Práctica 1: Primera implementación de circuito lógico en FPGA Xilinx-Spartan 3, Guía para Dummies

Universidad Pontificia Bolivariana

SECCIONAL BUCARAMANGA

Profesor encargado: Holguer Becerra
Realizado por: Ciro Alberto Gamboa

En esta guía paso a paso, se propone una tabla de verdad, cuyo circuito lógico hallado, se implementa posteriormente en la FPGA.

Objetivos:
-Practicar el diseño de circuitos lógicos utilizando los métodos de minterm y maxterm.
-Familiarizarse con las herramientas que ofrece Xilinx para el uso de sus FPGAs.
-Aprender la metodología básica para describir hardware.

Requisitos:
-FPGA Xilinx Spartan 3, con cable de alimentación y USB

-Computador con el software instalado y licenciado.

Nota:
Las FPGAs, son proveídas por la UPB, por lo que se debe tener cuidado al utilizarlas y se debe procurar seguir siempre las recomendaciones del profesor. SI no se tiene el software instalado, entrar al siguiente enlace :

http://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/design-tools/v14_1.html

Descargar una versión superior a la 14 e instalar el WebPack; es necesario registrarse en la página, descargar una licencia que se guarda en el computador y cargarla posteriormente seleccionando su archivo, seleccionando nuevamente WebPack.

Para más información, consultar con el profesor.

Desarrollo de la práctica:

-Se propone la siguiente tabla de verdad:

A	B	C	D	X	Y	Z
0	0	0	0	1	0	1
0	0	0	1	1	1	1
0	0	1	0	1	1	0
0	0	1	1	1	1	0
0	1	0	0	0	0	1
0	1	0	1	0	0	1
0	1	1	0	0	0	0
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	1	1	0
1	0	1	0	1	1	1
1	0	1	1	1	1	1
1	1	0	0	1	0	1
1	1	0	1	1	1	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

Esta tabla será resuelta por los métodos de **minterm** y **maxterm,** asumiendo un conocimiento previo de estos; para mayor información sobre estos métodos, visitar el siguiente enlace:

https://sites.google.com/site/ece31235upb/clases

Y seleccionar "Clase 2 del 26 enero y 2 Febrero".

X	**Y**	**Z**	Circuito X	Circuito Y	Circuito Z
			Maxterm	**Minterm**	**Minterm**
1	0	1	$A + B + C + D$	$\overline{A}B\overline{C}D$	$\overline{A}B\overline{C}\overline{D}$
1	1	1	$A + B + C + \overline{D}$	$\overline{A}\overline{B}\overline{C}D$	$\overline{A}\overline{B}\overline{C}\overline{D}$
1	1	0	$A + B + \overline{C} + D$	$\overline{A}\overline{B}CD$	$\overline{A}\overline{B}C\overline{D}$
1	1	0	$A + B + \overline{C} + \overline{D}$	$\overline{A}\overline{B}CD$	$\overline{A}\overline{B}CD$
0	0	1	$A + \overline{B} + C + D$	$\overline{A}B\overline{C}\overline{D}$	$\overline{A}B\overline{C}\overline{D}$
0	0	1	$A + \overline{B} + C + \overline{D}$	$\overline{A}B\overline{C}D$	$\overline{A}B\overline{C}D$
0	0	0	$A + \overline{B} + \overline{C} + D$	$\overline{A}BC\overline{D}$	$\overline{A}BC\overline{D}$
1	0	0	$A + \overline{B} + \overline{C} + \overline{D}$	$\overline{A}BCD$	$\overline{A}BCD$
0	1	0	$\overline{A} + B + C + D$	$A\overline{B}\overline{C}\overline{D}$	$A\overline{B}\overline{C}\overline{D}$
1	1	0	$\overline{A} + B + C + \overline{D}$	$A\overline{B}\overline{C}D$	$A\overline{B}\overline{C}D$
1	1	1	$\overline{A} + B + \overline{C} + D$	$A\overline{B}C\overline{D}$	$A\overline{B}C\overline{D}$
1	1	1	$\overline{A} + B + \overline{C} + \overline{D}$	$A\overline{B}CD$	$A\overline{B}CD$
1	0	1	$\overline{A} + \overline{B} + C + D$	$AB\overline{C}\overline{D}$	$AB\overline{C}\overline{D}$
1	1	0	$\overline{A} + \overline{B} + C + \overline{D}$	$AB\overline{C}D$	$AB\overline{C}D$
1	0	1	$\overline{A} + \overline{B} + \overline{C} + D$	$ABC\overline{D}$	$ABC\overline{D}$
1	1	0	$\overline{A} + \overline{B} + \overline{C} + \overline{D}$	$ABCD$	$ABCD$

Siendo así los circuitos para cada una de las salidas (X, Y, Z)son los siguientes:

Cabe resaltar que el diseño no va a ser el más adecuado, ya que es posible simplificar muchos más los circuitos con ayuda del algebra booleana. De igual manera el método de agrupación pudo no ser el indicado, se hizo de esta manera para posteriormente en Verilog, hacer una programación más didáctica y aprender los comandos básicos del lenguaje de descripción de hardware.

$X=(A+\bar{B}+C+D)(A+\bar{B}+C+\bar{D})(A+\bar{B}+\bar{C}+D)(\bar{A}+B+C+D)$
$Y=\bar{A}\bar{B}\bar{C}D+\bar{A}\bar{B}C\bar{D}+\bar{A}\bar{B}CD+A\bar{B}\bar{C}\bar{D}+A\bar{B}\bar{C}D+A\bar{B}C\bar{D}+A\bar{B}CD+AB\bar{C}\bar{D}+ABC\bar{D}+ABCD$
$Z=\bar{A}\bar{B}\bar{C}\bar{D}+\bar{A}\bar{B}\bar{C}D+\bar{A}B\bar{C}\bar{D}+\bar{A}BC\bar{D}+A\bar{B}C\bar{D}+AB\bar{C}\bar{D}+AB\bar{C}D+ABC\bar{D}$

El lenguaje de descripción de hardware que se usará, es Verilog; a continuación se presenta el código respectivo a las ecuaciones de circuito que recientemente se hallaron; abrir bloc de notas e ingresar el código.

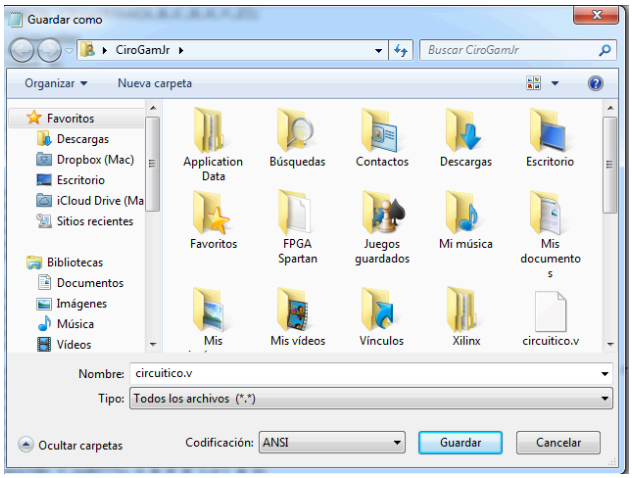
```
module circuitico(A,B,C,D,X,Y,Z);

//Entradas
input A;
input B;
input C;
input D;

//Salidas
output X;
output Y;
output Z;

//Cables;estos son uniones físicas
wire X_sub1;
wire X_sub2;
wire X_sub3;
wire X_sub4;
wire [8:0]Y_sub;
wire [7:0]Z_sub;

//Los X_subn son términos de la ecuación
assign X=X_sub1 & X_sub2 & X_sub3 & X_sub4;
assign X_sub1=A|(~B)|C|(~D);
assign X_sub2=A|(~B)|(~C)|D;
assign X_sub3=(~A)|B|C|D;
assign X_sub4=A|(~B)|C|D;

assign Y= | Y_sub[8:0]; //a la salida Y le asigna la logica de el vector Y_sub relacionados con or
assign Y_sub[0]=(~A) & (~B) & (~C) & D;
assign Y_sub[1]=(~A) & (~B) & C & (~D);
assign Y_sub[2]=(~A) & (~B) & C & D;
assign Y_sub[3]= A & (~B) & (~C) & (~D);
assign Y_sub[4]= A & (~B) & (~C) & D;
assign Y_sub[5]= A & (~B) & C & (~D);
assign Y_sub[6]= A & (~B) & C & D;
assign Y_sub[7]= A & B & (~C) & D;
assign Y_sub[8]= A & B & C & D;

assign Z= | Z_sub[7:0];
assign Z_sub[0]=(~A) & (~B) & (~C) & (~D);
assign Z_sub[1]=(~A) & (~B) & (~C) & D;
assign Z_sub[2]=(~A) & B & (~C) & (~D);
assign Z_sub[3]= A & (~B) & C & (~D);
assign Z_sub[4]= A & (~B) & C & D;
assign Z_sub[5]= A & B & (~C) & (~D);
assign Z_sub[6]= A & B & C & (~D);
assign Z_sub[7]= (~A) & B & (~C) & D;


endmodule
```

Es de suma importancia comprender que estás declaraciones y asignaciones, hacen referencia a componentes y variables físicas de los circuitos internos configurables de la FPGA.

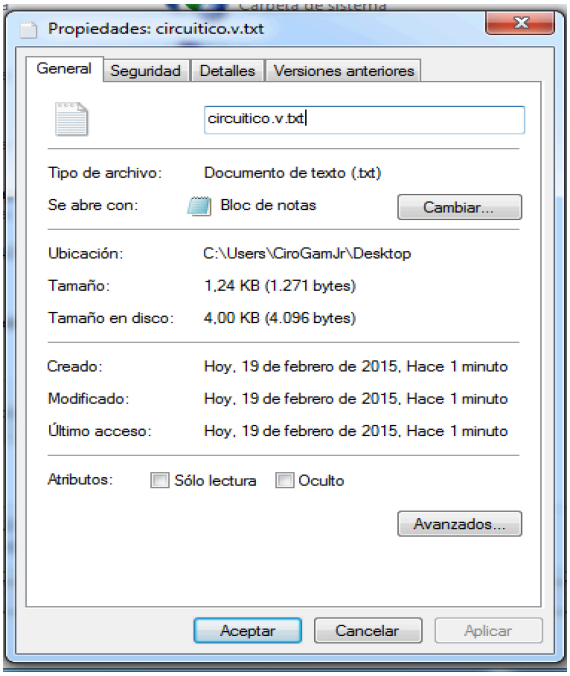
Algunas observaciones:

-Los arrays son declarados con el rango de mayor a menor y a la izquierda de su respectivo nombre y para ser llamados en alguna posición respectiva, el rango va a la derecha**: wire [8:0]Y_sub; .... assign Z_sub[0]= ....**

-And, or y not, son representadas de esta forma; **&, | , ~** .



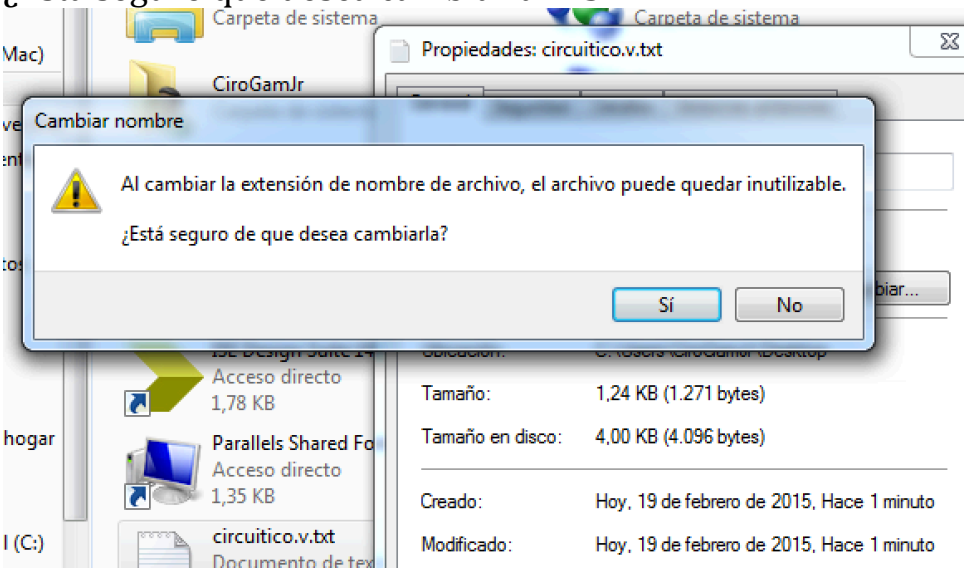
-Siempre es necesario inicializar el módulo de trabajo y cerrarlo. Luego de escribir el código, se guarda el archivo de la siguiente manera, recordar poner el **.v** y en tipo, colocar “Todos los archivos”.



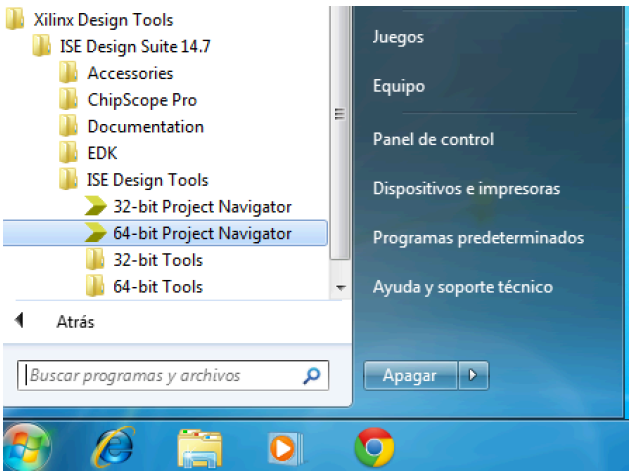
Tras guardar el archivo, buscamos su ubicación y con click derecho, abrir propiedades, borrar .txt

Acá se ilustra una manera alternativa de generar el archivo con la lógica que será programada en la FPGA, no es necesario usar bloc de notas todo el tiempo.

¿Está seguro que desea cambiarla? - Si

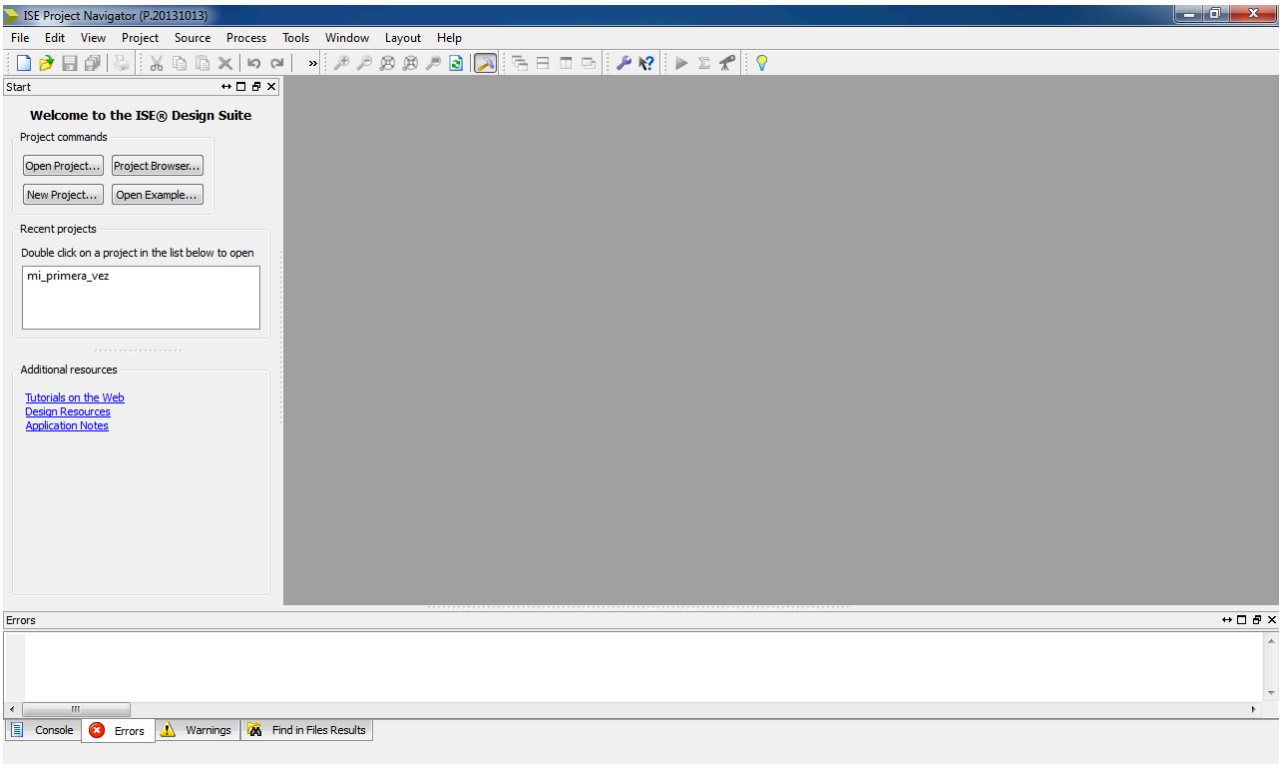


Se procede a abrir el navegador de proyectos, escoger de acuerdo a si su equipo es de 32 o de 64 bits; si es de 64 el de 32 también sirve.

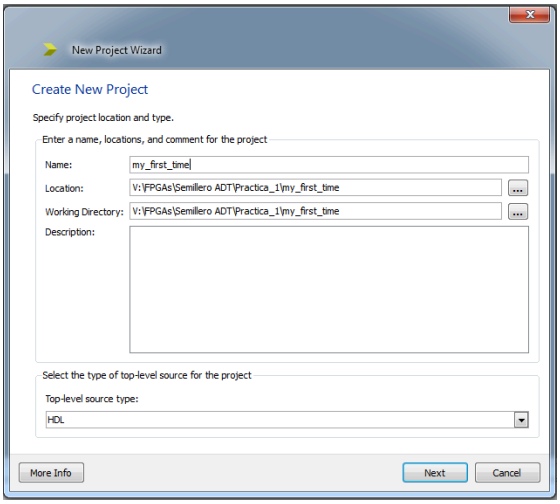


El navegador de proyectos es una de varias herramientas ofrecidas por Xilinx, en el cual se visualizan fácilmente los procesos, los archivos y describe un proceso debidamente estructurado para implementar finalmente el diseño en la FPGA.

Nuestro navegador de proyectos, luce de esta manera, es bueno permitirse explorar algunos iconos y funciones que este ofrece.

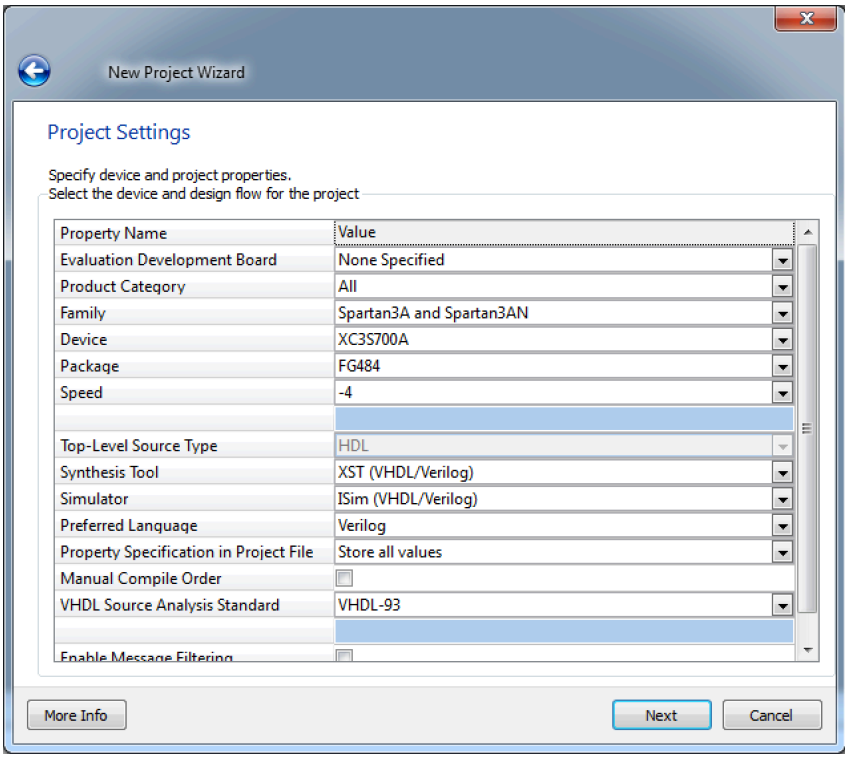


Seleccionamos “New Project” y se desplegará una ventana de esta manera:

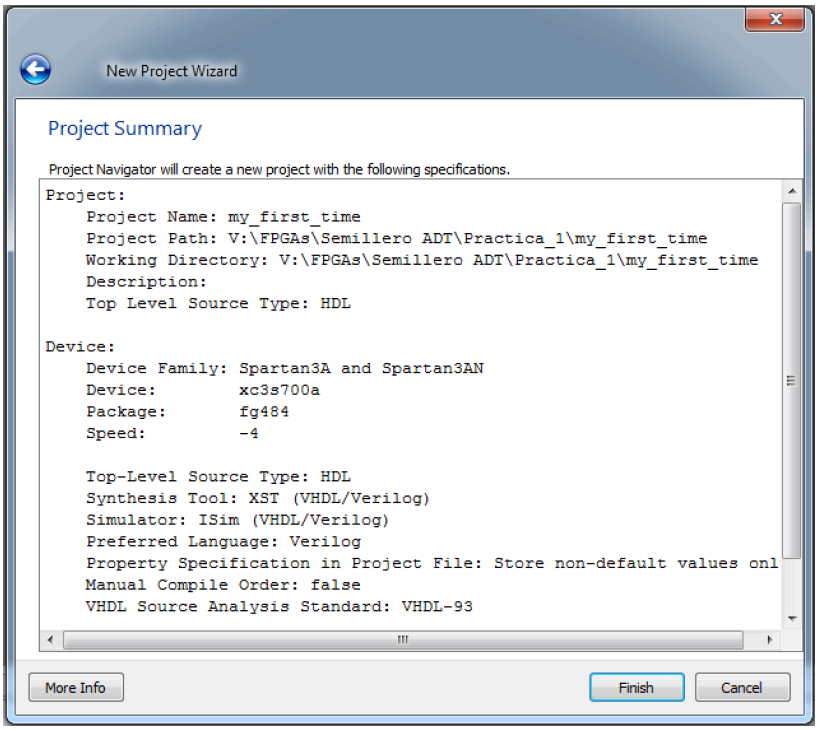


Se elige un nombre para el proyecto y se guarda en alguna ubicación, es recomendable crear una carpeta para todos los proyectos elaborados con este tipo de FPGA.

La configuración del proyecto es de extrema importancia, ya que allí le decimos al software, con que vamos a trabajar y ese acople es fundamental para el correcto funcionamiento de los circuitos creados. Para visualizar estos datos, mirar el chip principal de la FPGA, o el datasheet respectivo a la tarjeta.

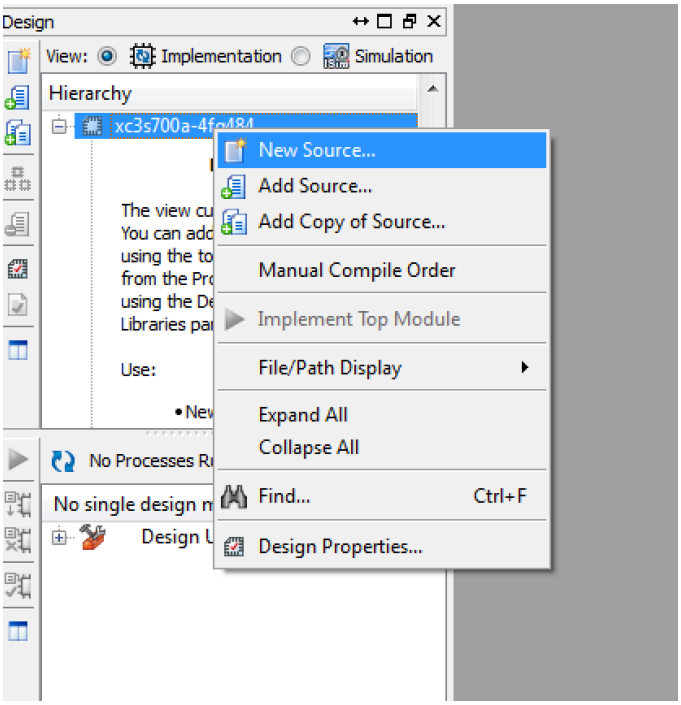


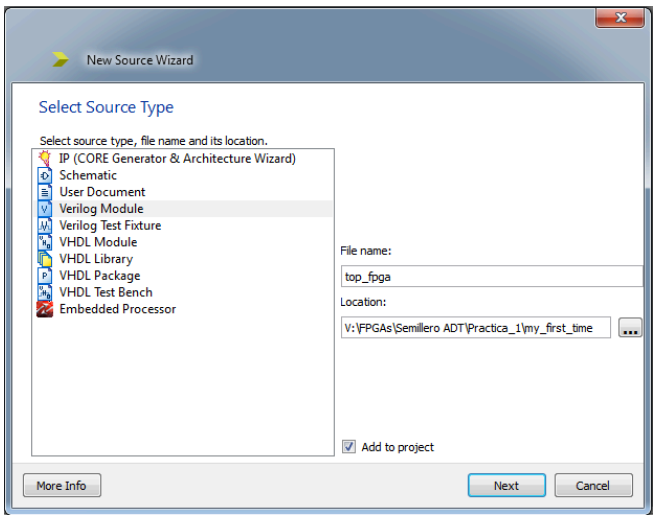
Dar click en “Next”



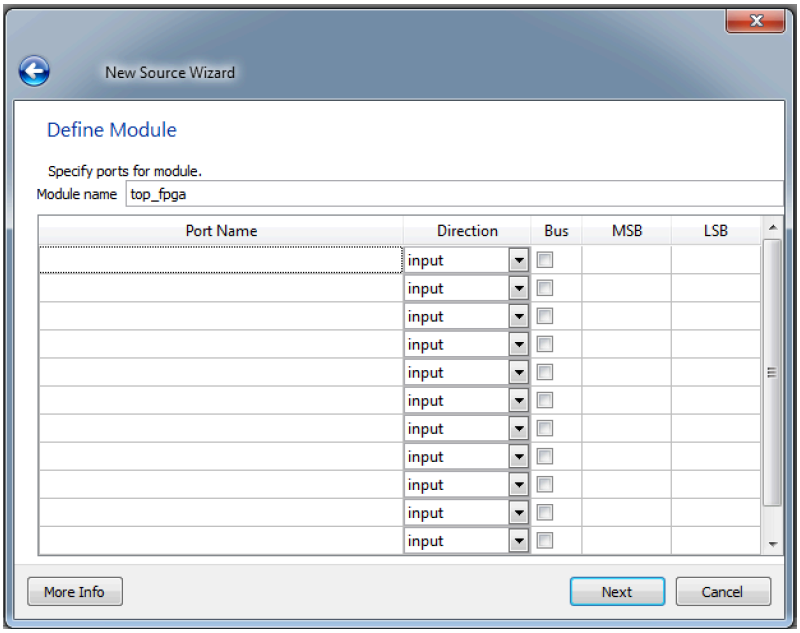
En este siguiente paso, se muestra un resumen de las opciones escogidas para el proyecto; presionar “Finish”.

La jerarquía describe el orden en que se llevan los procesos, de acuerdo a los módulos predominantes, y por fases del proyecto; manteniendo la vista en implementación, añadimos "New Source" con click derecho, a al figura con el símbolo del chip. Se selecciona "Verilog Module" y le damos un nombre al archivo.

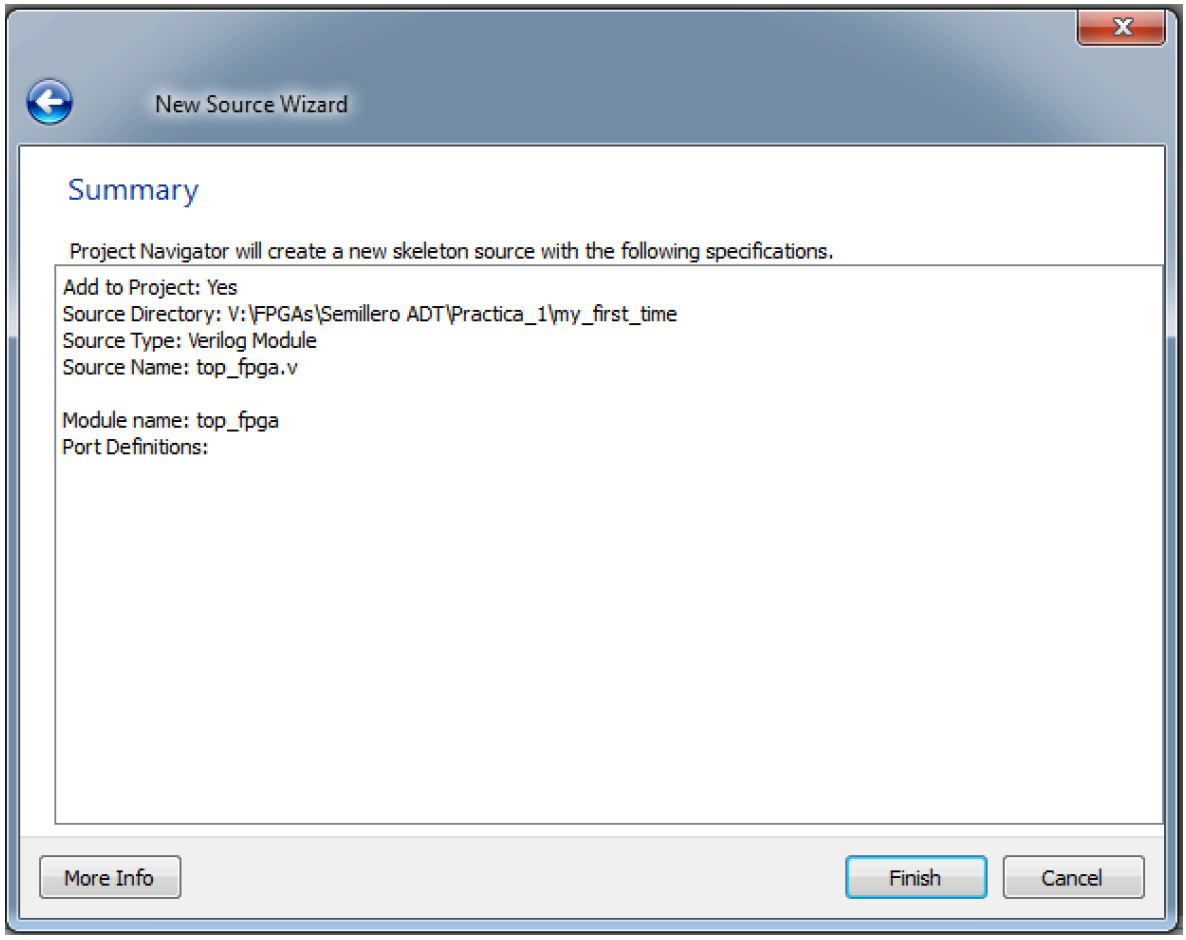




Por ahora se deja el archivo de manera predeterminada, así que damos "Next".



Aquí se presenta de nuevo un resumen de la configuración recientemente hecha, se da click en “Finish”.



De esta manera se produce un archivo programable, con una plantilla que sirve para una organización correcta del código. Este será el módulo “jefe”, en donde se realizara la transferencia de operaciones y variables simbólicas, a objetos reales de la FPGA.

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date:    14:50:08 02/19/2015
7  // Design Name:
8  // Module Name:    top_fpga
9  // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////////////////////////////////////
21 module top_fpga(
22     );
23
24
25 endmodule
26
```

Teniendo en cuenta el código escrito en el bloc de notas, y denotando las entradas como A,B,C y D así como las salidas X,Y y Z; asignamos el tipo de descripción circuital-lógica que se implementara, mediante Switches y Leds.

```
20 //////////////////////////////////////////////////////////////////////////////////
21 module top_fpga(SW,LED)  //Tipo de entrada (Switch),tipo de salida (Led)
22     );
23
24     // Entradas desde el switch 0 al switch 3
25     input[3:0]SW;
26     // Salidas desde el led 0 al led 7,solo usamos como respuesta al los switches, del 0 al 3
27     output[7:0]LED;
28     //Asignación de estado bajo para los leds que no se van a usar,para alto, poner 1
29     assign LED[7:3]=5'b00000;
30
31     //Asignación de variables simbólicas en código, a salidas y entradas físicas
32     circuitico ejercicio1(
33     .A(SW[0]),
34     .B(SW[1]),
35     .C(SW[2]),
36     .D(SW[3]),
37
38     .X(LED[2]),
39     .Y(LED[1]),
40     .Z(LED[0])
41     );
42
43
44
45 endmodule
46
```

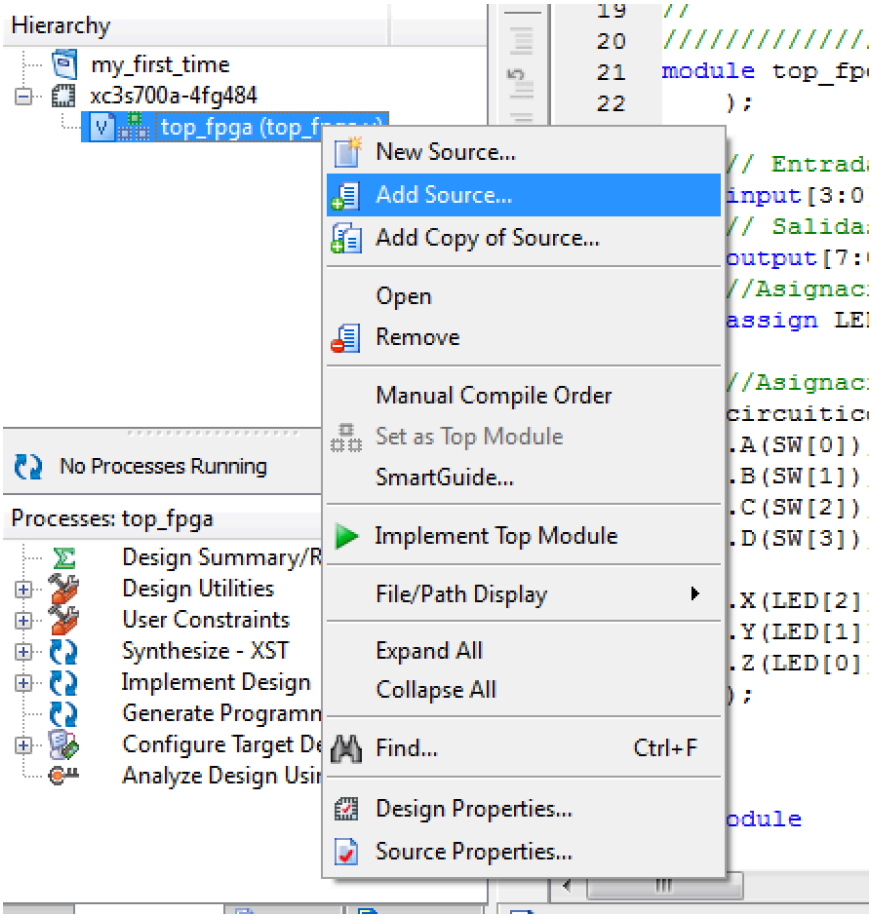
Algunas observaciones:

-Se denotan los switches y los leds, mediante arrays.

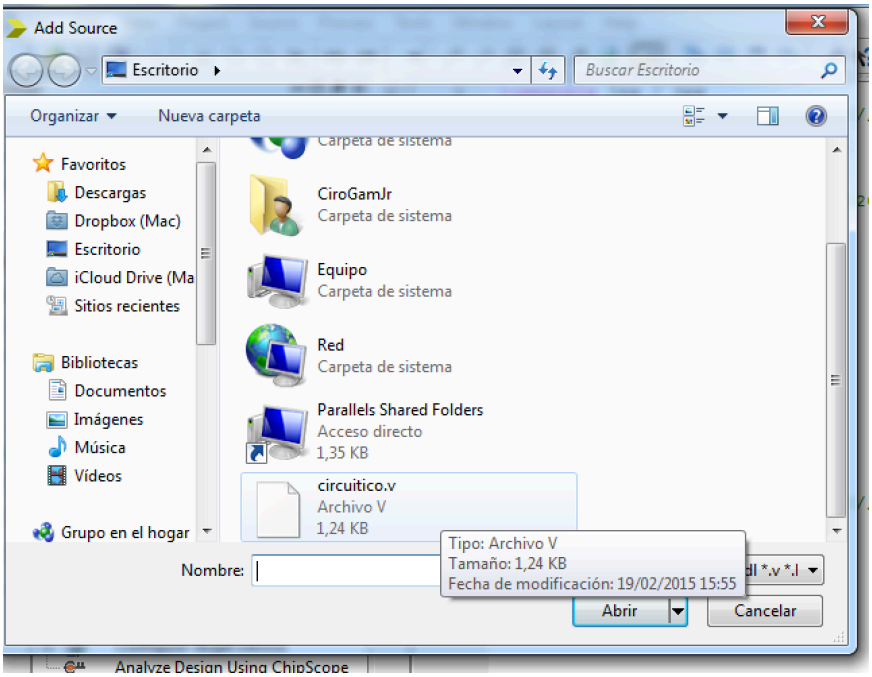
-assign LED[7:3]=5'b000000, lo que hace es configurar los 5 leds seleccionados inicialmente en binario, estado bajo.

-Se declara circuitico(el módulo hecho en bloc de notas) y un apodo para el, “ejercicio 1”, por dentro de el se hace la asignación simbólico-física, ya que ese módulo es quien contiene los parámetros simbólicos del circuito a realizar.

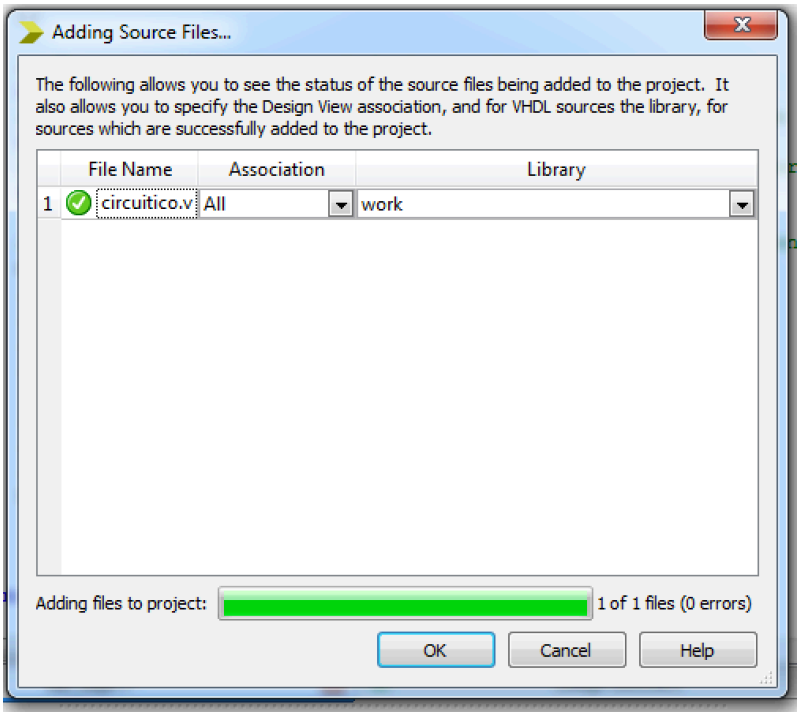
Añadir un nuevo módulo a top_fpga (Nuestro top_module o papá módulo), que será configurado como hijo, usamos “Add Source” porque este hijo, ya lo hemos creado:



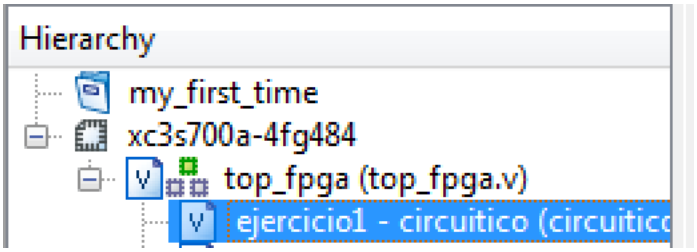
¡Así es!, seleccionamos circuitico.v

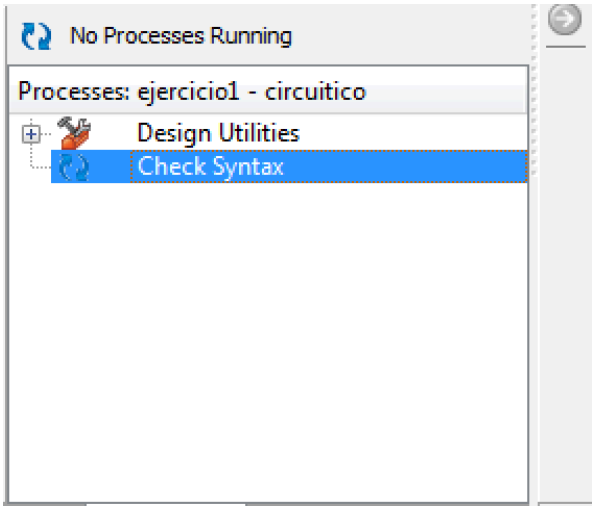


El chulito verde siempre es una buena noticia, se cargó correctamente el archivo creado previamente a nuestro proyecto.



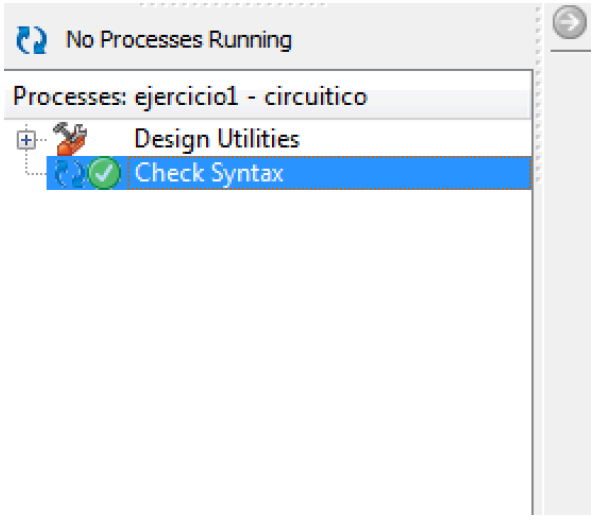
De esta manera luce circuitico, en la jerarquía; seleccionarlo



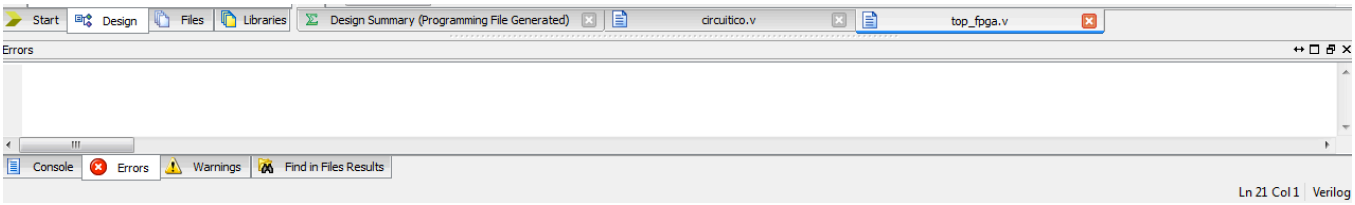


Es recomendable que el primer proceso que se realice con los módulos ya escritos, sea verificar si están bien escritos, para ello se usa: "Check Syntax".

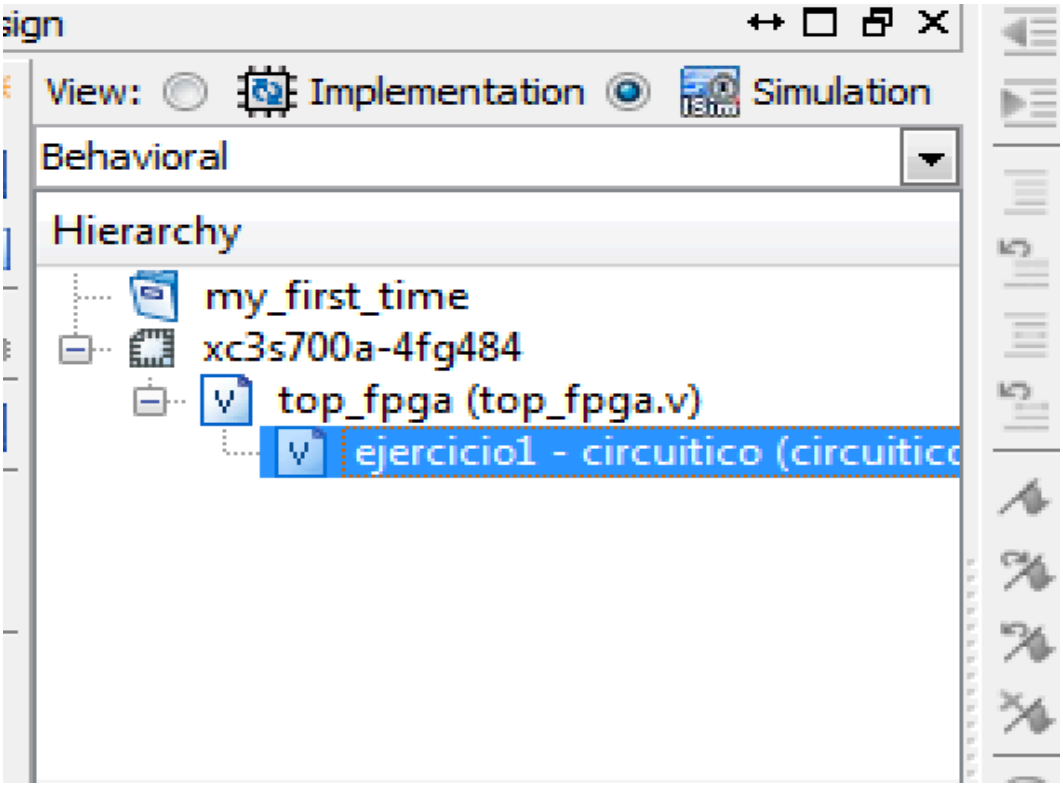
No hay problema con la sintaxis



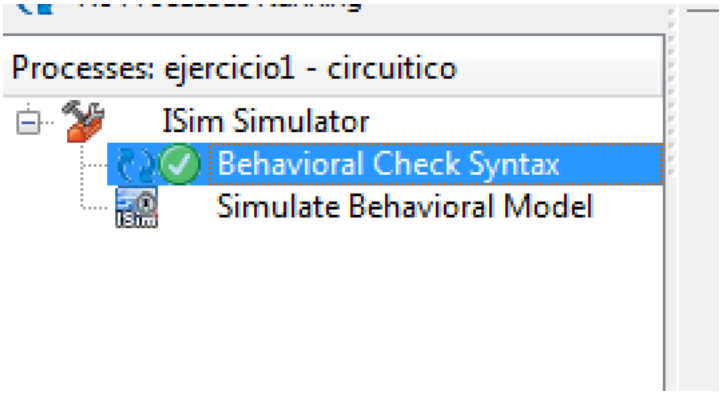
Pero si hay problemas, serán denotados en la parte inferior de la ventana del programa, nos dirá específicamente que tenemos como problema y donde; entonces se procedería a solucionarlo para poder continuar.



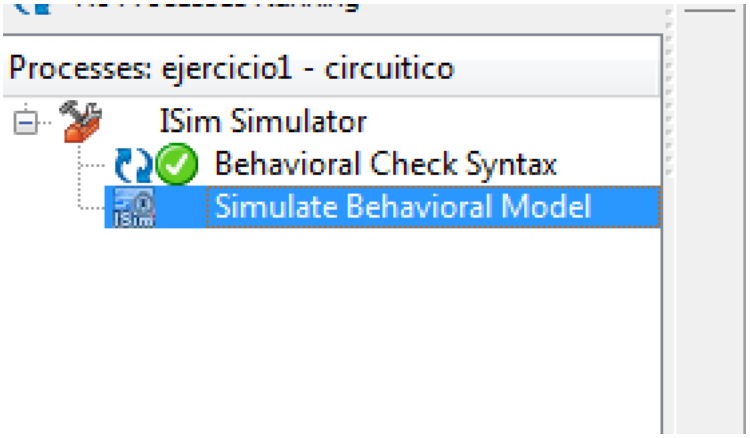
Ahora, nos pasamos a la vista de simulación y seleccionamos circuitico; siempre se debe probar el correcto funcionamiento del circuito mediante la simulación, de esta manera se evitan errores y posibles daños en la FPGA.



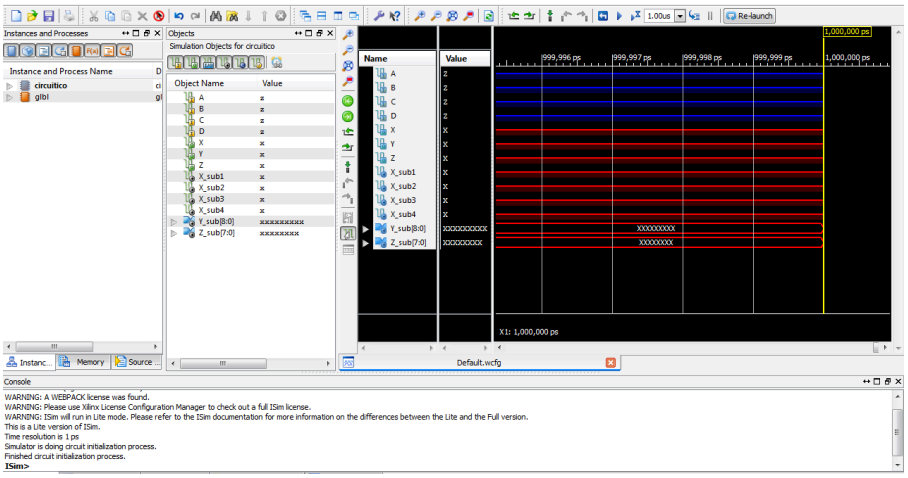
También es necesario chequear la sintaxis del comportamiento del circuito, con **"Behavioral Check Syntax"** antes de hacer la simulación como tal, de la misma manera, resulto estar correcta; si no lo hubiese estado, en la barra de errores y advertencias nombrada anteriormente, aparecerán los avisos necesarios para arreglar las faltas cometidas.



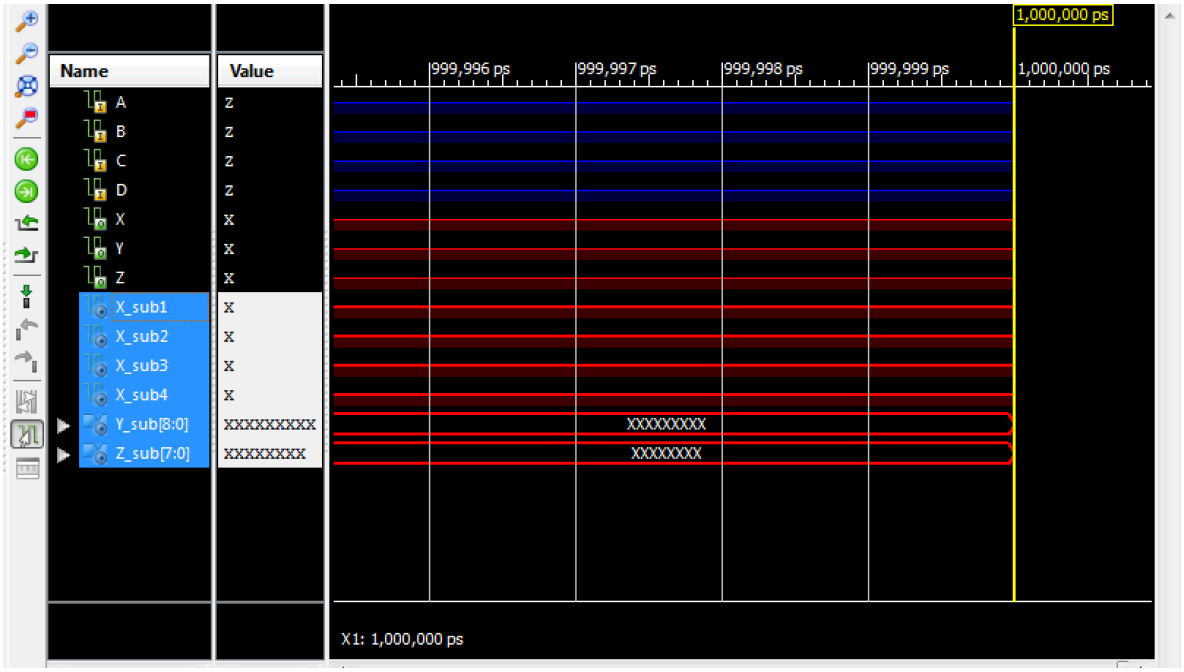
Tras cerciorarnos de la correcta escritura de la sintaxis de comportamiento de el circuito, se selecciona "**Simulate Behavioral Model**", para iniciar el proceso de simulación.



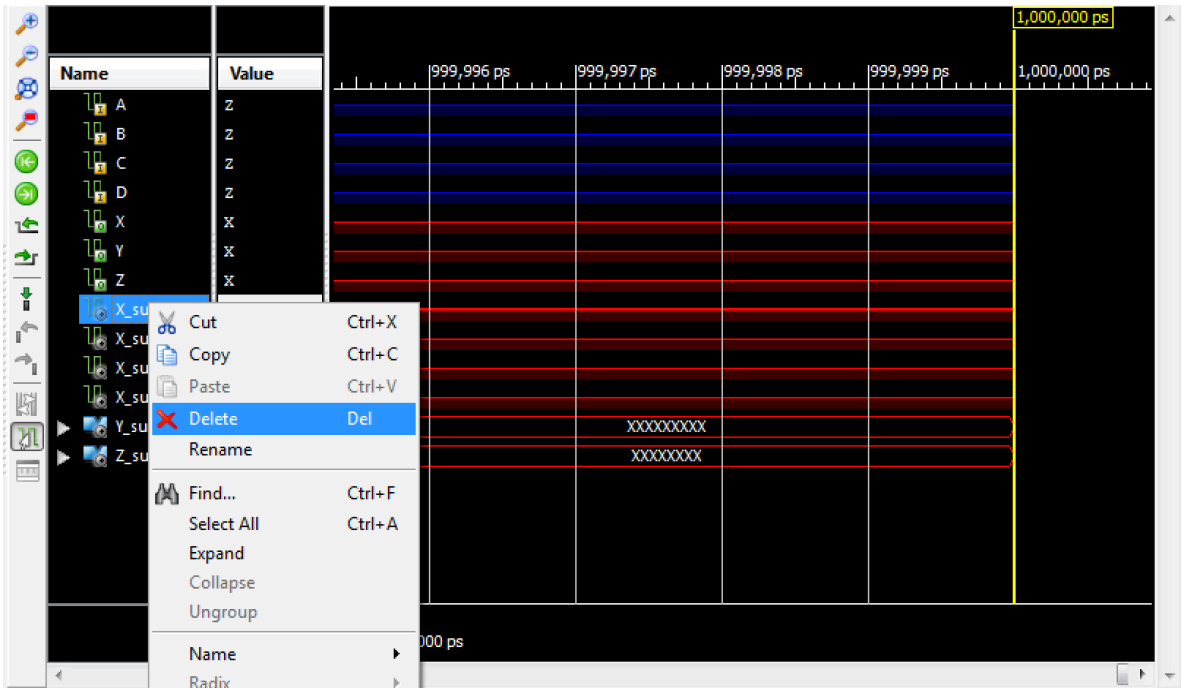
Se abre la herramienta **ISIM,** donde se probaran los estados de las salidas, por medio de la variación de las entradas.



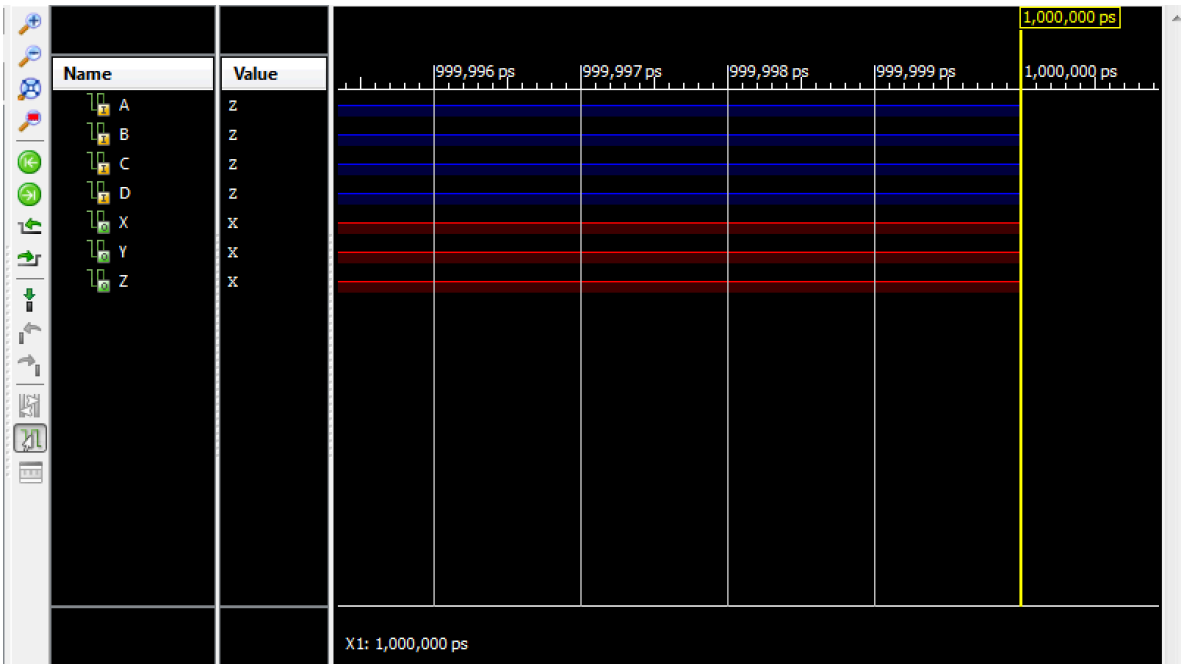
Seleccionamos los cables que creamos en el código (X_sub1,X_sub2.. etc)



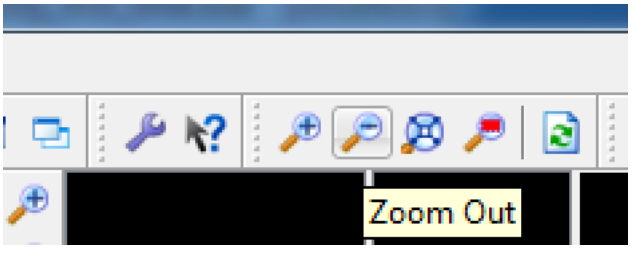
Y con click derecho, se despliegan algunas opciones, donde hemos de escoger "**Delete**", borramos estos cables, ya que no son ni salidas ni entradas, son los que conducen la información, solo necesitamos las salidas y las entradas para la simulación.



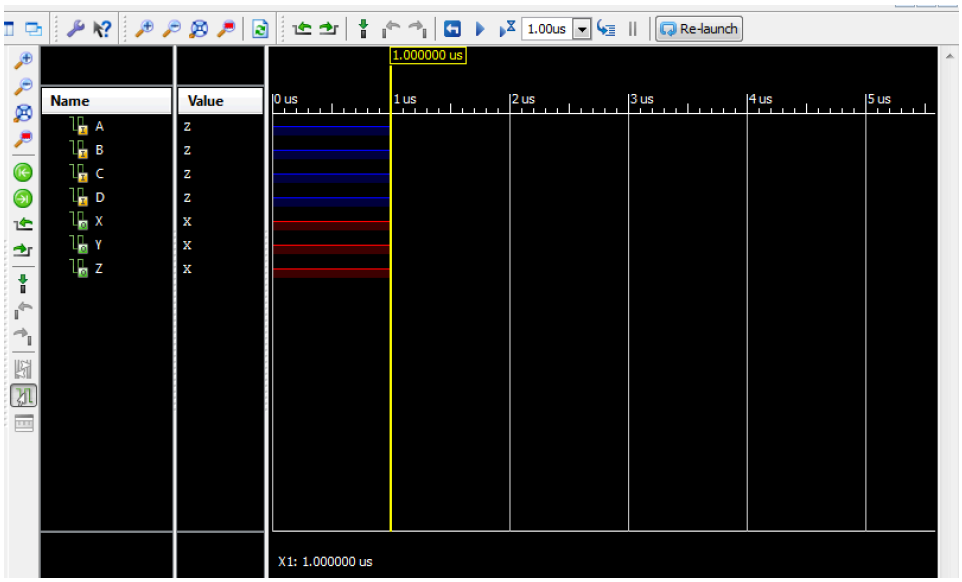
Y así queda nuestra ventana de simulación.

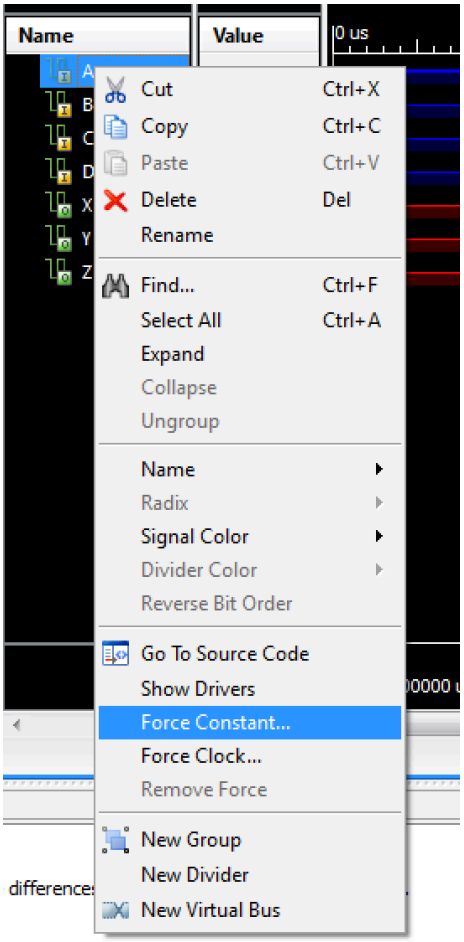


Los valores de tiempo predeterminados en la escala, puede que sean muy grandes y no nos permitan visualizar eficientemente lo que pasa en nuestro circuito después de introducir un par de valores para las entradas, así que es indicado contraer estos valores por comodidad; se busca en la parte superior de la ventana de visualización de estados, la lupa con el menos y damos click varias veces en ella, en "**Zoom Out**".

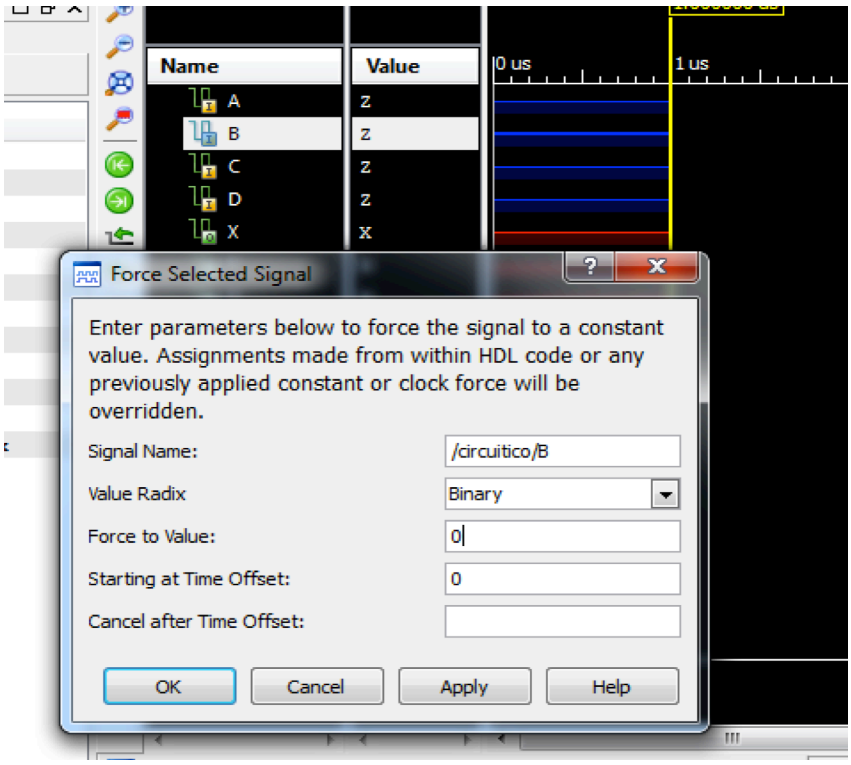


El resultado es este:



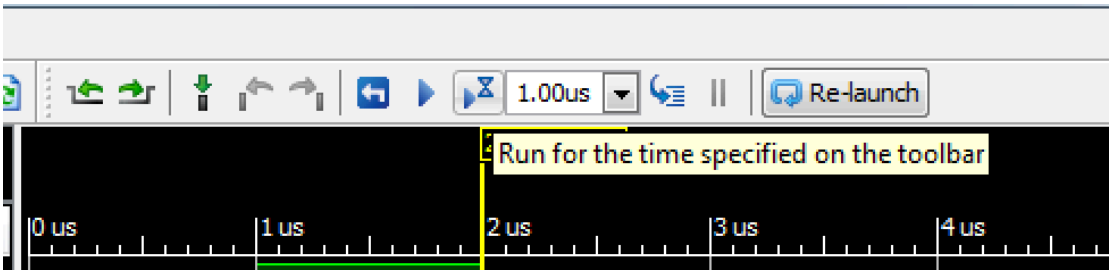


Ahora probaremos un valor, cualquier valor de entrada de la tabla de verdad, para verificar si la salida es la misma que deseamos o no; empezamos por la entrada **A,** le damos click derecho y buscamos el comando llamado "**Force Constant**", con el que forzaremos un valor específico para la entrada seleccionada. Se abre la siguiente ventana*:*

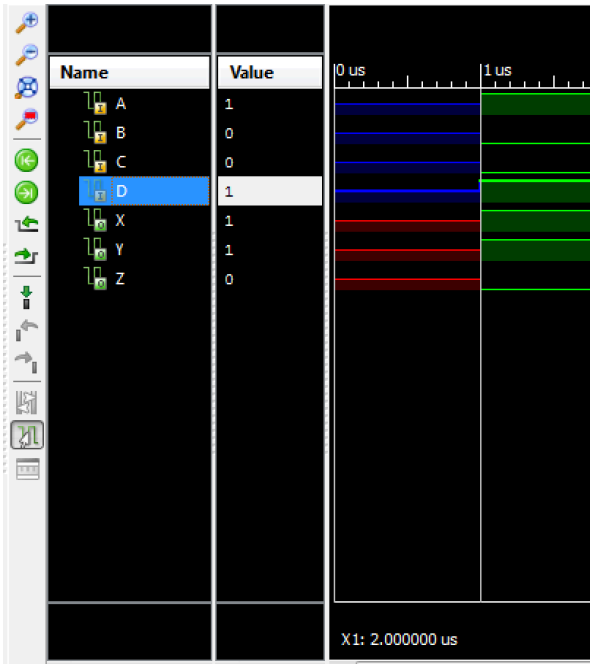


Ingresamos el valor deseado para **A,** en binario (0,1) y damos click en OK. Repetimos el procedimiento para el resto de entradas (**B,C,D**).

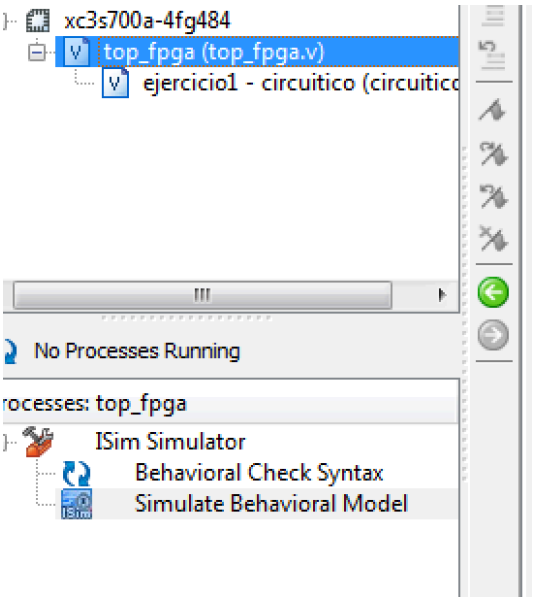
En la barra superior a la ventana de simulación, se acciona el botón "**Run for the time specified on the toolbar**", para iniciar la simulación.



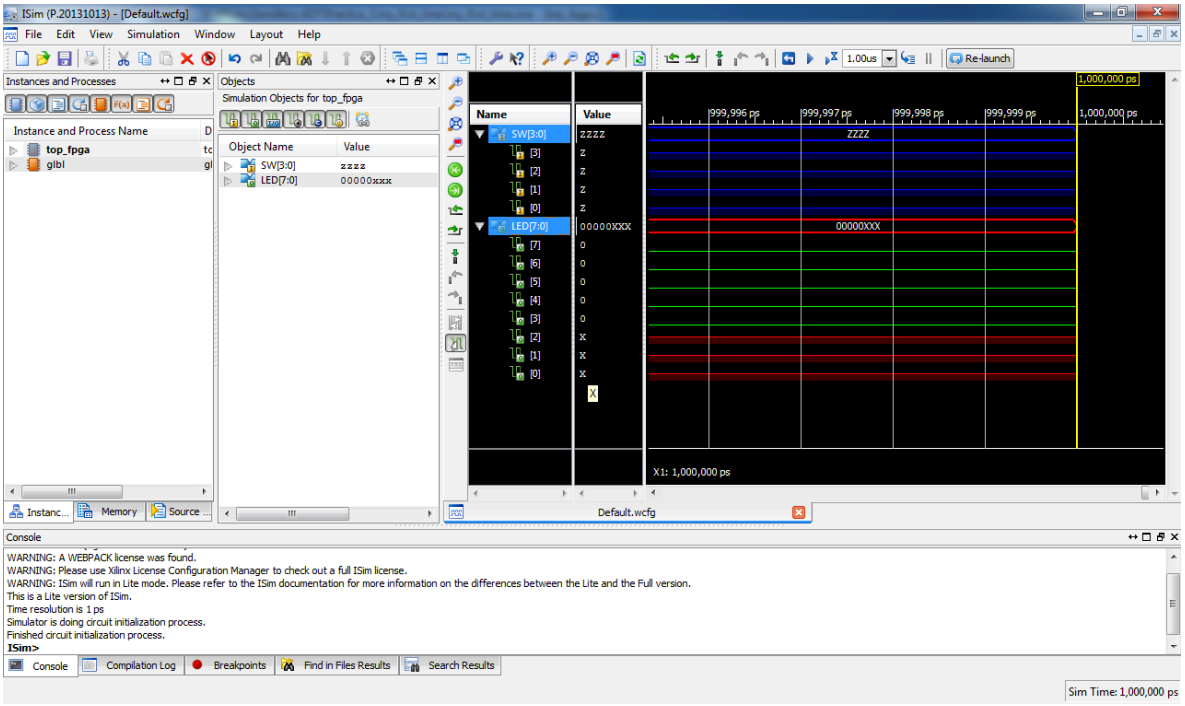
De esta manera podemos ver el resultado de la simulación, comprobando para el valor de entrada **1001**, la salida **110**; si comprobamos en la tabla de verdad, se puede ver la veracidad de esta relación. Se recomienda hacer el mismo procedimiento para diferentes valores de entrada y si es posible, para todas las combinaciones posibles.

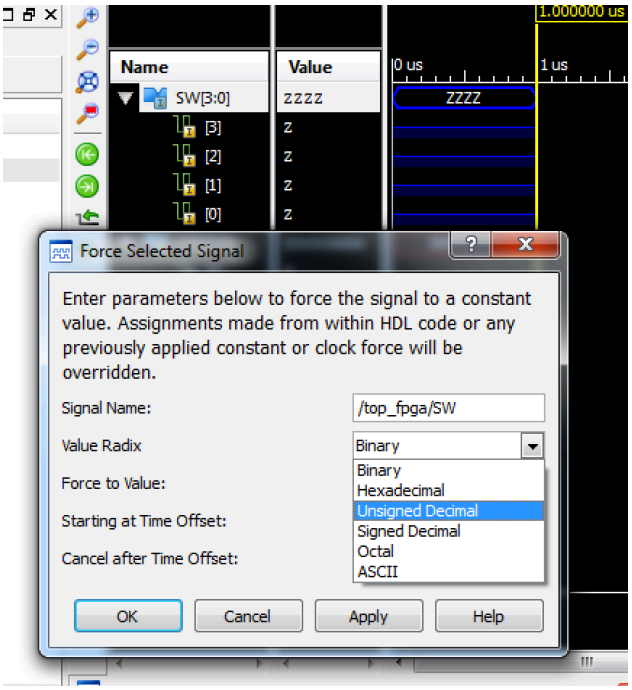


Habiendo hecho la simulación del módulo circuitico, se realiza el mismo proceso para el módulo principal, **top_fpga**, ya que con el correcto funcionamiento de este, notaremos que la relación entre las variables simbólicas y las reales aplicadas a la FPGA, funcionan de la manera deseada, por lo que seleccionamos top_fpga (en modo de simulación), chequeamos su la sintaxis de su modelo de comportamiento y luego seleccionamos **Simulate Behavioral Model.**



Se abre ISim de nuevo, pero ahora podremos notar que las entradas y salidas están designadas por **SW** y **LED,** allí se pueden visualizar también los demás leds designados, pero que no usaremos, se puede apreciar que su estado no se ve influenciado por las entradas SW.

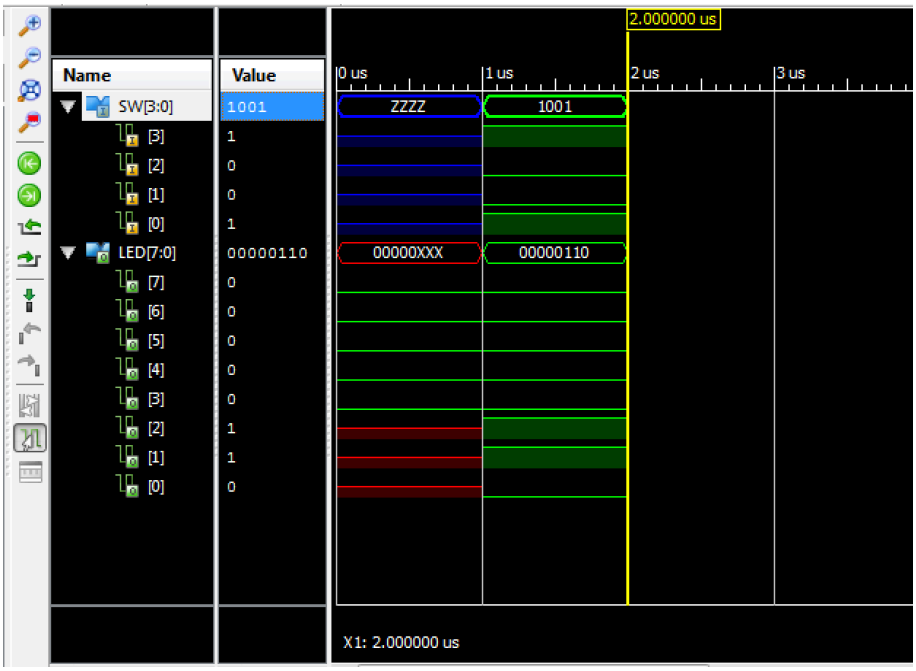


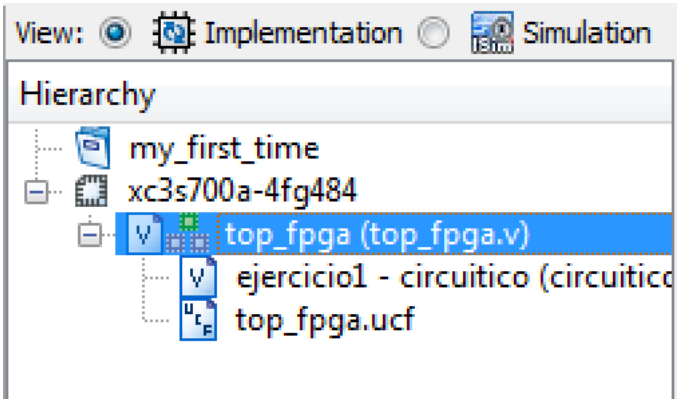


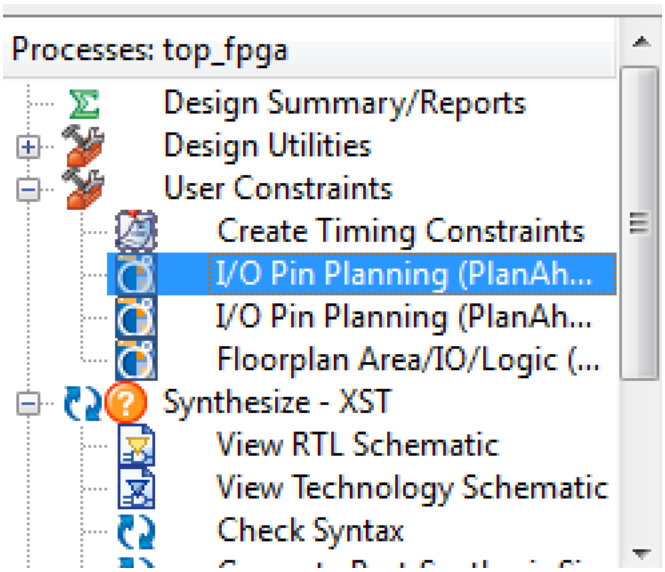
Ahora forzaremos el grupo en general de entradas, asignándoles un valor total; para ello se le da click derecho a **SW[3:0]** y se selecciona **Force Constant** al igual que como se hizo anteriormente, pero esta vez, cambiaremos el sistémica numérico en **Value Radix**, seleccionamos **Unsigned Decimal** y luego forzamos el valor, a un numero en base decimal, deseado, esto es para abreviar la asignación de un valor específico a cada entrada, el valor ingresado en decimal, será convertido a binario por el programa y esa entrada binaria, tendrá cierta respuesta en las salidas LED.

Aquí comprobamos de nuevo el funcionamiento del circuito y de acuerdo a la tabla de verdad, los valores de salida, corresponden con los de las entradas, todo bien.

El siguiente paso a realizar es la asignación de las entradas y salidas a la FPGA, de acuerdo al formato que esta maneja, al tipo de tecnología de cada elemento a usar y al nombre con que el programa reconoce que es que, dentro de los circuitos internos de la FPGA, así que de nuevo nos pasamos al modo de implementación.

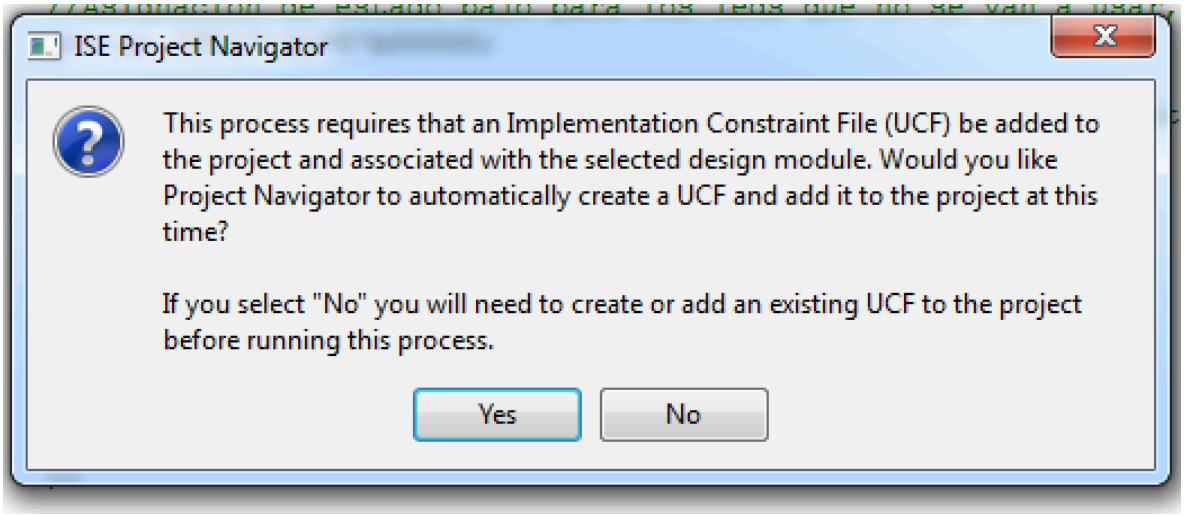




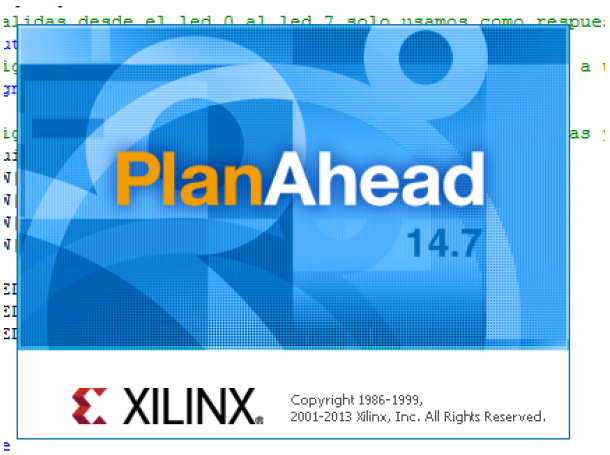


Buscamos entre la lista de procesos a realizar disponibles, el símbolo de la cajita que dice **User Constraints** y desplegamos sus opciones; seleccionamos **I/O Pin Planning**, al abrirlo tendremos una advertencia, donde nos dicen que es necesario un archivo UCF; este tipo de archivos es el que hace la implementación oficial de las entradas y salidas, le damos **YES**, para que se cree el archivo y después de ello, se abrirá la ventana de la herramienta **Plan Ahead.**

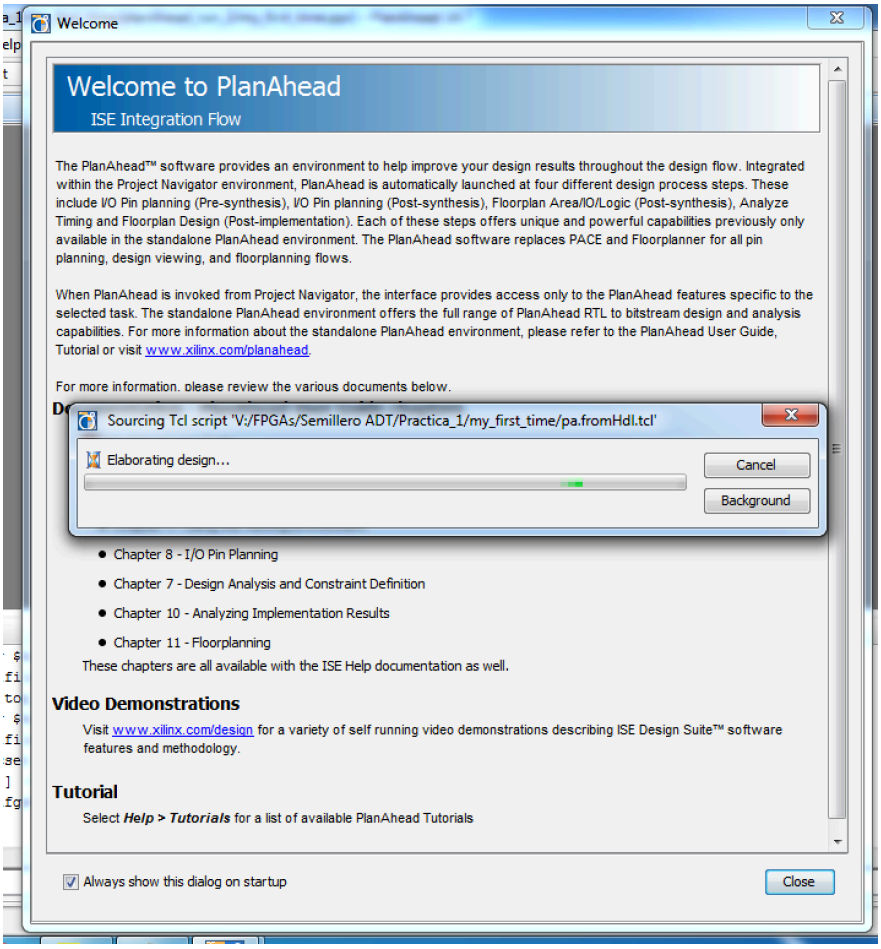
Dar click en **YES**



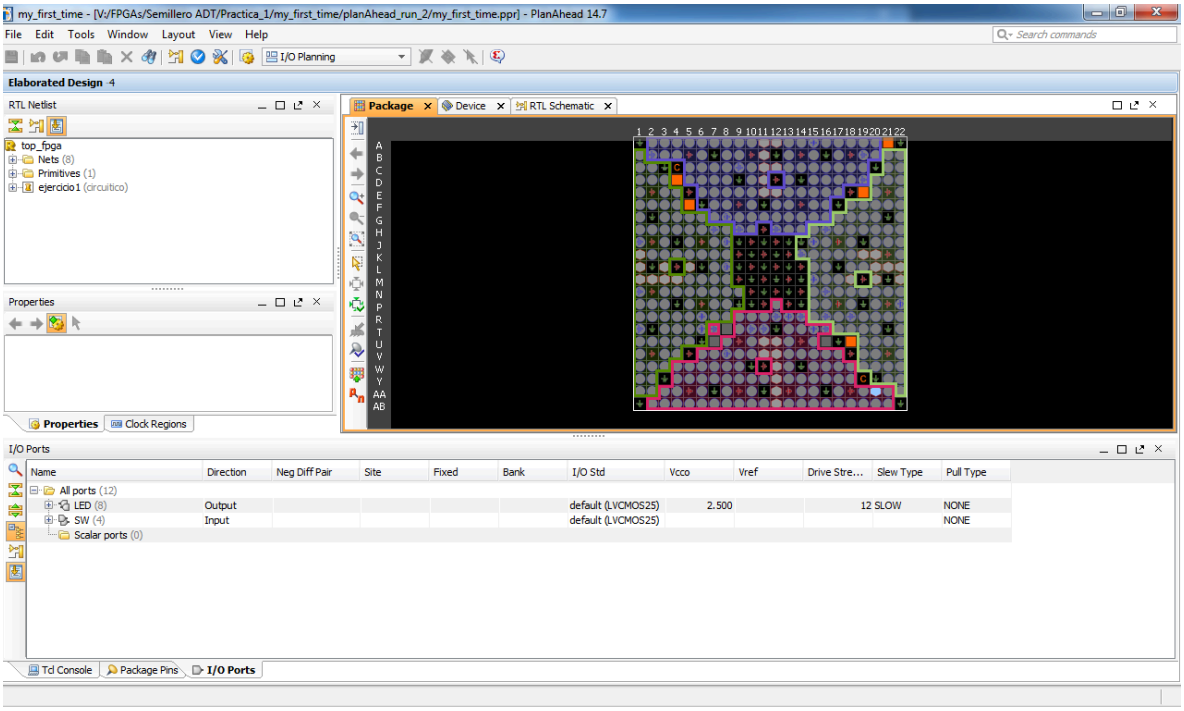
Se abre el programa:



El programa elabora diseños, carga algunos archivos, etc. Esto puede tardar algunos segundos, cuando acaben esos procesos, cerrar la ventana **Welcome**, para dar paso a la ventana principal donde configuraremos lo que necesitamos.



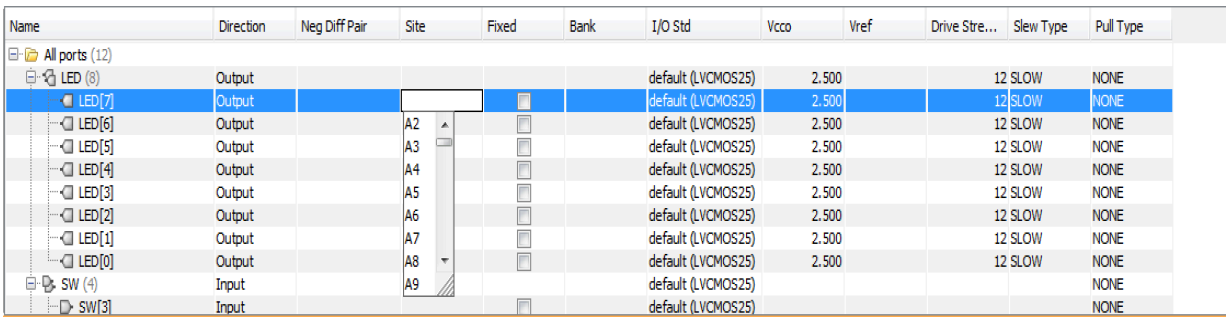
Esta es la ventana general del programa, la matriz con distintos colores y símbolos, representa la configuración input/output de la FPGA; nos centraremos en el panel inferior.



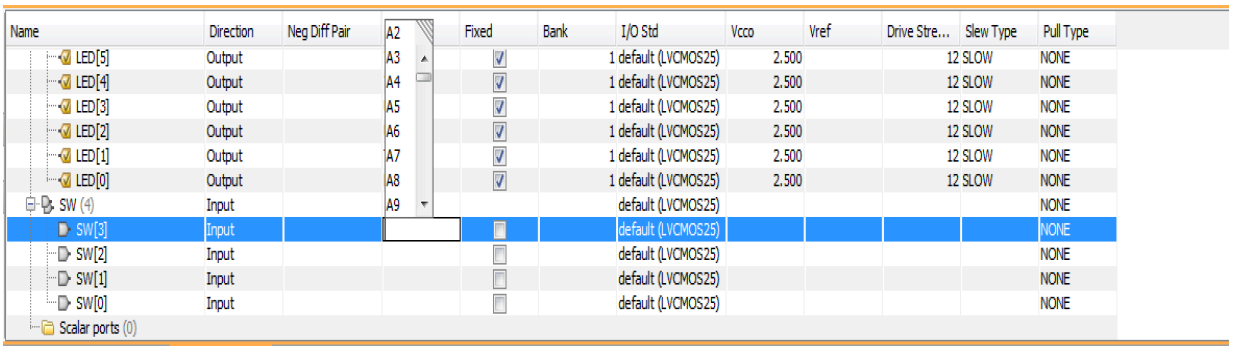
Seleccionamos en cada LED y cada SW su nomenclatura respectiva, en el diseño de la FPGA, esta nomenclatura se puede observar directamente desde la FPGA, pero no es tan recomendable, por lo que se consulta en el manual de usuario:

http://www.xilinx.com/support/documentation/user_guides/ug331.pdf

Se busca la asignación específica para cada Switch y cada Led, y en general para cualquier Input/Output/Inout (entrada,salida,entrada y salida).

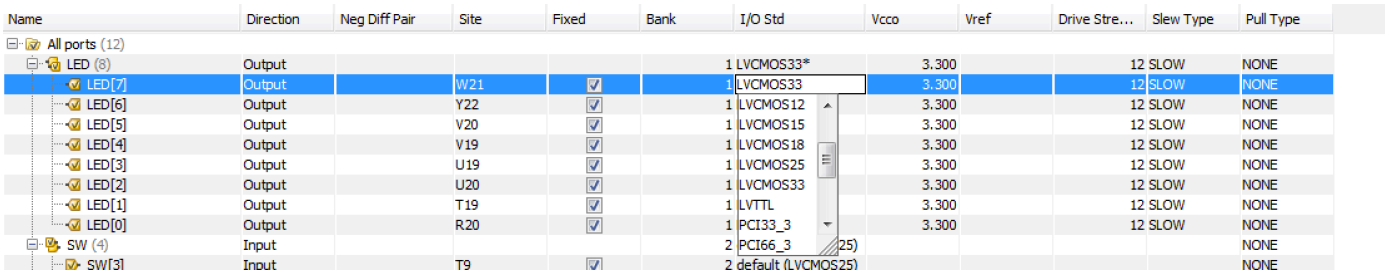


Tras asignar el nombre correspondiente a cada LED, se repite el procedimiento, ahora con los SW.



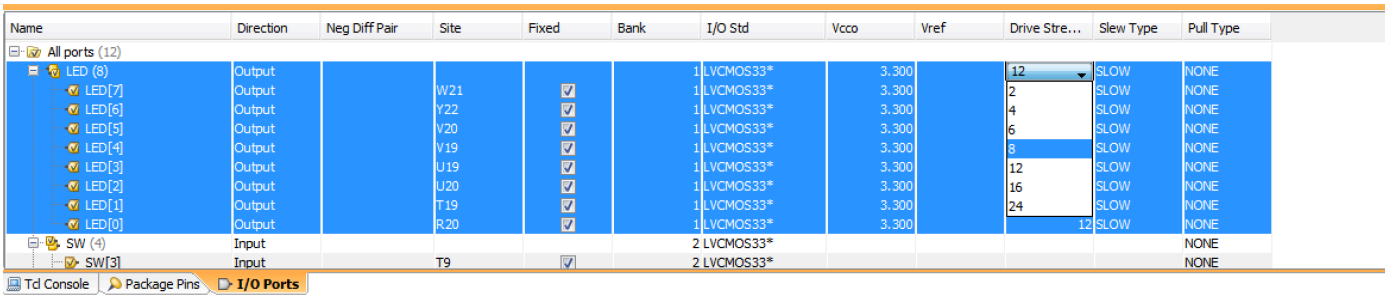
Name	Direction	Neg Diff Pair	A2	Fixed	Bank	I/O Std	Vcco	Vref	Drive Stre...	Slew Type	Pull Type
LED[5]	Output		A3	☑	1	default (LVCMOS25)	2.500		12	SLOW	NONE
LED[4]	Output		A4	☑	1	default (LVCMOS25)	2.500		12	SLOW	NONE
LED[3]	Output		A5	☑	1	default (LVCMOS25)	2.500		12	SLOW	NONE
LED[2]	Output		A6	☑	1	default (LVCMOS25)	2.500		12	SLOW	NONE
LED[1]	Output		A7	☑	1	default (LVCMOS25)	2.500		12	SLOW	NONE
LED[0]	Output		A8	☑	1	default (LVCMOS25)	2.500		12	SLOW	NONE
SW (4)	Input		A9			default (LVCMOS25)					NONE
SW[3]	Input			☐		default (LVCMOS25)					NONE
SW[2]	Input			☐		default (LVCMOS25)					NONE
SW[1]	Input			☐		default (LVCMOS25)					NONE
SW[0]	Input			☐		default (LVCMOS25)					NONE
Scalar ports (0)											

Teniendo ya los nombres de cada entrada y salida, se escoge el tipo de tecnología, de las entradas y las salidas, en este caso será la misma: **LVCMOS33*,** esta información también se encuentra en el manual de usuario.



Name	Direction	Neg Diff Pair	Site	Fixed	Bank	I/O Std	Vcco	Vref	Drive Stre...	Slew Type	Pull Type
All ports (12)											
LED (8)	Output				1	LVCMOS33*	3.300		12	SLOW	NONE
LED[7]	Output		W21	☑	1	LVCMOS33	3.300		12	SLOW	NONE
LED[6]	Output		Y22	☑	1	LVCMOS12	3.300		12	SLOW	NONE
LED[5]	Output		V20	☑	1	LVCMOS15	3.300		12	SLOW	NONE
LED[4]	Output		V19	☑	1	LVCMOS18	3.300		12	SLOW	NONE
LED[3]	Output		U19	☑	1	LVCMOS25	3.300		12	SLOW	NONE
LED[2]	Output		U20	☑	1	LVCMOS33	3.300		12	SLOW	NONE
LED[1]	Output		T19	☑	1	LVTTL	3.300		12	SLOW	NONE
LED[0]	Output		R20	☑	1	PCI33_3	3.300		12	SLOW	NONE
SW (4)	Input				2	PCI66_3					NONE
SW[3]	Input		T9	☑	2	default (LVCMOS25)					NONE

Se pueden seleccionar todas las entradas y salidas y escoger el tipo de tecnología y esta será aplicada a todos los elementos seleccionados.



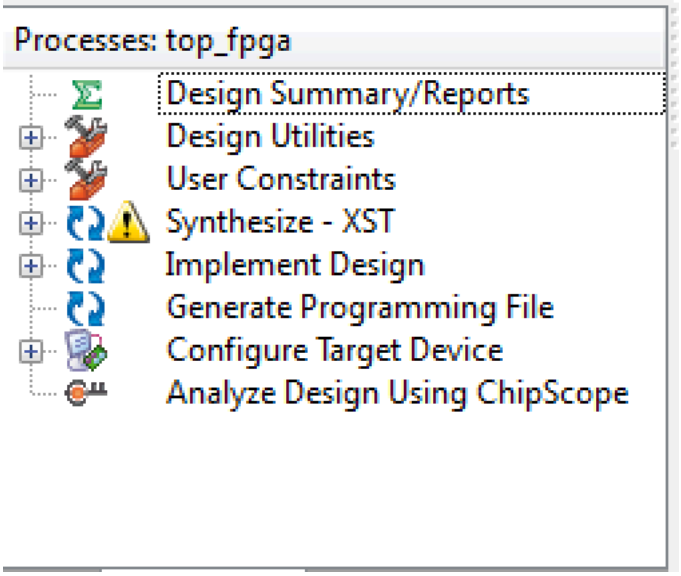
Name	Direction	Neg Diff Pair	Site	Fixed	Bank	I/O Std	Vcco	Vref	Drive Stre...	Slew Type	Pull Type
All ports (12)											
LED (8)	Output				1	LVCMOS33*	3.300		12	SLOW	NONE
LED[7]	Output		W21	☑	1	LVCMOS33*	3.300			SLOW	NONE
LED[6]	Output		Y22	☑	1	LVCMOS33*	3.300			SLOW	NONE
LED[5]	Output		V20	☑	1	LVCMOS33*	3.300			SLOW	NONE
LED[4]	Output		V19	☑	1	LVCMOS33*	3.300			SLOW	NONE
LED[3]	Output		U19	☑	1	LVCMOS33*	3.300			SLOW	NONE
LED[2]	Output		U20	☑	1	LVCMOS33*	3.300			SLOW	NONE
LED[1]	Output		T19	☑	1	LVCMOS33*	3.300			SLOW	NONE
LED[0]	Output		R20	☑	1	LVCMOS33*	3.300		12	SLOW	NONE
SW (4)	Input				2	LVCMOS33*					NONE
SW[3]	Input		T9	☑	2	LVCMOS33*					NONE

2 4 6 8 12 16 24

Tcl Console | Package Pins | I/O Ports

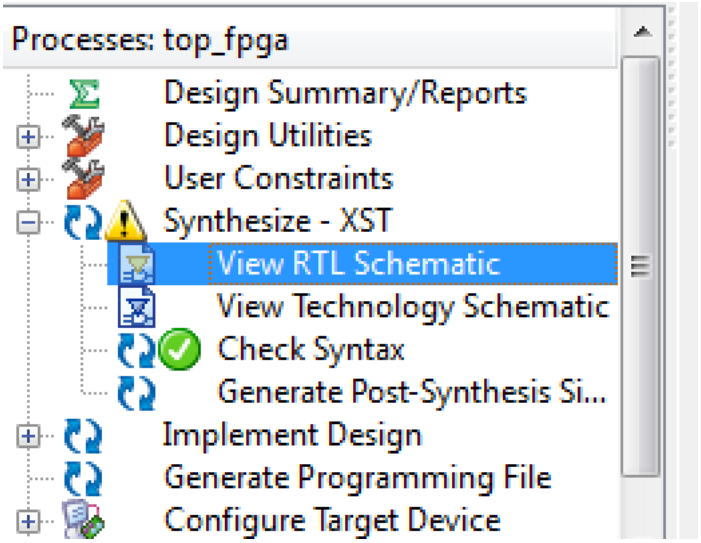
Este es el resultado; cerciorarse de que esté todo igual a lo mostrado en las imágenes, por prevención.

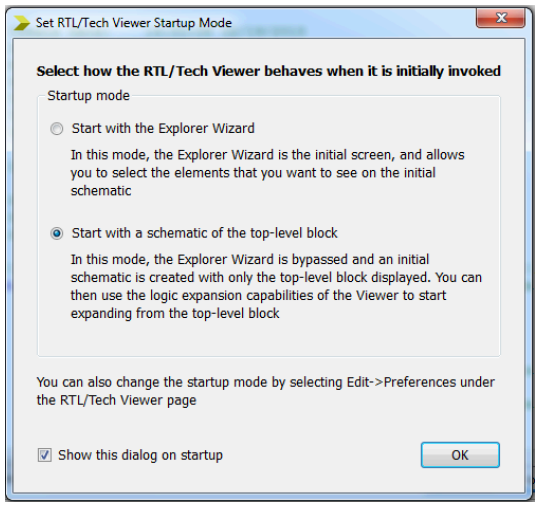
Name	Direction	Neg Diff Pair	Site	Fixed	Bank	I/O Std	Vcco	Vref	Drive Stre...	Slew Type	Pull Type
All ports (12)											
LED (8)	Output					1 LVCMOS33*	3.300		8*	SLOW	NONE
LED[7]	Output		W21	☑		1 LVCMOS33*	3.300		8*	SLOW	NONE
LED[6]	Output		Y22	☑		1 LVCMOS33*	3.300		8*	SLOW	NONE
LED[5]	Output		V20	☑		1 LVCMOS33*	3.300		8*	SLOW	NONE
LED[4]	Output		V19	☑		1 LVCMOS33*	3.300		8*	SLOW	NONE
LED[3]	Output		U19	☑		1 LVCMOS33*	3.300		8*	SLOW	NONE
LED[2]	Output		U20	☑		1 LVCMOS33*	3.300		8*	SLOW	NONE
LED[1]	Output		T19	☑		1 LVCMOS33*	3.300		8*	SLOW	NONE
LED[0]	Output		R20	☑		1 LVCMOS33*	3.300		8*	SLOW	NONE
SW (4)	Input					2 LVCMOS33*					NONE
SW[3]	Input		T9	☑		2 LVCMOS33*					NONE
SW[2]	Input		U8	☑		2 LVCMOS33*					NONE
SW[1]	Input		U10	☑		2 LVCMOS33*					NONE
SW[0]	Input		V8	☑		2 LVCMOS33*					NONE
Scalar ports (0)											



Ya se tiene lo necesario para hacer la implementación en la FPGA, entonces conectamos la FPGA, asumiendo que los drivers ya están en orden y no hay problema para reconocerla al conectarla. Ya estuvimos en **User Constraints,** asignando las entradas y las salidas, ahora tan solo tenemos que seguir la lista, hasta terminar en **Configure Target Device.**

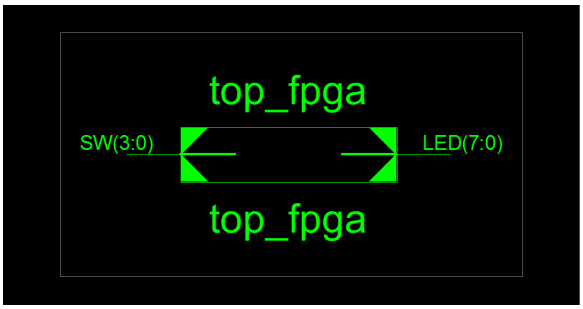
Seleccionamos **Synthesize-XST** y abrimos **View RTL Schematic**, en esta ocasión no abriremos View Technology Schematic, ya que para el ejercicio solo nos interesa ver el arreglo de compuertas generado por el programa.



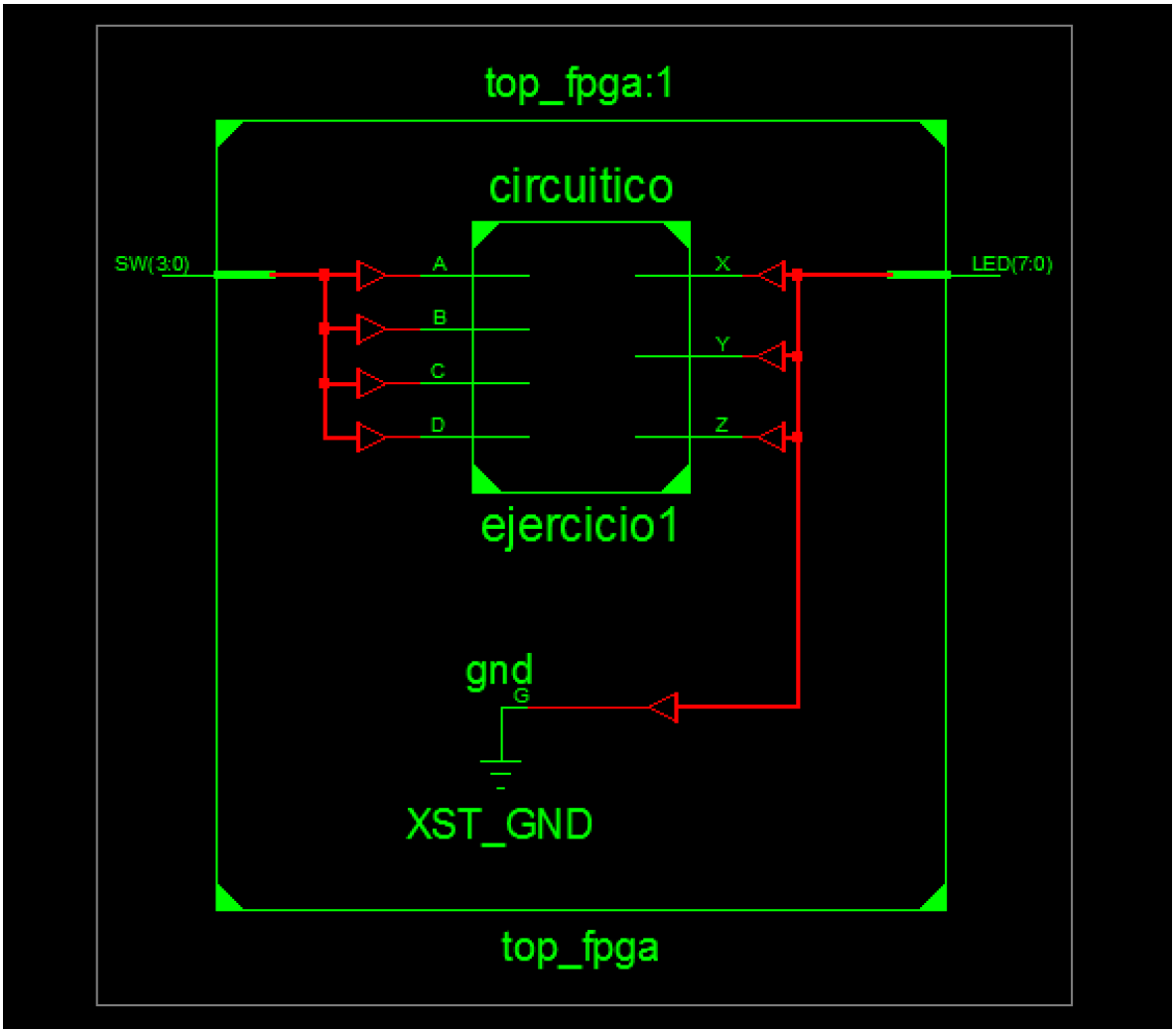


Le damos OK a las opciones predeterminadas que nos ofrece la ventana

Se abre en nuestra pantalla en siguiente esquema, que representa superficialmente el arreglo circuital que el programa genero a partir de la programación que

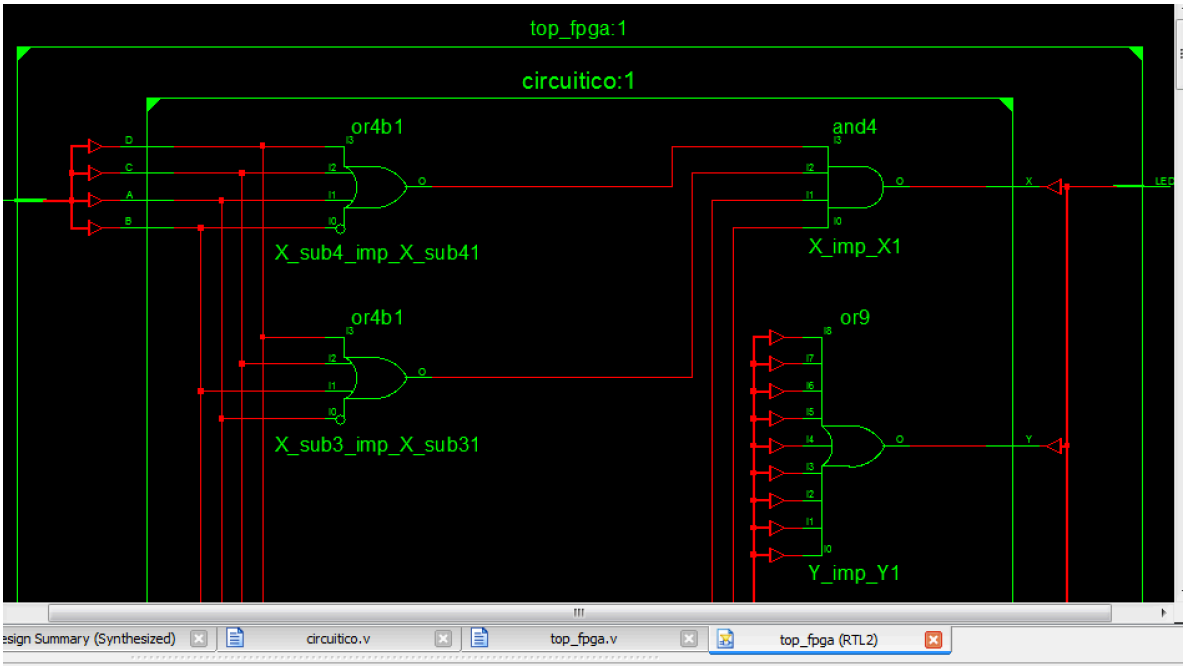


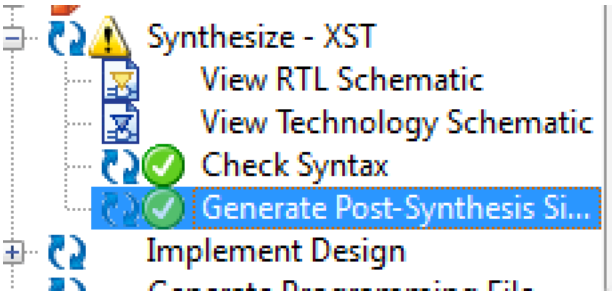
Al dar doble click sobre el diagrama, miraremos más adentro la configuración del circuito, buscamos la lupa y seleccionamos el icono mostrado en la imagen de la izquierda, para hacer cierto zoom y ver el diagrama completo.



Si volvemos a dar click en el diagrama, iremos más adentro aun y veremos las compuertas tal cual las programamos en Verilog, también sirve para comprobar algún diseño de circuito en esquema, para saber si corresponde al diseño, y a la descripción de hardware ingresada.

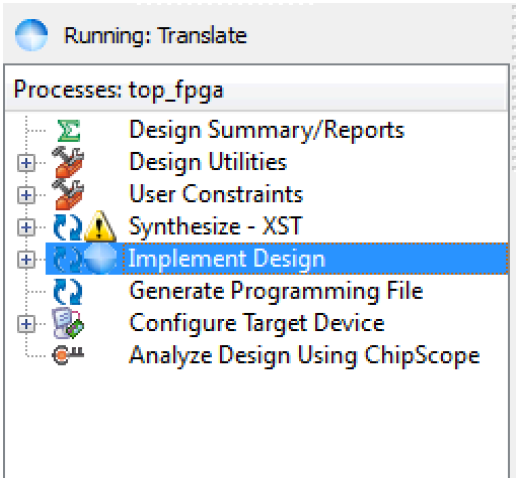
Así es como luce lo más adentro a lo que podemos llegar en nuestro circuito; teniendo una versión paga de los programas, podríamos ver también los arreglos de transistores de las compuertas y demás.

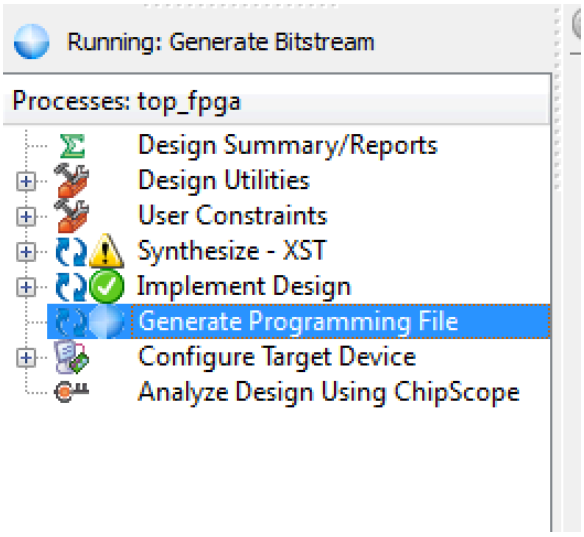




Luego de mirar el diseño esquemático del circuito, seguimos con los pasos para la implementación, damos click en **Check Syntax** y si está todo en orden, proseguimos con **Generate Post-Synthesis**

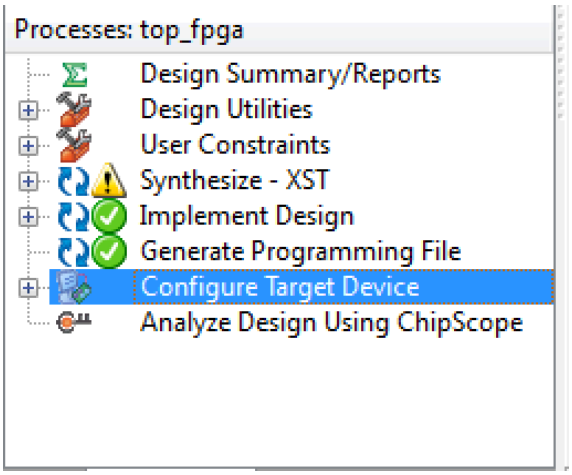
Ahora pasamos a la implementación del diseño, si, dando click en **Implement Design.**



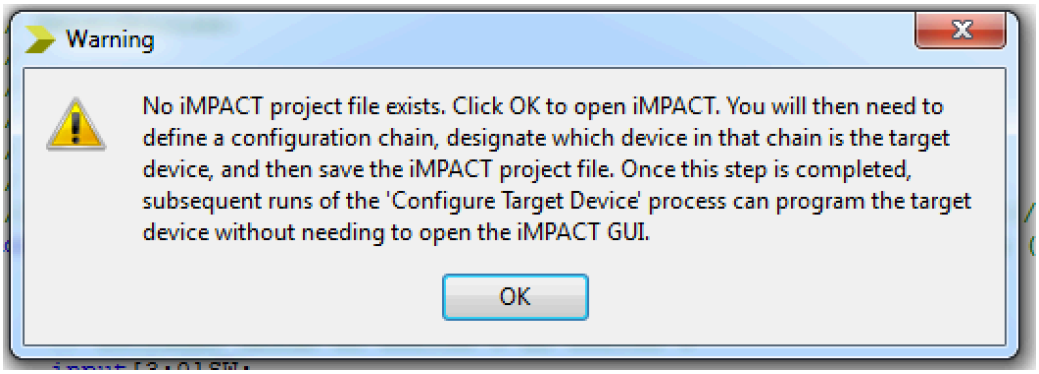


Todo en orden con la implementación del diseño, proceder a dar click en **Generate Programming File**

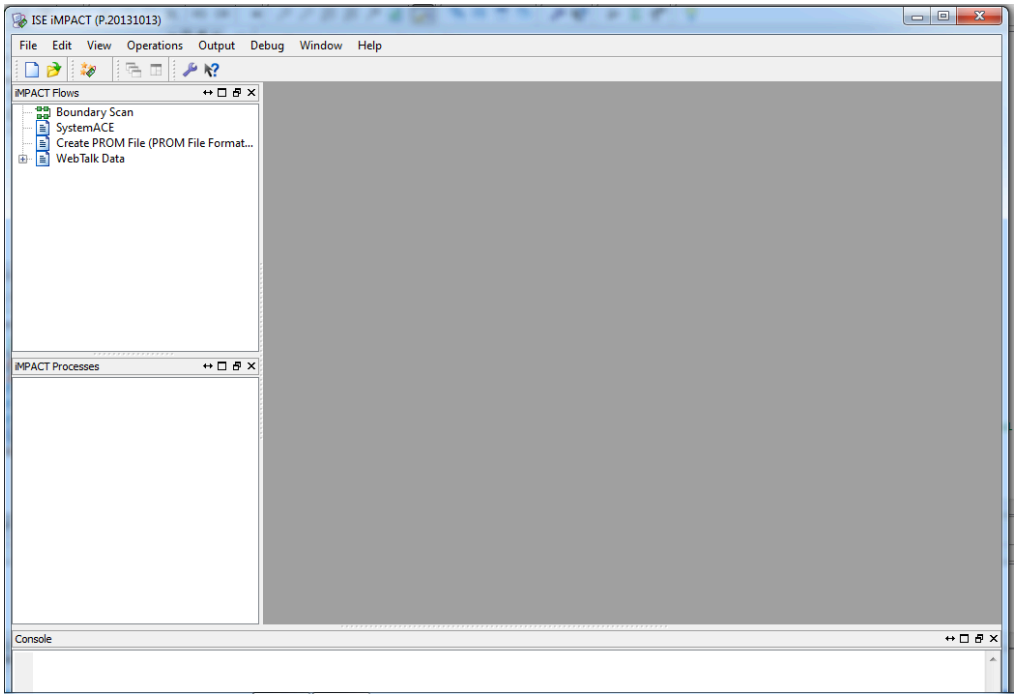
Hemos llegado al ultimo paso en la lista de procesos, **Configure Target Device,** le damos click.



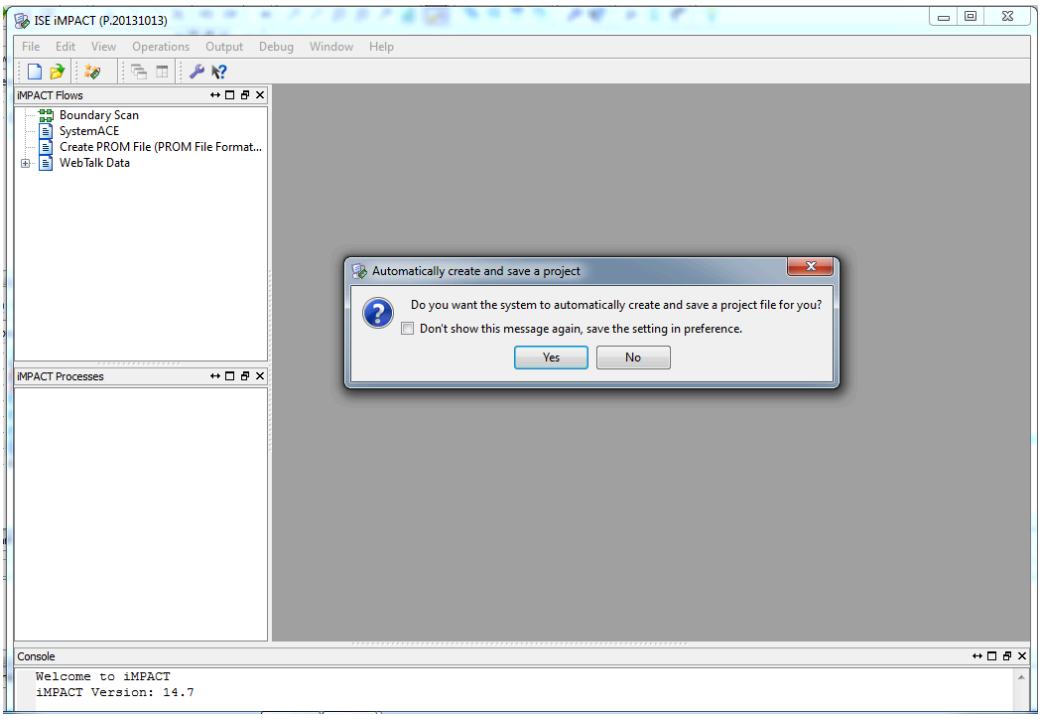
La advertencia nos dice que no hay un archivo iMPACT, pero preisonando OK, se abrirá la ventana que nos dará las opciones necesarias para subir el archivo a la FPGA, se generará el archivo iMPACT.



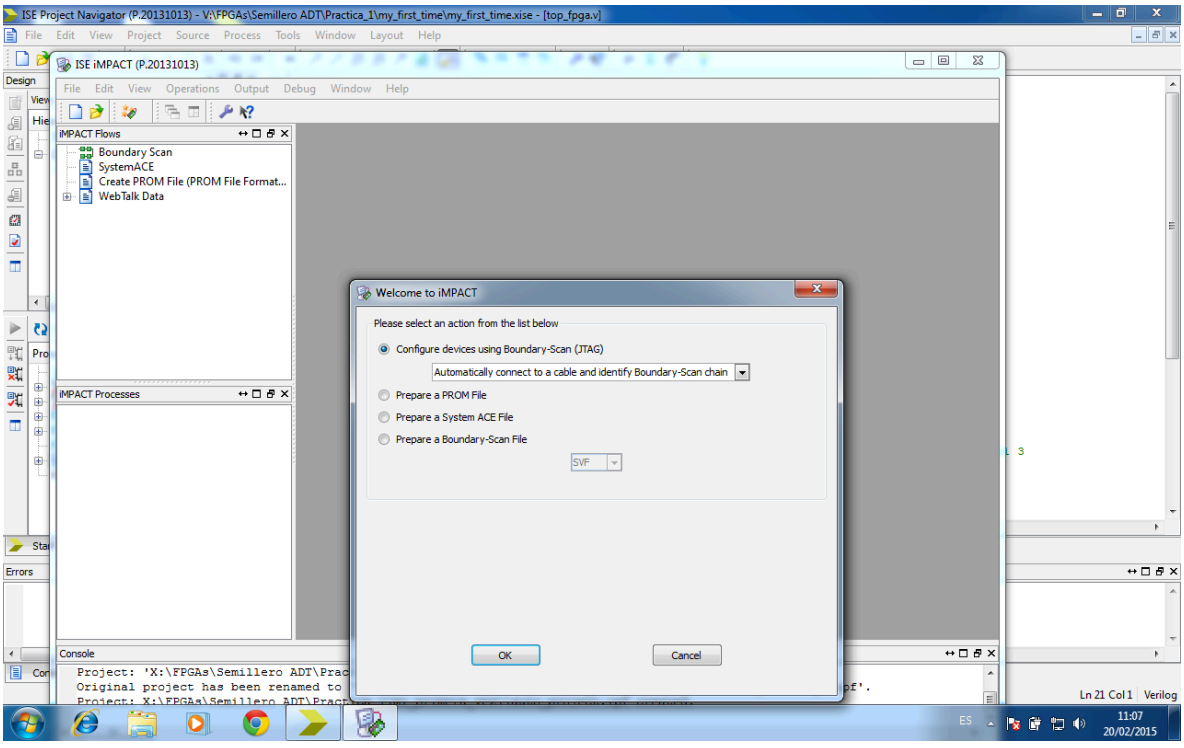
Esta es otra de las tantas herramientas que Xilinx ofrece, pero esta es determinante, ya que es la que da el golpe final, su nombre es **ISE iMPACT**, y así es como luce.



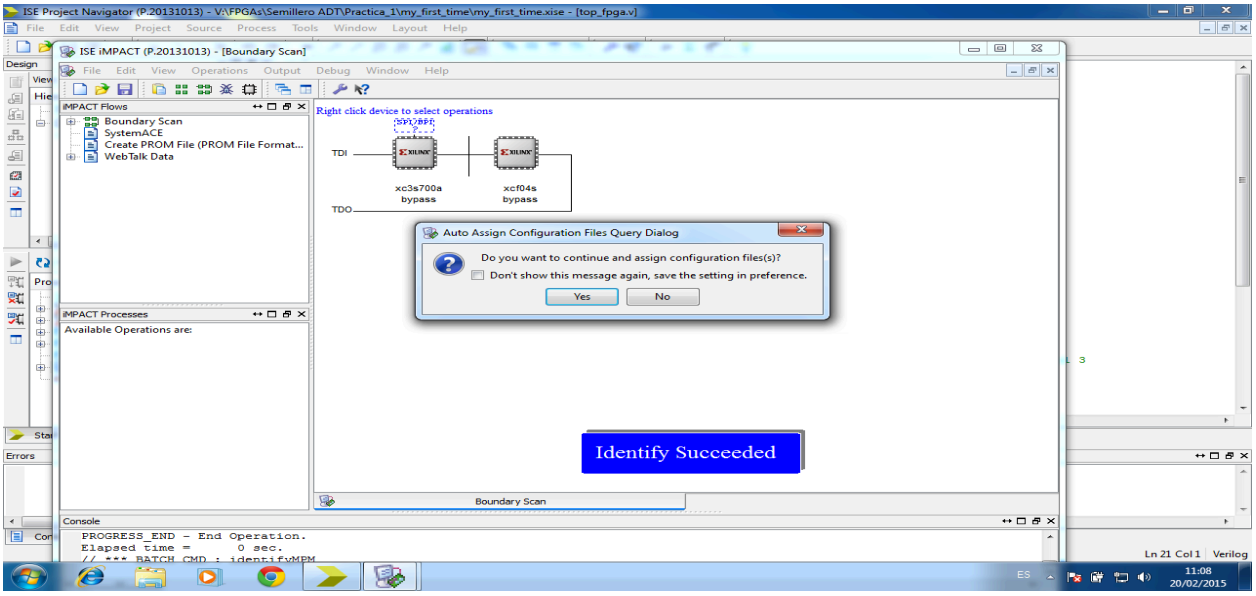
Escogemos **YES** a la ventana emergente, para crear y guardar proyecto automáticamente.



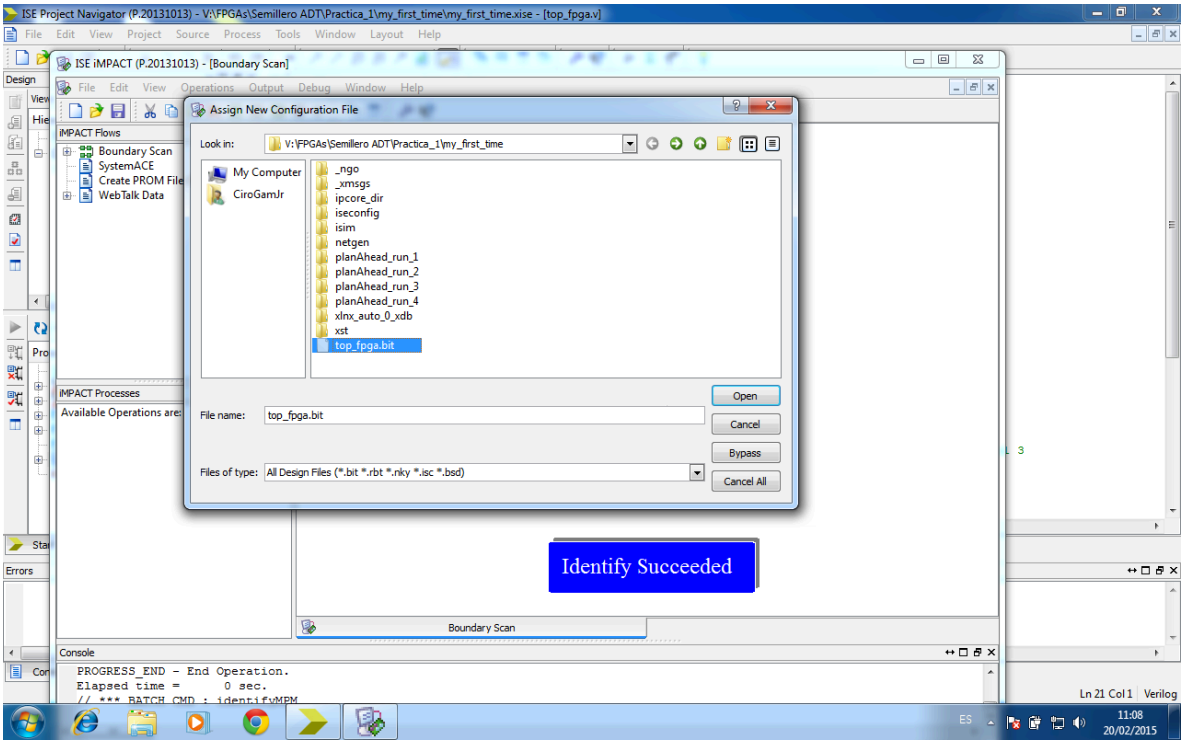
Seleccionamos OK para configurar con los parámetros predeterminados.



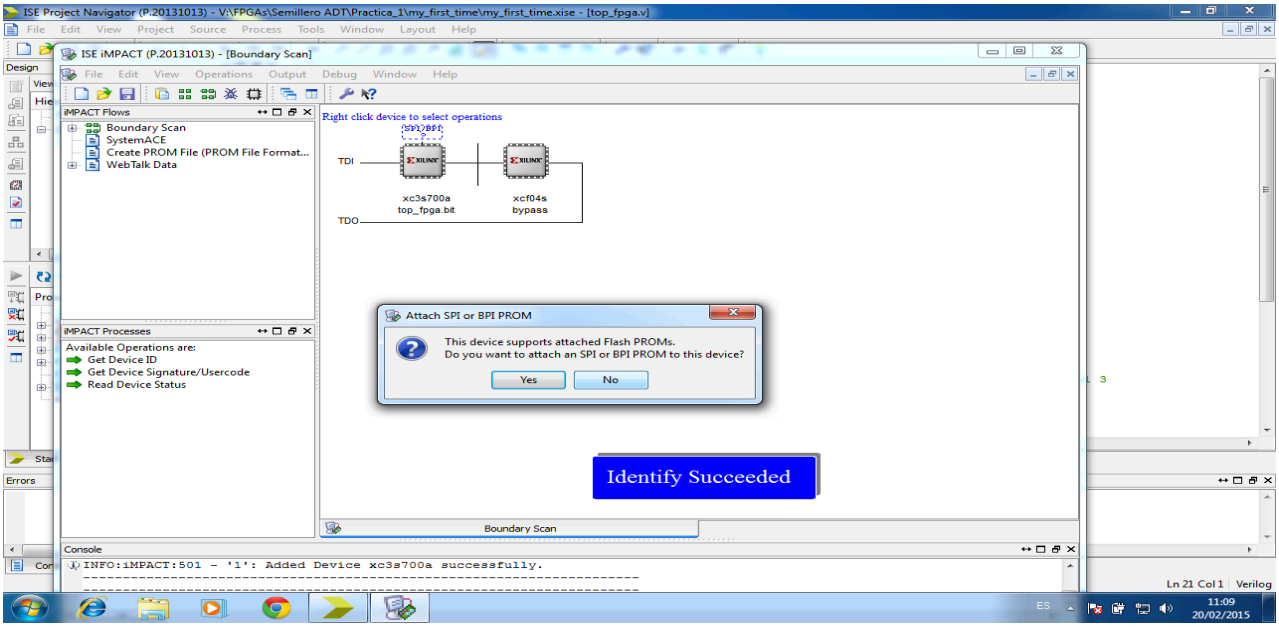
De esta manera salen un par de cuadritos, que representan el arreglo circuital y la memoria del circuito. El **Identify Succeded,** significa que ya fue reconocida la FPGA y las configuraciones realizadas. Clicl en YES.



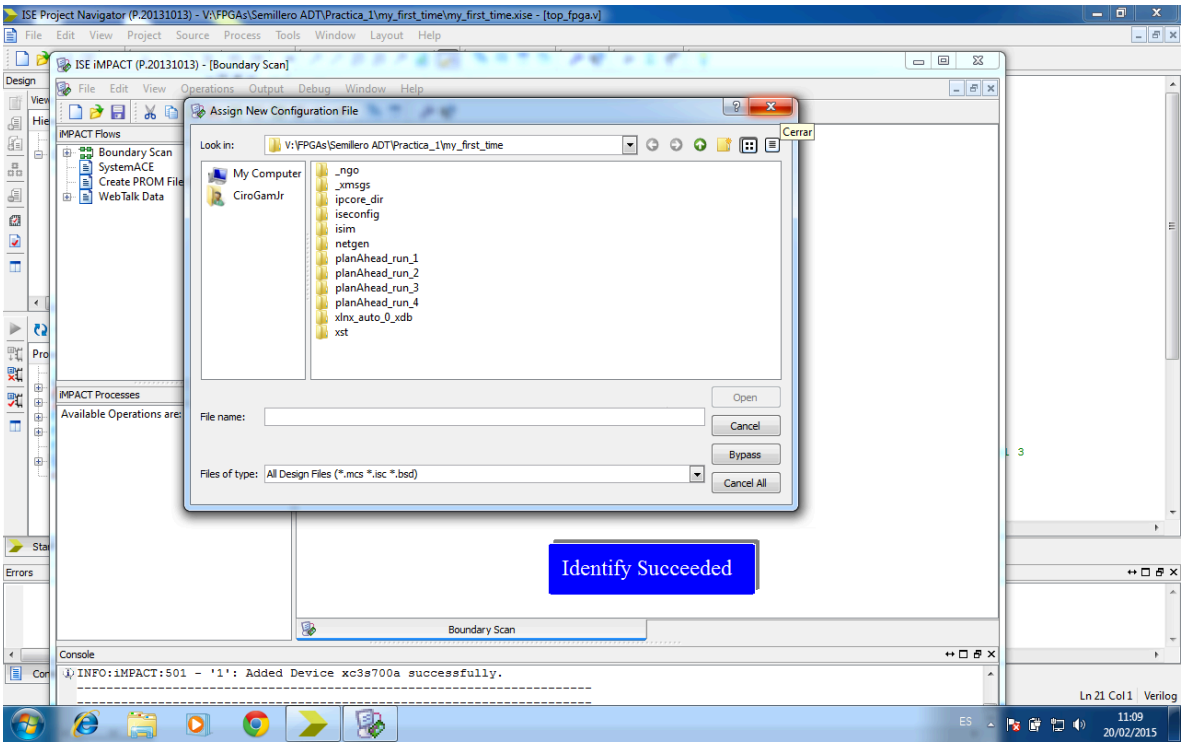
Debemos añadir el archivo.bit, que es la información que queremos que la FPGA procese, en un arreglo de bits, es como hablarle a la FPGA en su lenguaje, en binario. Seleccionar **top_fpga.bit.**



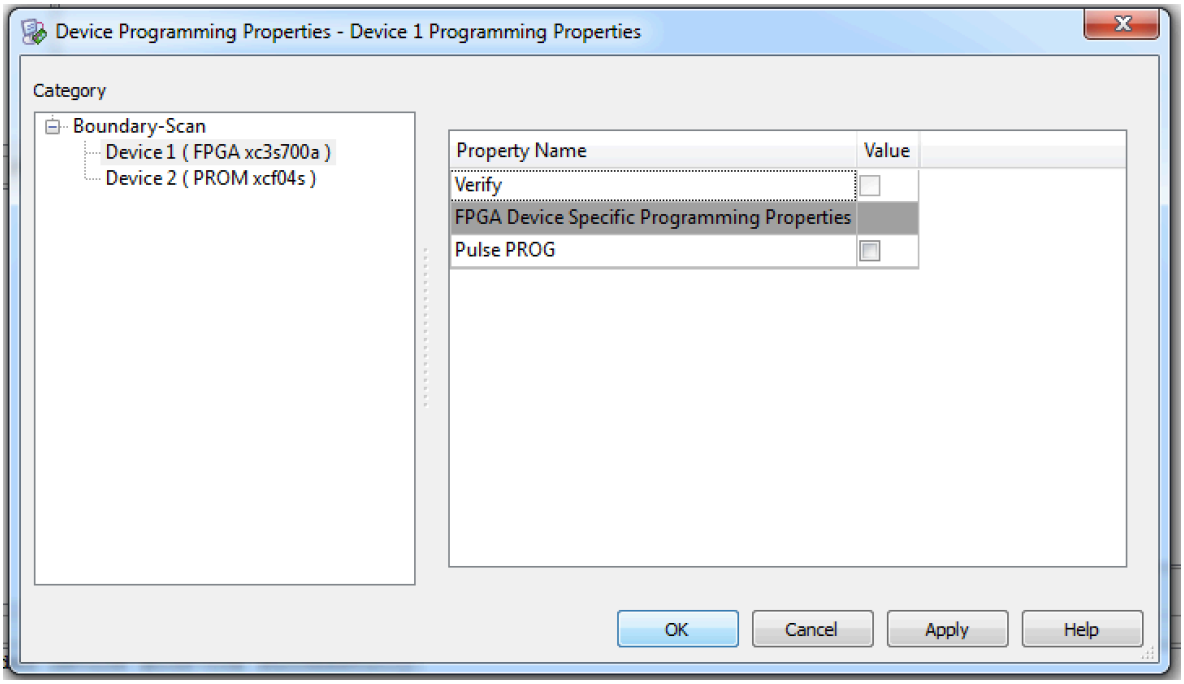
Después de ello se nos presentaran algunas opciones sobre añadir memoria al circuito, no lo haremos esta vez, así que cerraremos las ventanas emergentes que salen a continuación.



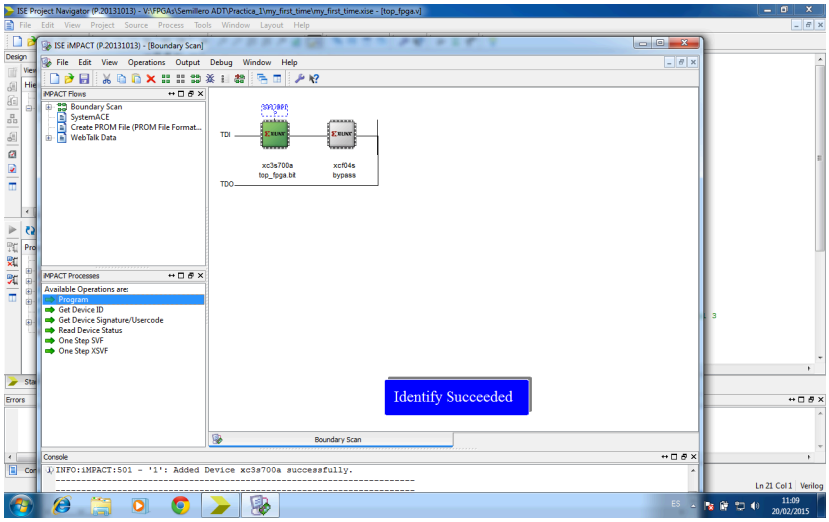
Cerrar



A esta ventana, le damos ok, para configurar de acuerdo a todos los parámetros que hemos tenido en cuenta durante todo el proceso.



A esto llegamos, el primer cuadrito esta en verde, y el verde es positivo como hemos podido apreciar a lo largo de los pasos anteriores; el segundo cuadro esta en gris, este corresponde al de la memoria, no se la asignamos, esto significa que si apagamos la FPGA después de subir el programa, esta no recordara la configuración que le dimos.



Por ultimo, seleccionamos **Program**, esperamos un momento y si sale en la parte inferior, el cuadro con el aviso: **Program Succeded**, felicitaciones, no hay errores en el proceso. Lo siguiente, es verificar la tabla de verdad mediante los estados de los interruptores y sus respectivas salidas con los led, si son correspondientes, hemos de alegrarnos por lo saludable de nuestra FPGA, de nuestra lógica y del conocimiento adquirido para la correcta manipulación de la misma.