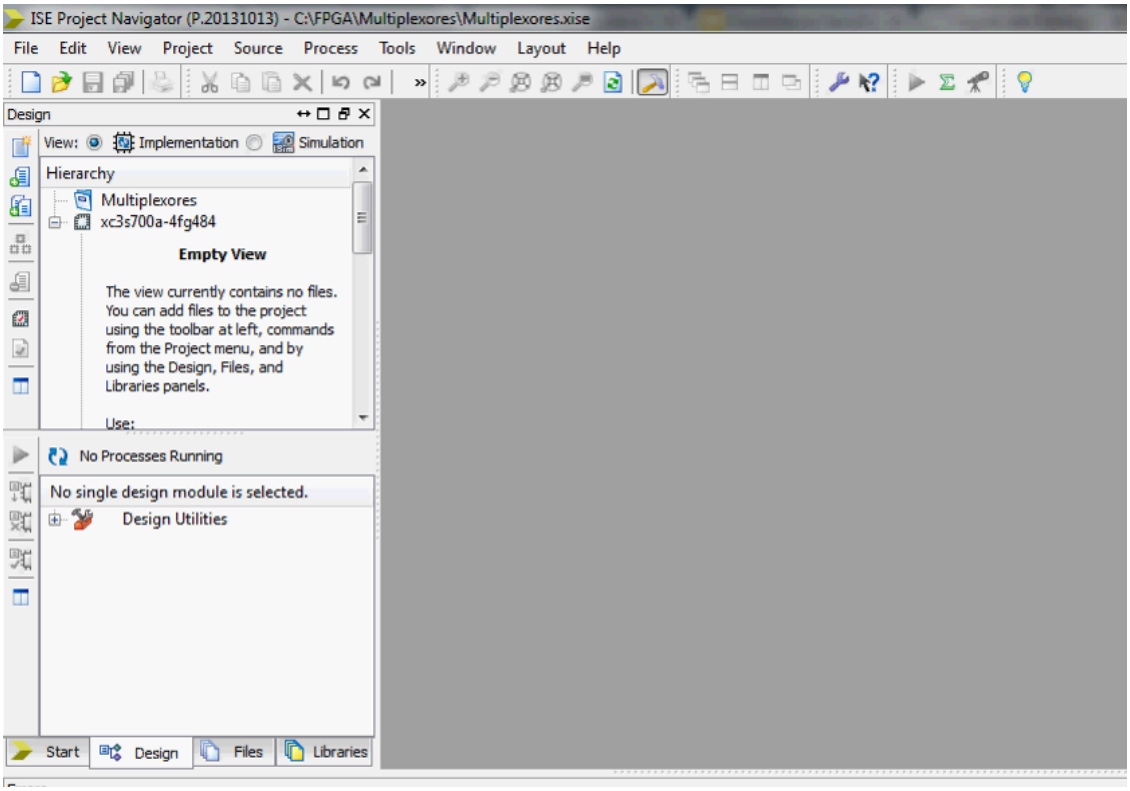
Para implementar este circuito combinacional es necesario conocer su expresión booleana (La cual se puede obtener por mapas de Karnaugh y agrupación por minterms),

$$Y = \bar{S}D_o + SD_1$$

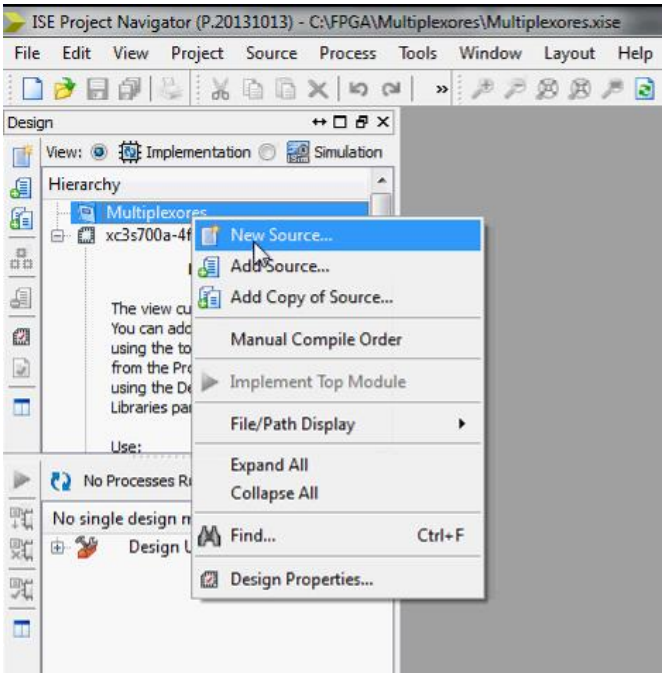
Donde se ha llamado S al selector, Y a la salida y D_o y D_1 a cada una de las entradas.

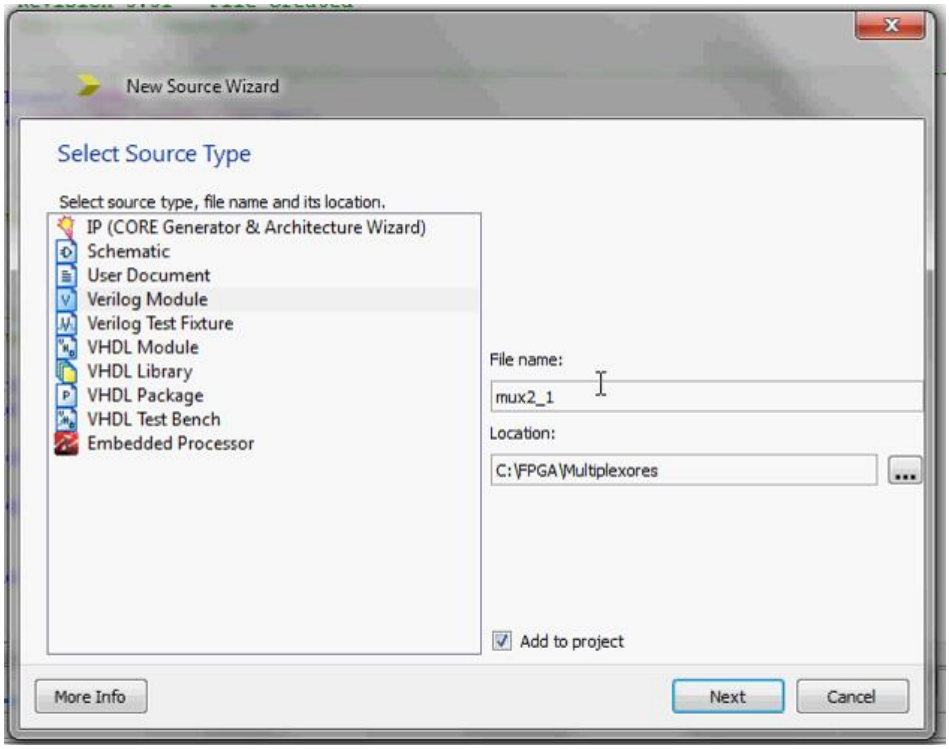
Teniendo esto claro se puede empezar describir este circuito en Verilog.

Los pasos a seguir serán:
En primer lugar, crear un nuevo proyecto o por conveniencia usar la plantilla trabajada en la práctica anterior, para este caso se hizo sobre un proyecto en blanco (Si es la primera vez que se usa el software puede remitirse a https://drive.google.com/file/d/0B_sA6lW3ej35X0c3bGo2SGQ4Q0k/view donde se explica cómo introducir algunas especificaciones para esta tarjeta en particular)

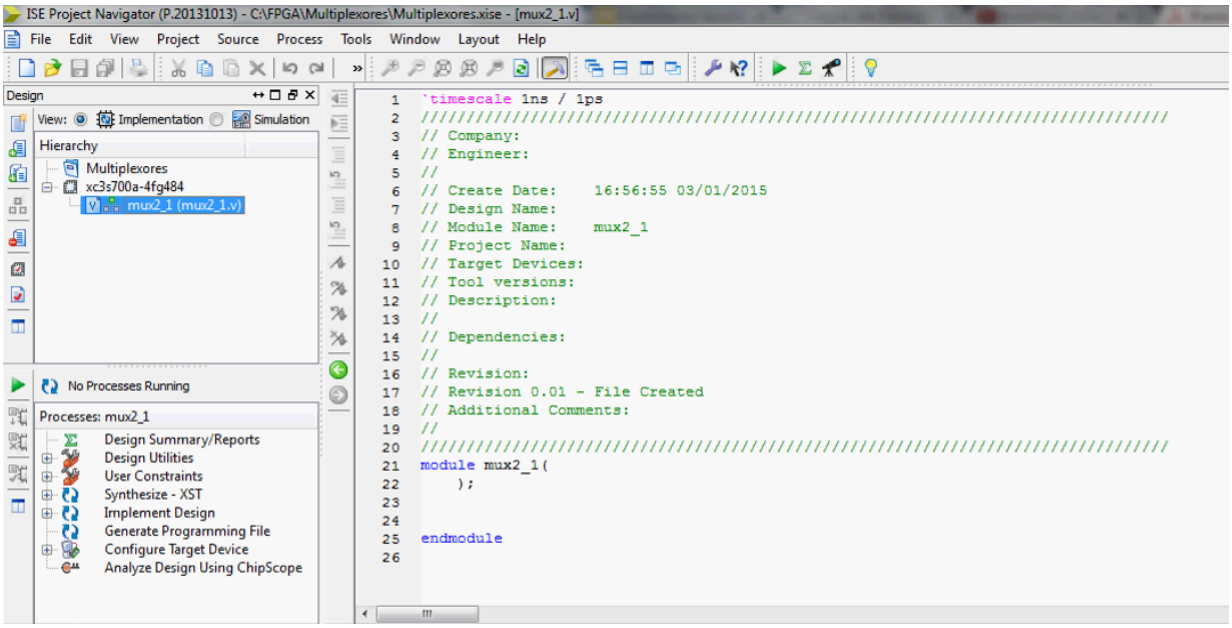


Agregar el primer módulo que será el que contenga la descripción del multiplexor de 2 entradas a 1 salida





Se tendrá algo como esto:



Donde se hará la descripción de la siguiente manera:

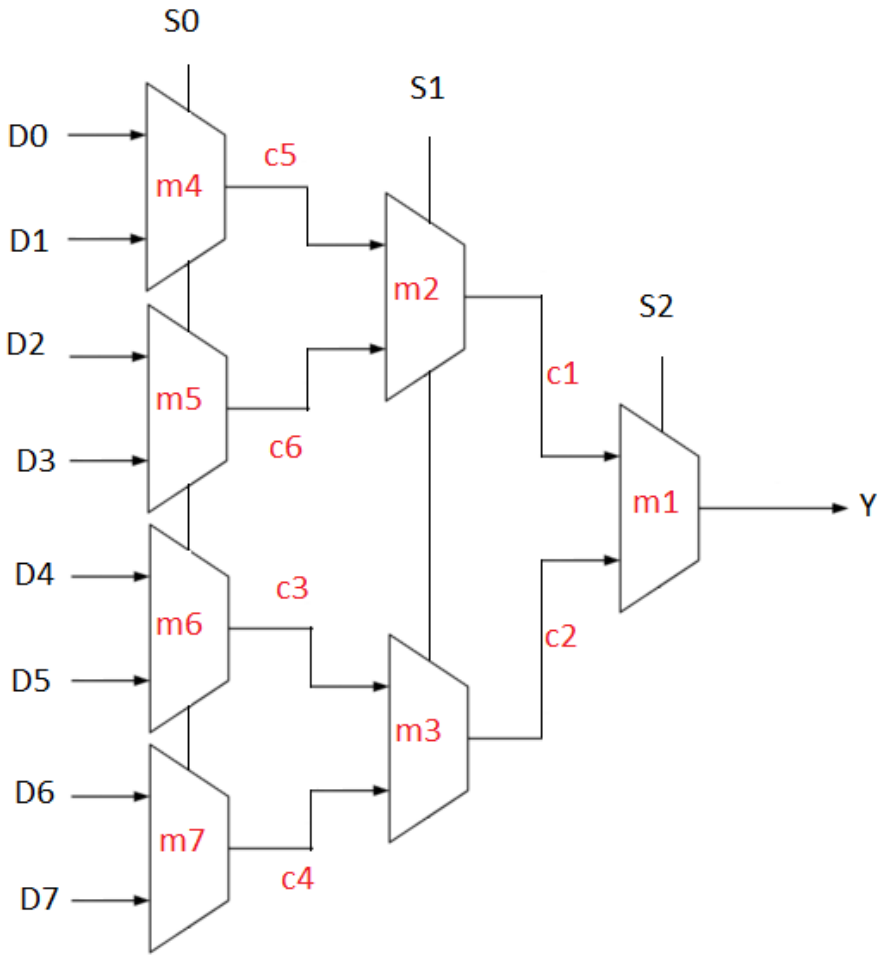
- *S, D0 y D1* son las entradas
- *Y* es la salida
- *Y* tiene por consiguiente asignada la expresión booleana del circuito

```
module mux2_1(
input S,
input D0,
input D1,
output Y
   );

assign Y=(~S|D0)&(S|D1);

endmodule
```

En este punto se puede comprobar la sintaxis del programa, si hay algún error es necesario corregirlo antes de continuar.

Si al verificar el programa ha indicado que la sintaxis es correcta, ya ha hecho su primera descripción de un multiplexor ☺

Para describir el multiplexor de 8 a 1, se instanciará el multiplexor descrito anteriormente de la siguiente manera:



Con 7 multiplexores de 2 a 1 para un total de tres selectores que serán las tres entradas variables como se indican en la tabla mostrada previamente y una salida de la que hablaremos más adelante.

Ahora bien, su descripción en Verilog será:

- Un bus de ocho entradas llamado D
- Un bus de tres selectores llamados S0, S1, y S2 respectivamente
- Una salida llamada LED
- Siete cables para las respectivas conexiones

```
module mux8_1(
input [7:0]D,
input [2:0]S,
output LED
   );
wire [6:1]C;

        mux2_1 m1 (
        .S(S[2]),
        .D0(C[1]),
```

```
.D1(C[2]),
.Y(LED)
);

mux2_1 m2 (
.S(S[1]),
.D0(C[5]),
.D1(C[6]),
.Y(C[1])
);

mux2_1 m3 (
.S(S[1]),
.D0(C[3]),
.D1(C[4]),
.Y(C[2])
);

mux2_1 m4 (
.S(S[0]),
.D0(D[0]),
.D1(D[1]),
.Y(C[5])
);

mux2_1 m5 (
.S(S[0]),
.D0(D[2]),
.D1(D[3]),
.Y(C[6])
);

mux2_1 m6 (
.S(S[0]),
.D0(D[4]),
.D1(D[5]),
.Y(C[3])
);

mux2_1 m7 (
.S(S[0]),
.D0(D[6]),
.D1(D[7]),
.Y(C[4])
);
```

endmodule

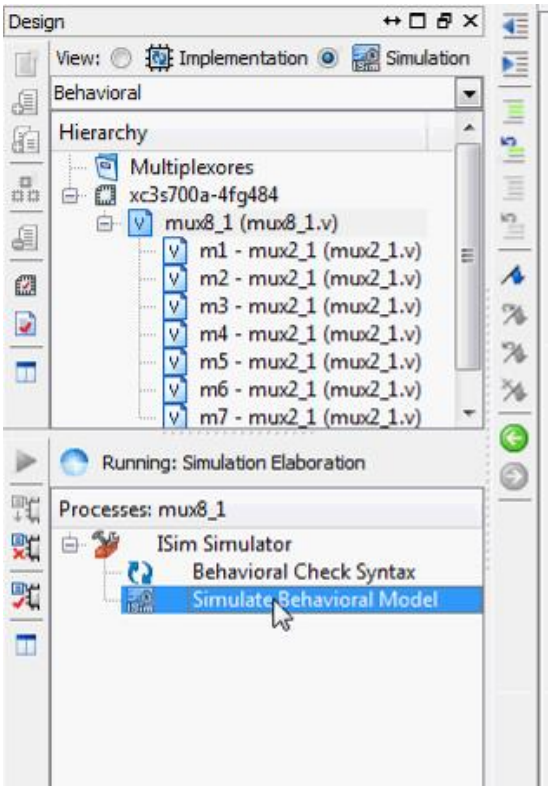
Se puede ver claramente cada una de las instanciaciones denominadas según la imagen anterior y las asignaciones de la misma manera. Por ejemplo, para el primer multiplexor de 2 a 1, llamado m1 el código escrito se indicó así:

```
mux2_1 m1 (          //Nombre asignado a la instanciación
.S(S[2]),        //Selector único en el multiplexor de dos a 1 que
                 corresponde al segundo en el multiplexor de 8 a 1
.D0(C[1]),       //Primera entrada que va conectada al cable número 1
.D1(C[2]),       //Segunda entrada que va conectada al cable número 2
.Y(LED)          //Salida conectada a la salida completa del mux de 8 a 1
);
```

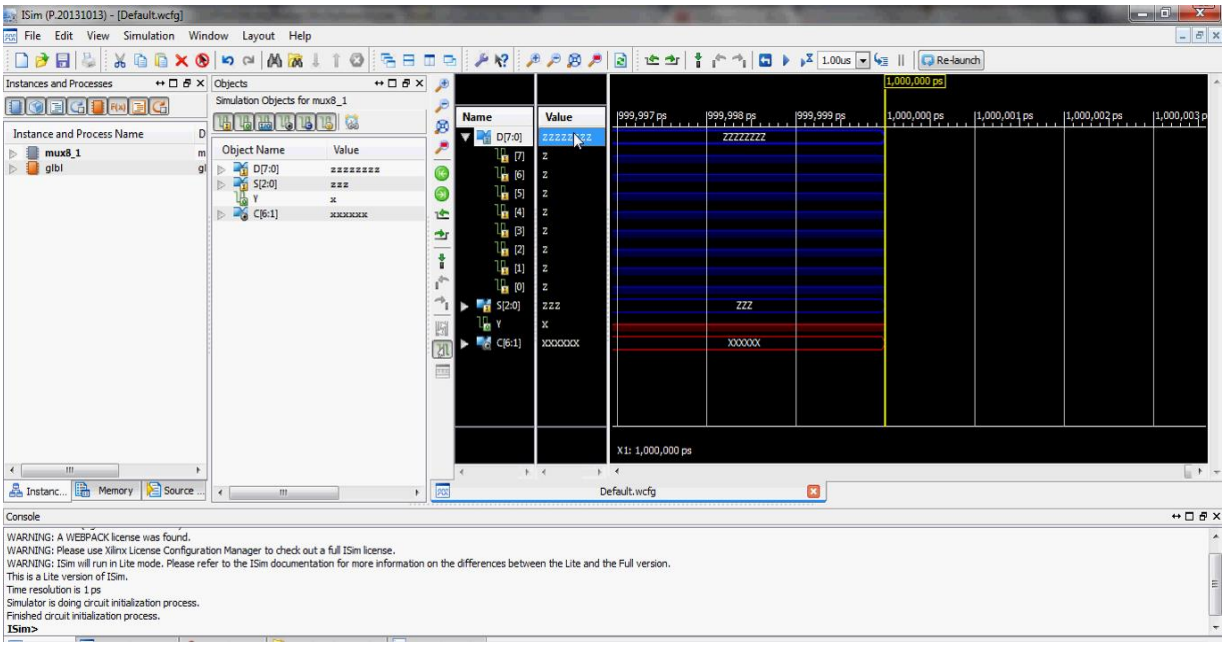
Comprobar la sintaxis nuevamente, de nuevo esto es indispensable para continuar con la descripción.

Si se desea, y con una tabla de verdad para un multiplexor de 8 a 1 a la mano se puede realizar la simulación de éste. (Recomendable)

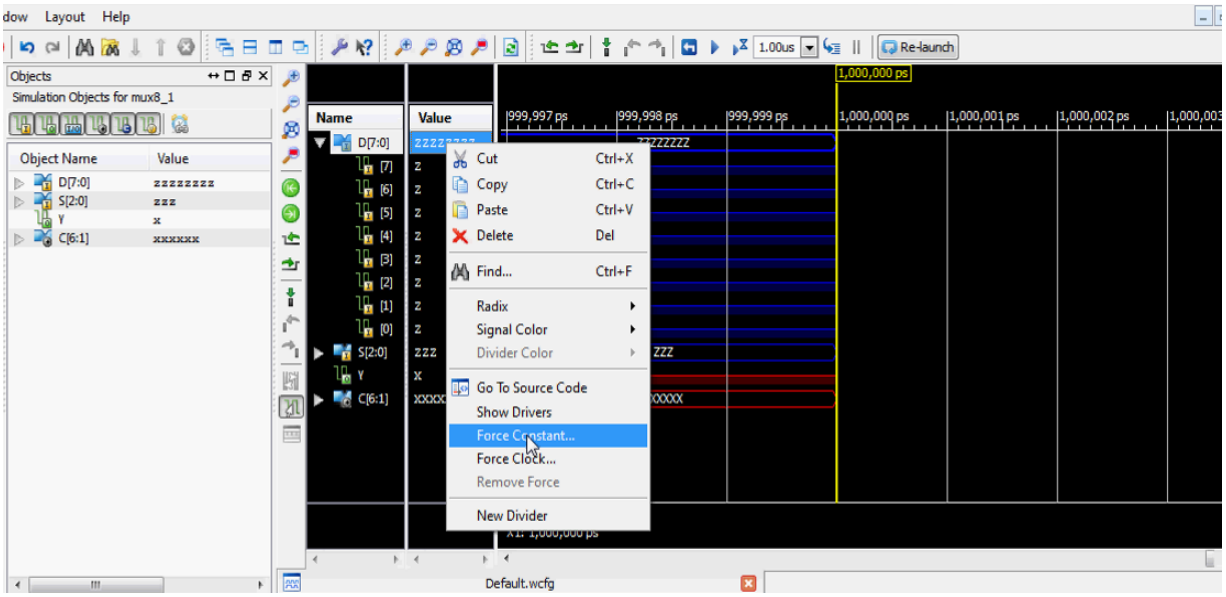
En el caso de realizarla, con la vista de simulación seleccionada y el módulo a simular, seleccionar *Simulate Behavioral Model*



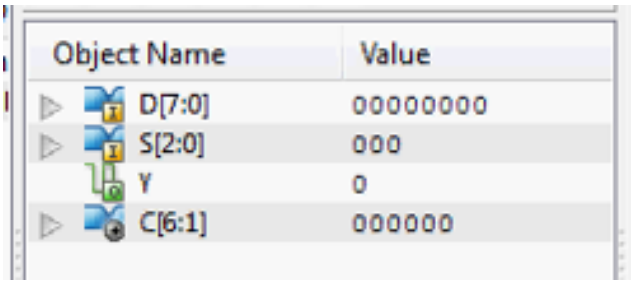
Se abre la ventana de simulación

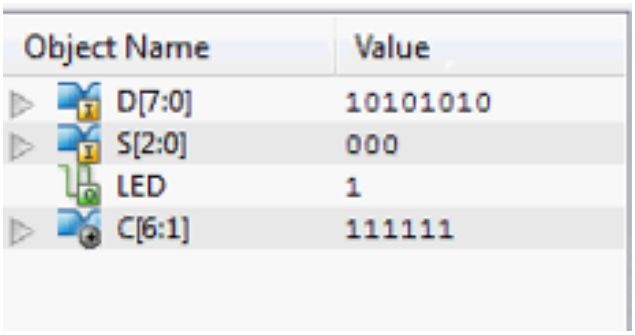


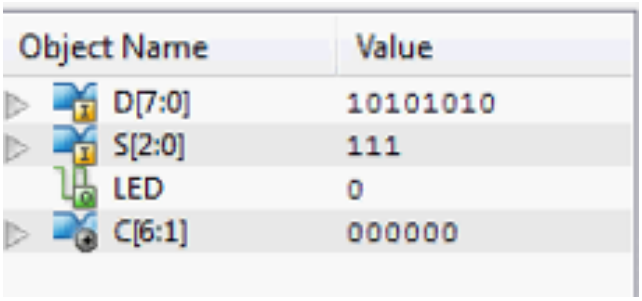
Forzar constante para cada una las entradas D_n y para los selectores S

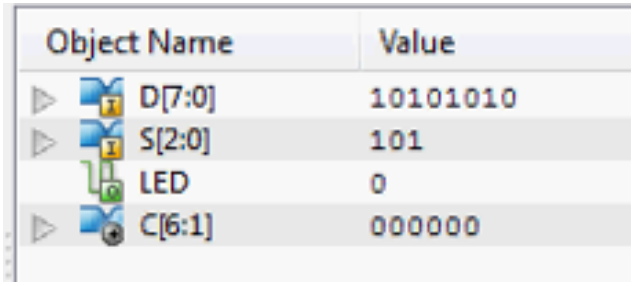


A continuación algunos ejemplos, para comprobar el correcto funcionamiento del circuito.







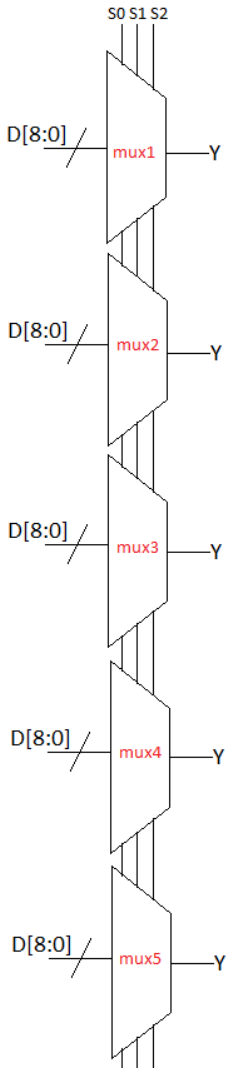


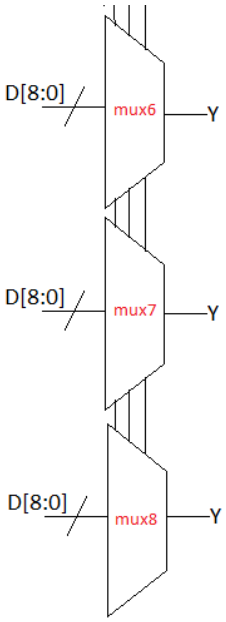
Si ya se ha comprobado que todo funciona correctamente, empezar a realizar la descripción del módulo para el circuito final.

Viendo nuevamente la tabla

SW [2:0]	Y
000	8'b10101010
001	8'b11001100
010	8'b10011001
011	8´b10100001
. . .	X X x

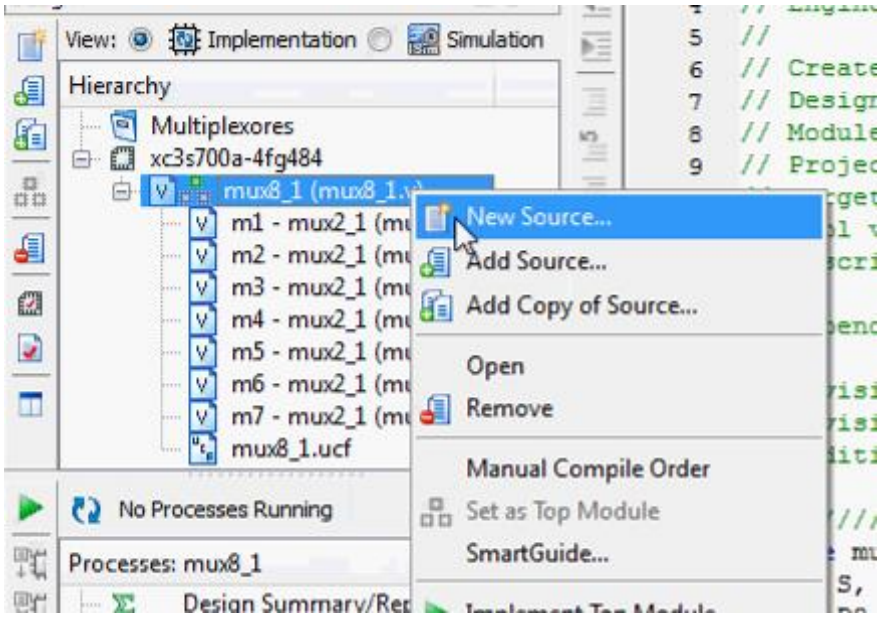
Se muestra que la salida Y se hace mediante 8 bits lo que físicamente en la FPGA se representaría por medio de LEDs, y ya que el multiplexor construido (mux de 8 a 1) tiene una sóla salida, se concluye que son necesarios ocho multiplexores de este tipo para la visualización de la salida Y, se puede representar el circuito obtenido de la siguiente manera:

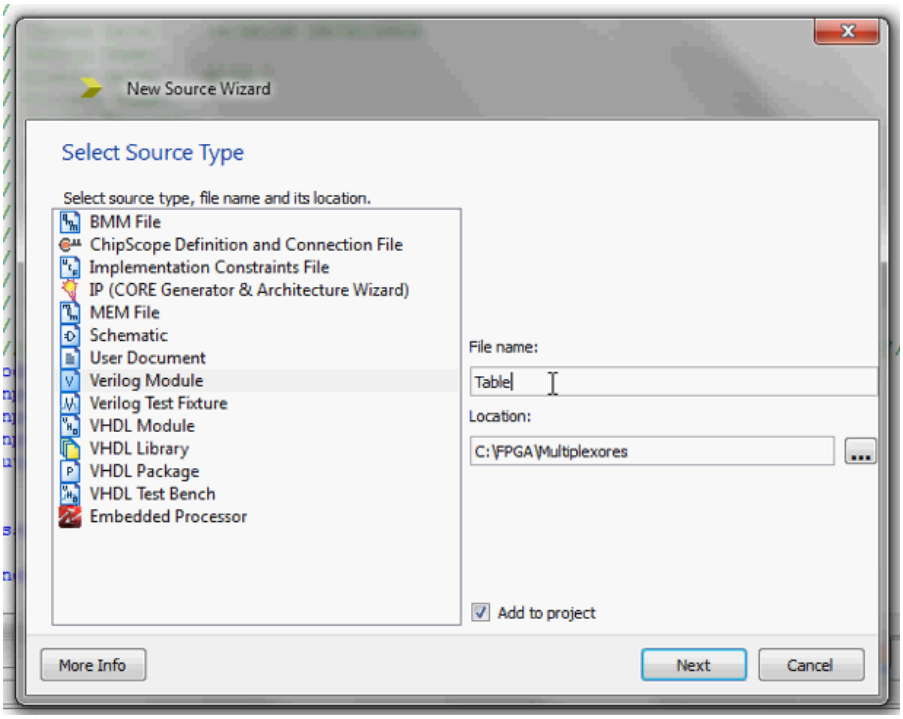




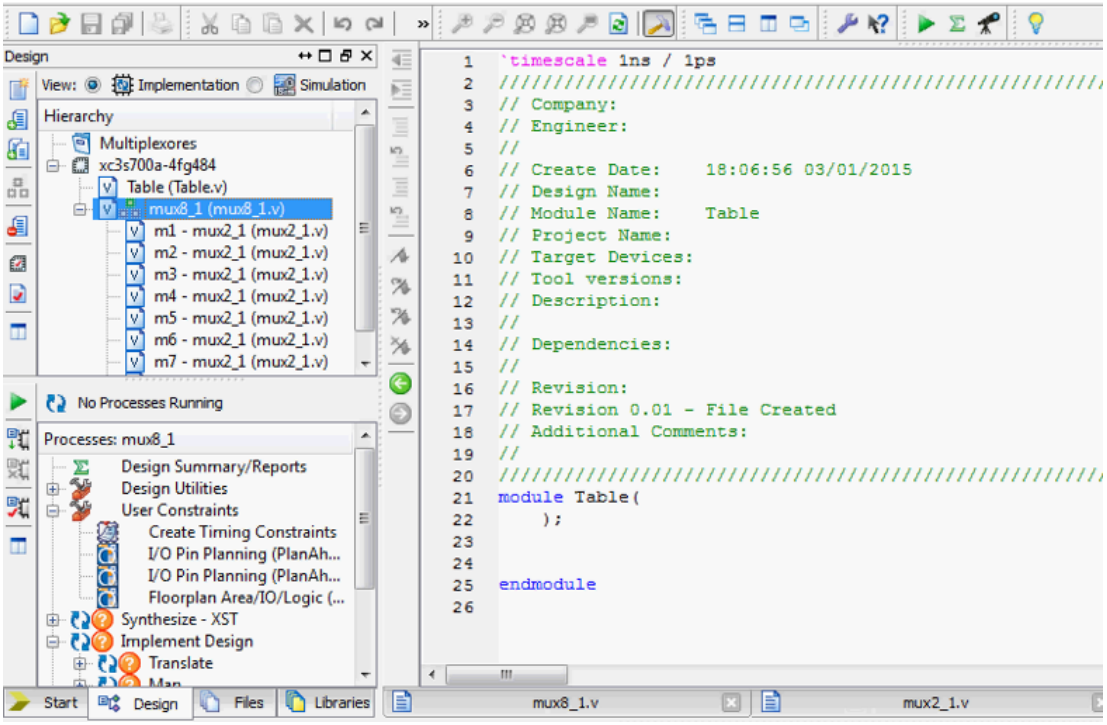
Ahora la descripción en Verilog

De nuevo:





Ya en la ventana del editor,



Y lo siguiente será describir el circuito completo para que funcione de la misma manera que lo indica la tabla, el código entonces:

```
module Table(
input [2:0]SW,
output [7:0]LED
    );
mux8_1 l0(
.D(8'b00110000),
.S(SW[2:0]),
.LED(LED[0])
);

mux8_1 l1(
.D(8'b10000000),
.S(SW[2:0]),
.LED(LED[1])
);

mux8_1 l2(
.D(8'b01000000),
.S(SW[2:0]),
.LED(LED[2])
);

mux8_1 l3(
.D(8'b11100000),
.S(SW[2:0]),
.LED(LED[3])
);

mux8_1 l4(
.D(8'b00100000),
.S(SW[2:0]),
.LED(LED[4])
);

mux8_1 l5(
.D(8'b10010000),
.S(SW[2:0]),
.LED(LED[5])
);

mux8_1 l6(
.D(8'b01000000),
.S(SW[2:0]),
.LED(LED[6])
);

mux8_1 l7(
.D(8'b11110000),
```

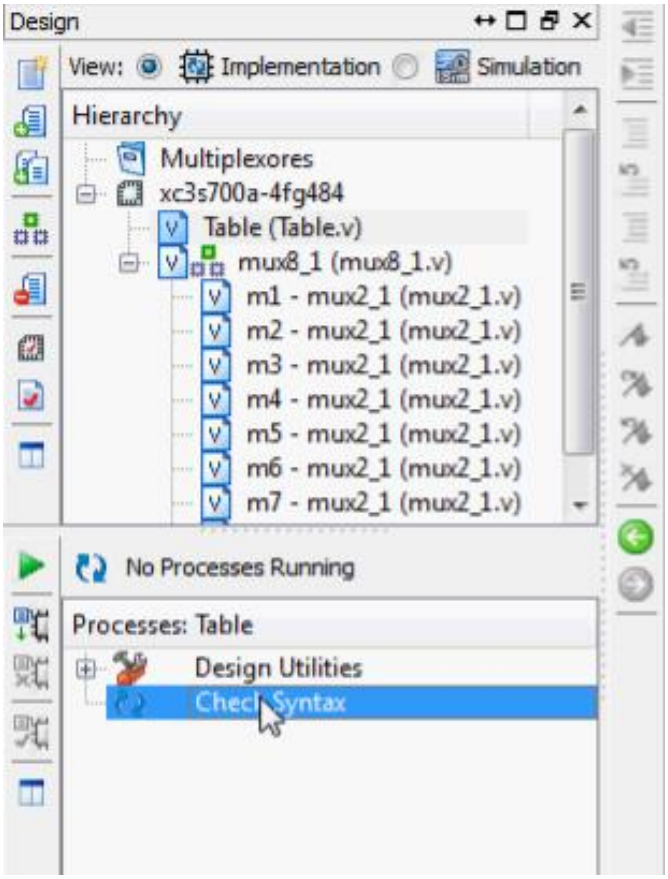
```
.S(SW[2:0]),
.LED(LED[7])
);
```

Endmodule

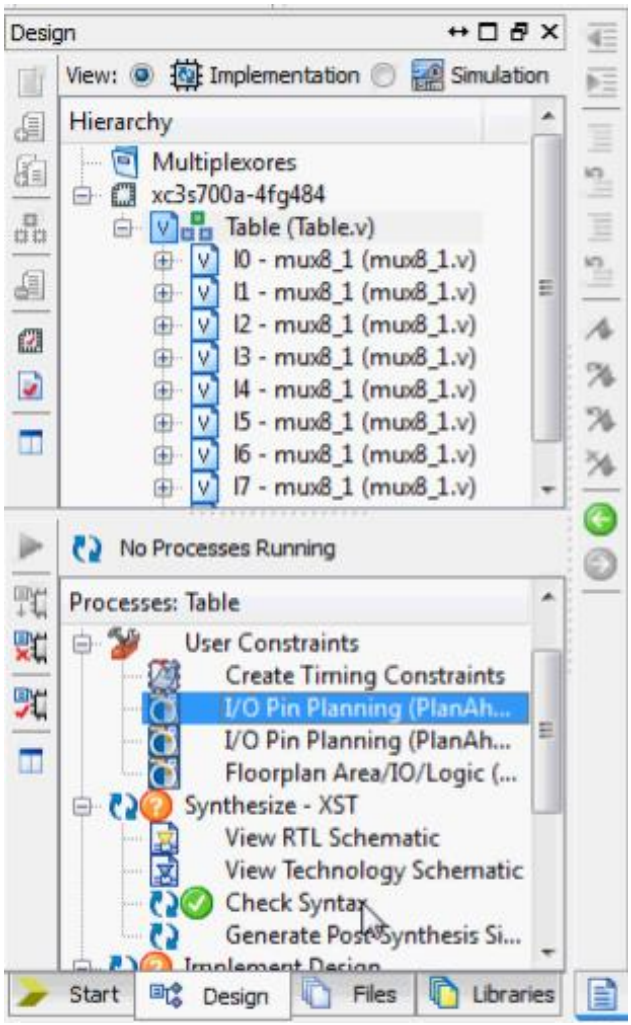
Donde,

- Define un bus de ocho salidas usando leds
- Define tres entradas manipuladas a través de tres interruptores
- Indica como constantes todas las entradas de los multiplexores de 8 a 1
- Instancia los ocho multiplexores asignando una configuración diferente para las entradas constantes, según sea necesario y un led diferente para cada salida Y

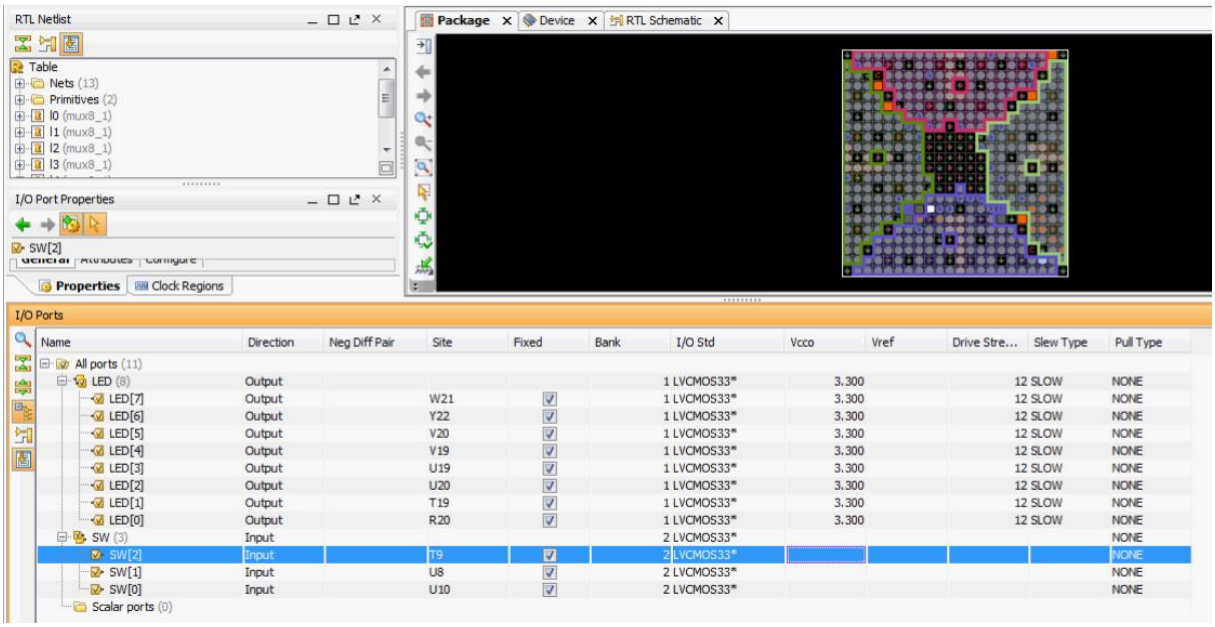
Ya asignado todo el código necesario, revisar que la sintaxis haya sido adecuada



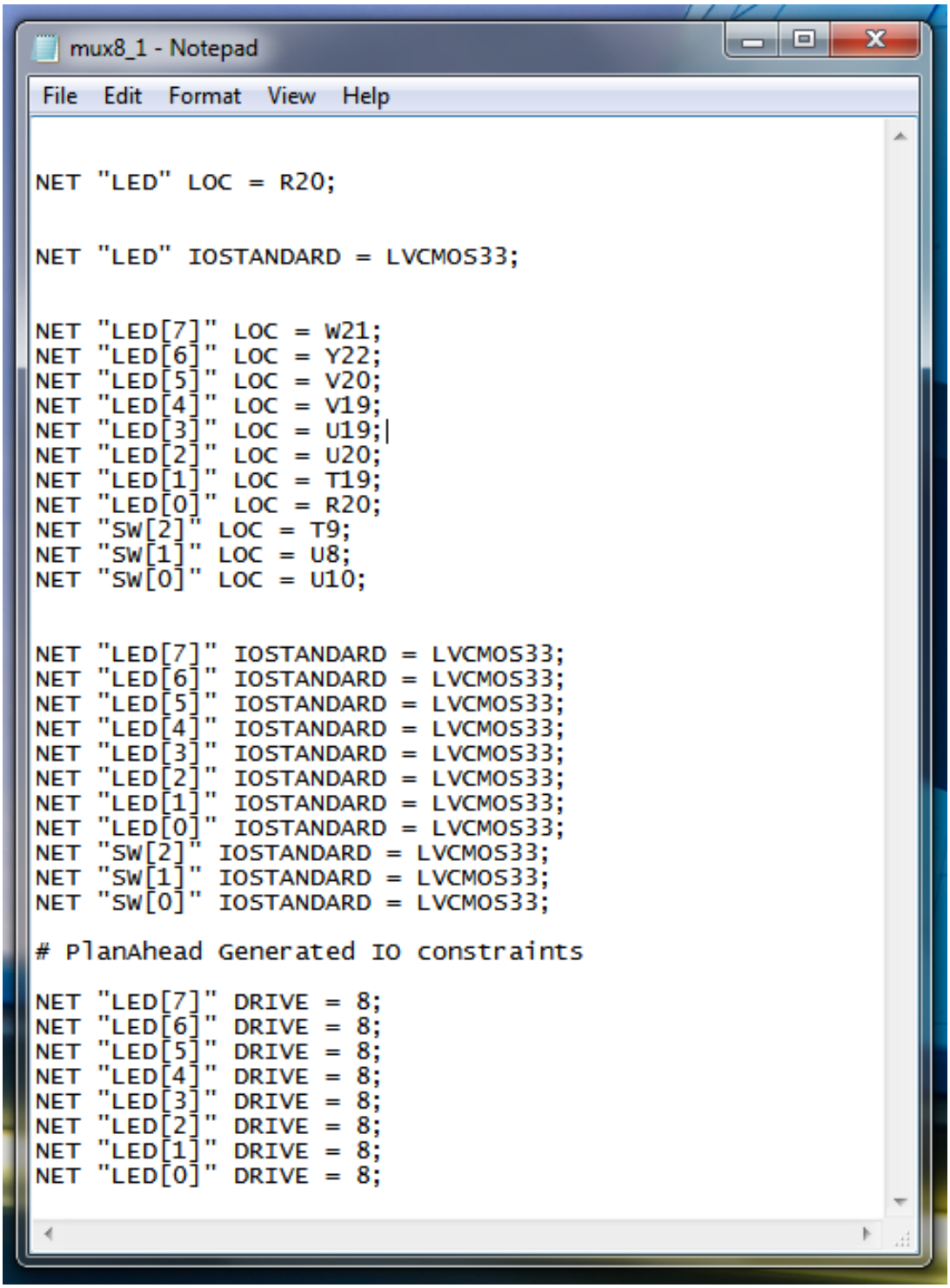
Ahora es necesario hacer las respectivas referencias a las entradas y salidas, para esto se pueden asignar mediante Xilinx PlanAhead de la siguiente forma



Asignados de acuerdo a la siguiente configuración

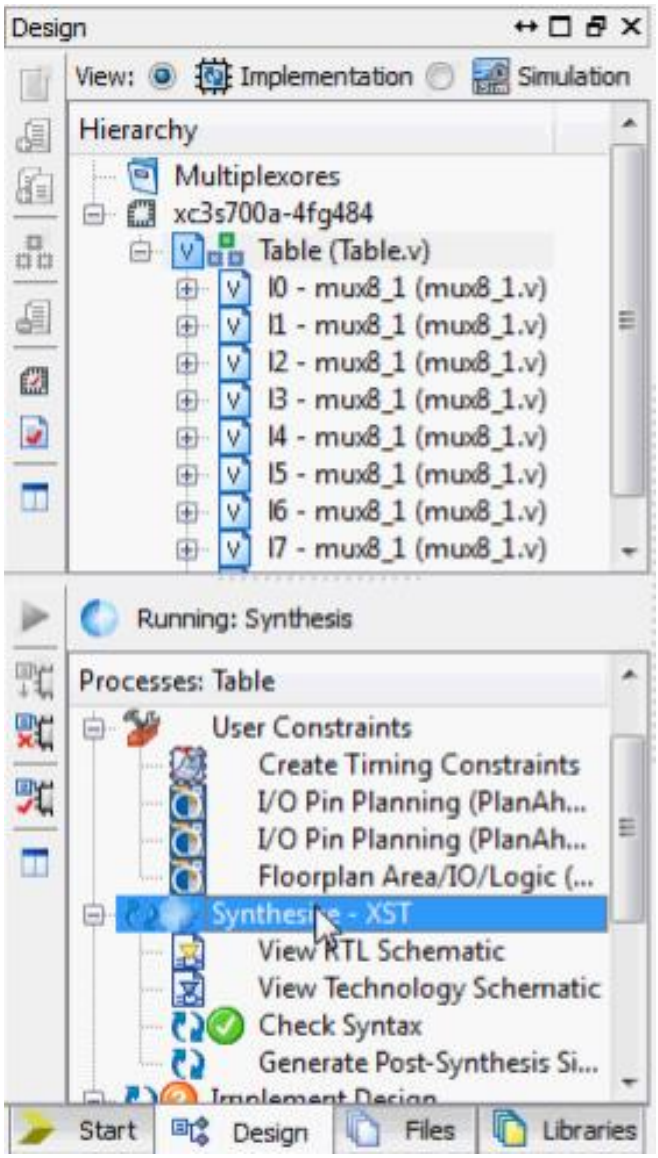


O bien, se puede editar directamente un archivo de tipo .ucf que contiene la misma información que permite relacionar el PlanAhead



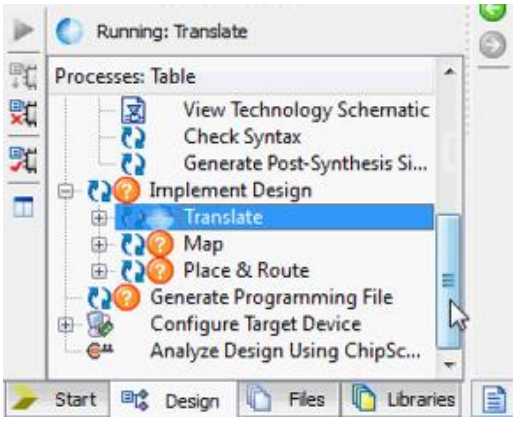
```
NET "LED" LOC = R20;
NET "LED" IOSTANDARD = LVCMOS33;


NET "LED[7]" LOC = W21;
NET "LED[6]" LOC = Y22;
NET "LED[5]" LOC = V20;
NET "LED[4]" LOC = V19;
NET "LED[3]" LOC = U19;
NET "LED[2]" LOC = U20;
NET "LED[1]" LOC = T19;
NET "LED[0]" LOC = R20;
NET "SW[2]" LOC = T9;
NET "SW[1]" LOC = U8;
NET "SW[0]" LOC = U10;


NET "LED[7]" IOSTANDARD = LVCMOS33;
NET "LED[6]" IOSTANDARD = LVCMOS33;
NET "LED[5]" IOSTANDARD = LVCMOS33;
NET "LED[4]" IOSTANDARD = LVCMOS33;
NET "LED[3]" IOSTANDARD = LVCMOS33;
NET "LED[2]" IOSTANDARD = LVCMOS33;
NET "LED[1]" IOSTANDARD = LVCMOS33;
NET "LED[0]" IOSTANDARD = LVCMOS33;
NET "SW[2]" IOSTANDARD = LVCMOS33;
NET "SW[1]" IOSTANDARD = LVCMOS33;
NET "SW[0]" IOSTANDARD = LVCMOS33;

# PlanAhead Generated IO constraints

NET "LED[7]" DRIVE = 8;
NET "LED[6]" DRIVE = 8;
NET "LED[5]" DRIVE = 8;
NET "LED[4]" DRIVE = 8;
NET "LED[3]" DRIVE = 8;
NET "LED[2]" DRIVE = 8;
NET "LED[1]" DRIVE = 8;
NET "LED[0]" DRIVE = 8;
```

Sintetizar para comprobar que el proceso de asignación de entradas y salidas se haya hecho correctamente

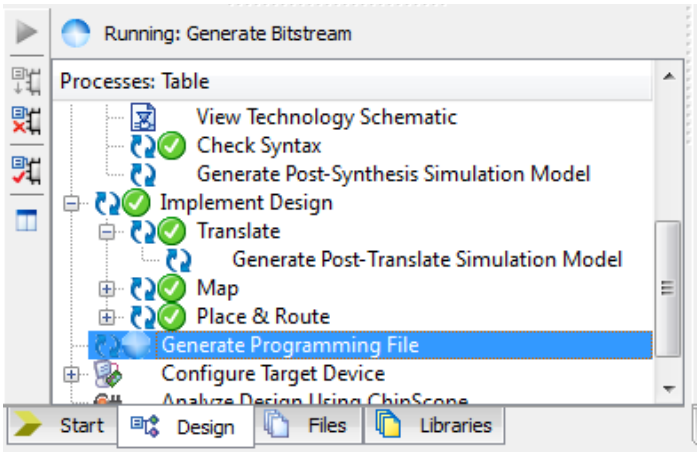


Y para finalizar el proceso de descripción y antes de poder visualizarlo en la FPGA

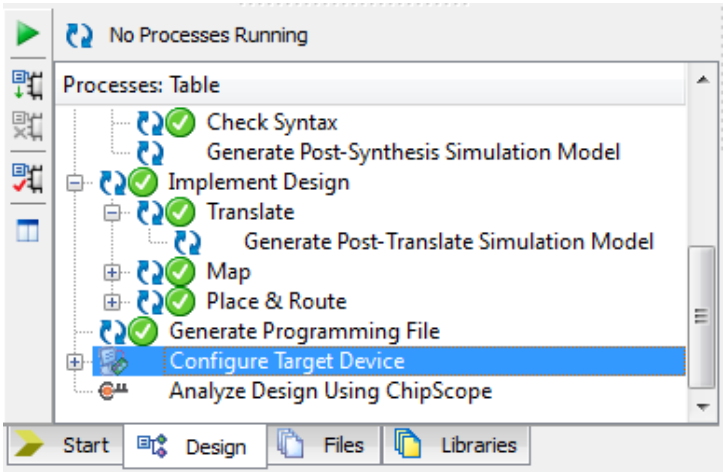
Implementar diseño:

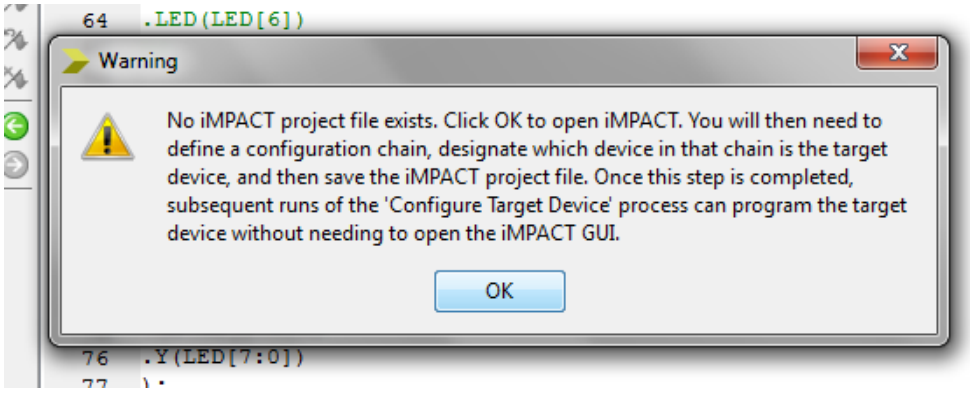


Generar archivo:



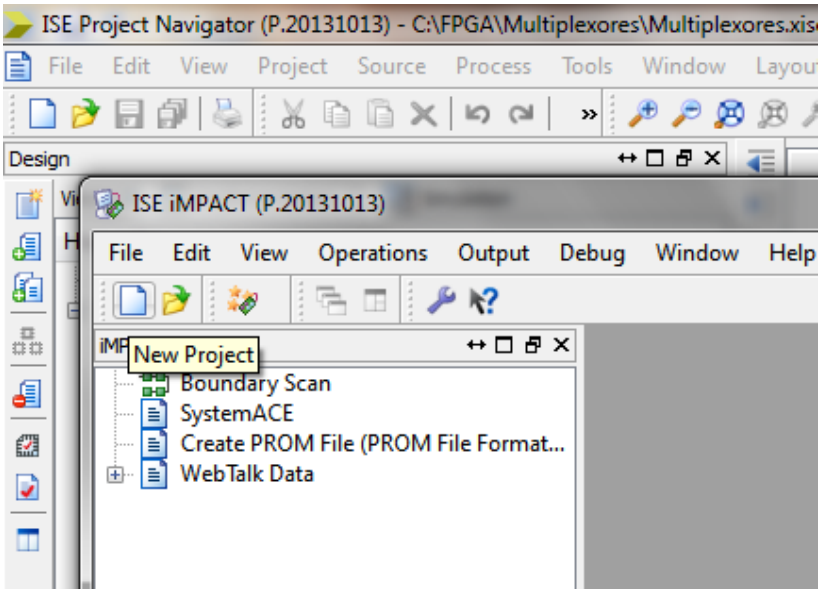
Y para finalizar



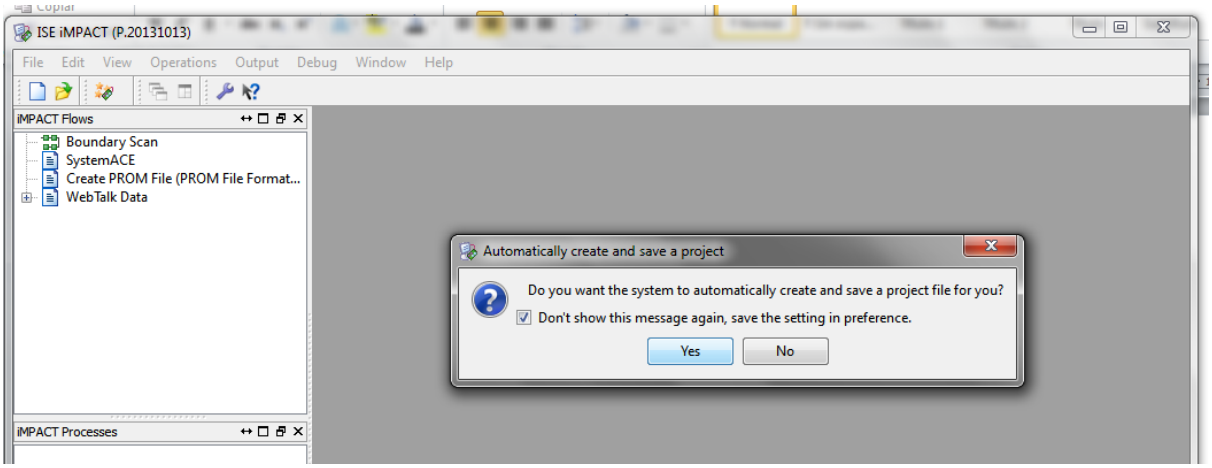


OK

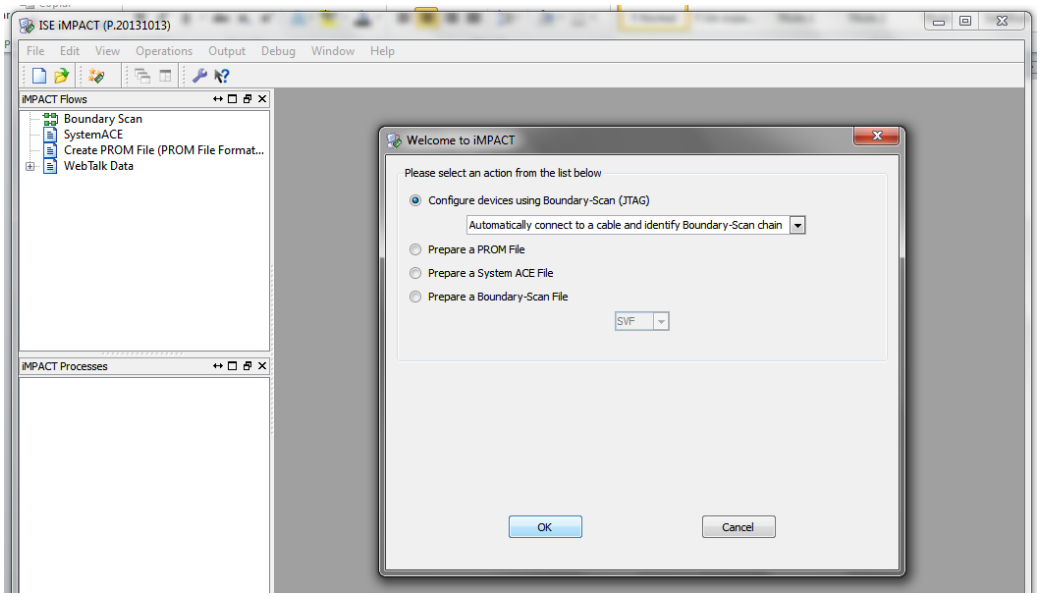
En la ventana del ISE iMPACT:



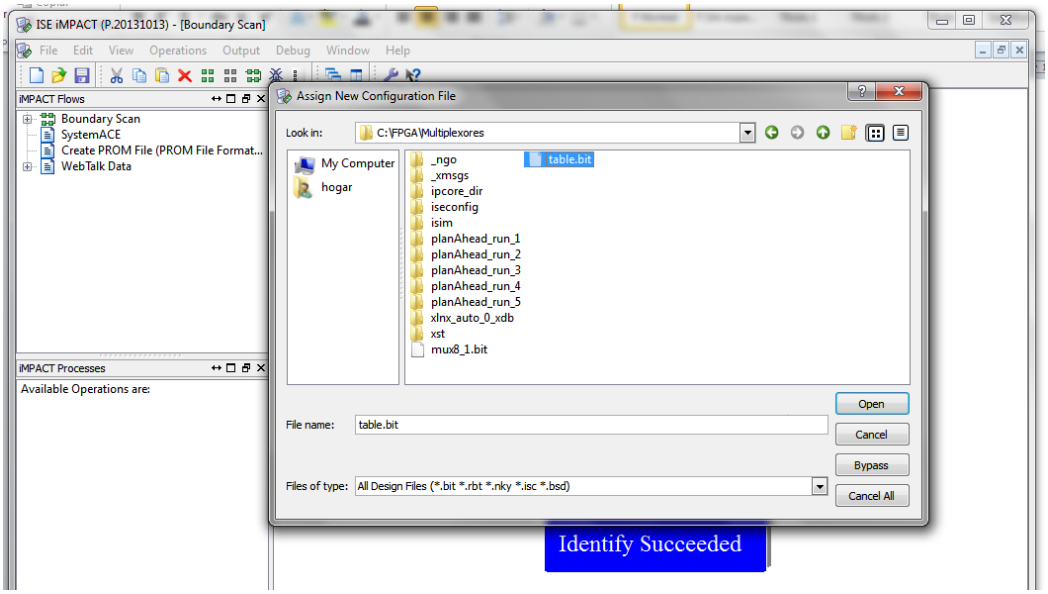
Crear nuevo proyecto



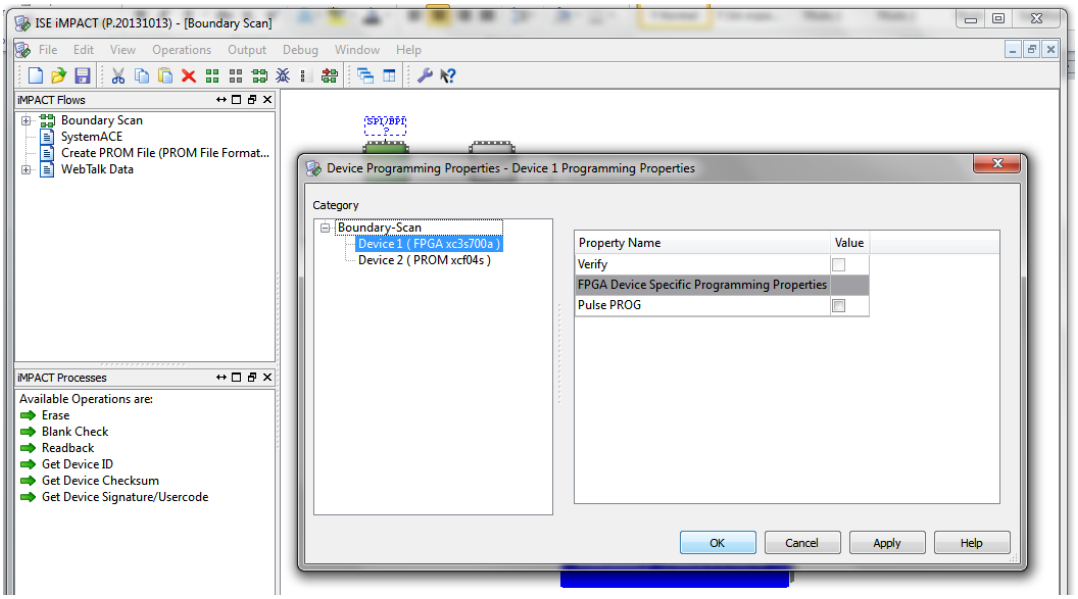
Yes



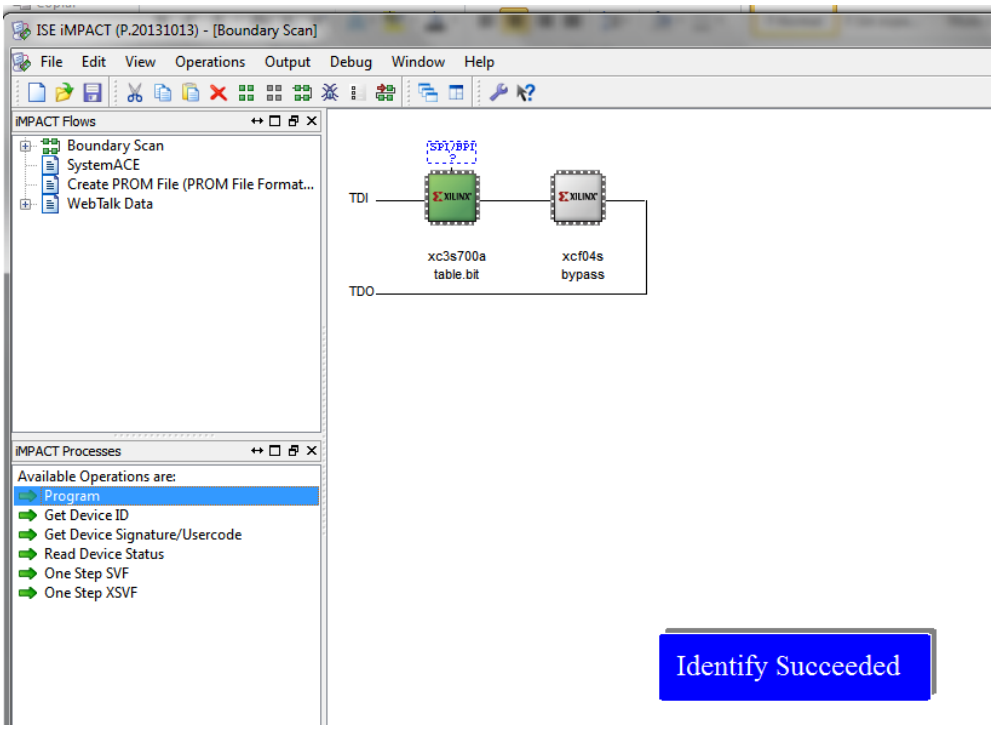
OK



Open table.bit



OK



Program

Ya debería correr correctamente en la FPGA.