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







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/e n/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:

Α	В	С	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	ABCD	ABCD
1	1	1	$A + B + C + \overline{D}$	ABCD	ABCD
1	1	0	$A + B + \overline{C} + D$	ABCD	ĀBCD
1	1	0	$A + B + \overline{C} + \overline{D}$	ABCD	ABCD
0	0	1	$A + \overline{B} + C + D$	ĀBĒD	ABCD
0	0	1	$A + \overline{B} + C + \overline{D}$	ĀBĒD	ABCD
0	0	0	$A + \overline{B} + \overline{C} + D$	ĀBCD	ĀBCD
1	0	0	$A + \overline{B} + \overline{C} + \overline{D}$	ĀBCD	ĀBCD
0	1	0	$\overline{A} + B + C + D$	ABCD	ABCD
1	1	0	$\overline{A} + B + C + \overline{D}$	ABCD	ABCD
1	1	1	$\overline{A} + B + \overline{C} + D$	ABCD	ABCD
1	1	1	$\overline{A} + B + \overline{C} + \overline{D}$	ABCD	ABCD
1	0	1	$\overline{A} + \overline{B} + C + D$	ABCD	ABCD
1	1	0	$\overline{A} + \overline{B} + C + \overline{D}$	ABCD	ABĒD
1	0	1	$\overline{A} + \overline{B} + \overline{C} + D$	ABCD	ABCD
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.

 $X=(A+B^{+}C+D)(A+B^{-}+C+D^{-})(A+B^{-}+C^{-}+D)(A^{-}+B+C+D)$ $Y=A^{-}B^{-}C^{-}D+A^{-}B^{-}CD^{-}+A^{-}B^{-}C^{-}D+A^{-}B^$

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 4;
input 5;
input 0;
//Salidas
output X;
output Y;
output Z;
//Cables; estos son uniones fisicas
wire X_sub1;
wire X_sub2;
wire X_sub1;
wire X_sub2;
wire X_sub3;
wire X_sub4;
wire [8:01Y_sub;
//Los X_subn son términos de la ecuación
assign X_Sub1=A[(-B)[C](-D);
assign X_Sub1=A[(-B)[C](-D);
assign X_Sub3=(-A)[B[C][)
assign X_Sub3=(-A)[B[C][)
assign X_Sub3=(-A)[B[C][)
assign X_Sub3=(-A)[B[C][)
assign Y_Sub4=(-B)[C](-D);
assign Y_Sub4=(-B)[C](-D);
assign Y_Sub4=(-B)[C](-D);
assign Y_Sub3=(-A)[B[C][)
assign Y_Sub3=(-A)[B[C][]
assign Y_Sub3=(-A)[B[C][]
assign Y_Sub3=(-A)[B[C][]
assign Y_Sub4=(-B) & (-C) & D;
assign Y_Sub1=(-A) & (-B) & (-C) & D;
assign Y_Sub1=(-A) & B & (-C) & (-D);
assign Y_Sub1=(-A) & B & (-C) & (-D);
assign Y_Sub1=(-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]=

🔄 Guardar como X 💭 🗢 🚺 🕨 CiroGamJr 🕨 ▼ ⁴→ Buscar CiroGamJr Q Organizar 🔻 Nueva carpeta -----0 🔆 Favoritos 0 🗼 Descargas 📴 Dropbox (Mac) Búsquedas Escritorio Contacto Descargas Applicati Data 🧱 Escritorio iCloud Drive (Ma 1 -Sitios recientes FPGA Juegos guardado: Mis Mi músio Sparta documento Bibliotecas Documentos 🔤 Imágenes 🎝 Música circuitico.v Mis víde Vínculos Vídeos Nombre: circuitico.v Tipo: Todos los archivos (*.*) Codificación: ANSI • Guardar Cancelar Ocultar carpetas

-<u>And</u>, <u>or</u> 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".

Propiedades: circu	uitico.v.txt
General Seguridad	Detalles Versiones anteriores
	circuitico.v.bd
Tipo de archivo:	Documento de texto (.txt)
Se abre con:	Bloc de notas
Ubicación:	C:\Users\CiroGamJr\Desktop
Tamaño:	1.24 KB (1.271 bytes)
Tamaño en disco:	4,00 KB (4.096 bytes)
Creado:	Hoy, 19 de febrero de 2015, Hace 1 minuto
Modificado:	Hoy, 19 de febrero de 2015, Hace 1 minuto
Último acceso:	Hoy, 19 de febrero de 2015, Hace 1 minuto
Atributos: 🔲 Só	ólo lectura 🔲 Oculto
	Avanzados
	Aceptar Cancelar Aplicar

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, <u>no es necesario usar bloc de notas todo</u> <u>el tiempo.</u>



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:

New Project	Wizard	
Create New Pro	ject nand type.	
Name: Location: Working Directory: Description:	W:PFGAsSemilero ADTPractica_11my_first_time V:PFGAsSemilero ADTPractica_11my_first_time V:PFGAsSemilero ADTPractica_11my_first_time	
Select the type of to Top-level source typ HDL More Info	p-level source for the project e:	Next Cancel

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.

roject Settings		
pecify device and project properties. elect the device and design flow for the p	roject	
Property Name	Value	
Evaluation Development Board	None Specified	-
Product Category	All	-
Family	Spartan3A and Spartan3AN	-
Device	XC3S700A	-
Package	FG484	•
Speed	-4	•
Top-Level Source Type	HDL	-
Synthesis Tool	XST (VHDL/Verilog)	•
Simulator	ISim (VHDL/Verilog)	•
Preferred Language	Verilog	•
Property Specification in Project File	Store all values	•
Manual Compile Order		
VHDL Source Analysis Standard	VHDL-93	•
Frankla Marcana Filtration		

Dar click en "Next"

New Project Wizard
Project Summary
Project Navigator will create a new project with the following specifications.
Project: Project Name: my_first_time Project Path: V:\FPGAs\Semillero ADT\Practica_1\my_first_time
Working Directory: V:\FPGAs\Semillero ADT\Practica_1\my_first_time Description: Top Level Source Type: HDL
Device: Device Family: Spartan3A and Spartan3AN Device: xc3s700a Package: fg484 Speed: -4
Top-Level Source Type: HDL Synthesis Tool: XST (VHDL/Verilog) Simulator: ISim (VHDL/Verilog) Preferred Language: Verilog Property Specification in Project File: Store non-default values onl Manual Compile Order: false VHDL Source Analysis Standard: VHDL-93
More Info Finish Cancel

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.

New Source Wizard Select Source Type Select Source type, file name and its location. P (P (CORE Generator & Architecture Wizard) Schematic User Document Verilog Module Verilog Test Fixture VHDL Library VHDL Library VHDL Library VHDL Library Embedded Processor	File name: top_fpga Location: V:YPGAs(Semillero ADT)Practica_1\my_first_time
More Info	Add to project Next Cancel



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

New Source Wizard						×
Define Module Specify ports for module. Module name top_fpga						
Port Name	Direction		Bus	MSB	LSB	*
	input	-				
	input	-				
	input	-				
	input	-				
	input	-				Ξ
	input	-				
	input	-				
	input	-				
	input	-				
	input	-				
	input	-				-
More Info				Next	Cancel	

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

		×
6	New Source Wizard	
	Summary Project Navigator will create a new skeleton source with the following specifications. Add to Project: Yes Source Directory: V:\FPGAs\Semillero ADT\Practica_1\my_first_time Source Type: Verilog Module Source Name: top_fpga,v Module name: top_fpga Port Definitions:	
	More Info Finish Car	ncel

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.

```
21 module top_fpga(SW,LED) //Tipo de entrada (Switch),tipo de salida (Led)
22
       );
23
      // Entradas desde el switch 0 al switch 3
input[3:0]SW;
// Salidas desde el led 0 al led 7,solo usamos como respuesta al los switches, del 0 al 3
userent/3:02128
24
25
26
      output[7:0]LED;
//Asignación de estado bajo para los leds que no se van a usar,para alto, poner 1
27
28
      assign LED[7:3]=5'b00000;
29
30
31
       //Asignación de variables simbólicas en código, a salidas y entradas físicas
32
       circuitico ejercicio1(
33
       .A(SW[0]),
       .B(SW[1]),
34
       .C(SW[2]),
35
36
       .D(SW[3]),
37
38
        .X(LED[2]),
39
       .Y(LED[1]),
        .Z(LED[0])
40
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:

Hierarchy my_first_time xc3s700a-4fg484 W top_fpga (top_f	1	Image: second	9 // 0 // 1 mc	<pre>// //////////////////////////////////</pre>
No Processes Running Processes: top_fpga	00	Manual Compile Order Set as Top Module SmartGuide		<pre>//Asignac: circuitic(.A(SW[0])) .B(SW[1]) .C(SW[2]) .C(SW[2])</pre>
Design Summary/R	-	File/Path Display		• .X(LED[2])
User Constraints Synthesize - XST Call Implement Design Generate Programn		Expand All Collapse All		.Y(LED[1] .Z(LED[0]);
Configure Target De Analyze Design Usir	Μ	Find	Ctrl+	F
e mayze besign osi		Design Properties Source Properties		odule
100	1.1			

¡Así es!, seleccionamos circuitico.v



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

cuitico.v	All	▼ wo	ork		

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".



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.

≽ Start 💐 Design 🖺 Files 🖺 Libraries 💟 Design Summary (Programming File Generated) 💟 📄 circuitco.v 🛛 📄 top_fpga.v	
Errors	↔ □ & ×
	*
	-
e	E. F.
🔋 Console 🙆 Errors 🔔 Warnings 🙀 Find in Files Results	
	Ln 21 Col 1 Verilog

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.



,₽ ,₽	Name	Value	1,000,0 . 1999,996 ps 1999,997 ps 1999,998 ps 1999,999 ps 1,000,0	000 ps
▲ ● ● ● ● ● ●	Name Image: A Image: B Image: C Image: C <t< th=""><th>Value 2 2 2 2 x x x x x x x x x x x x</th><th></th><th></th></t<>	Value 2 2 2 2 x x x x x x x x x x x x		
	 ↓ X_sub3 ↓ X_sub4 ↓ Y_sub[8:0] ↓ Z_sub[7:0] 	x x xxxxxxxx xxxxxxx		
			X1: 1,000,000 ps	

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.

Æ								1,000,000 ps	*
2	Name	Value		999,996 ps	999,997 ps	999,998 ps	999,999 ps	1,000,000 ps	
~	Ц 🖌	2							
~	1 🖬 в	Z							
6	l <mark>la</mark> c	Z							
9	l <mark>e</mark> D	Z							
1 <u>+</u>	l <mark>la</mark> x	х							
er	Цаү	х							
-	l <mark>la</mark> z	х							
-	🔥 X_su 💥 Cu	t	Ctrl+X						
	📙 X_su 🛅 Co	py	Ctrl+C						
<u> </u>	Lita X_su 🛄 Pa	ste	Ctrl+V						
5	Le X_su 🖌 De	lete	Del						
(XI)	P Su Re	name							
	F C Z_SU	nume			XXXXXXXXX				
	A Fin	id	Ctrl+F						
	Sel	ect All	Ctrl+A						
	Exp	band							
	Co	llapse							
	Un	group							
	Na	me	•	00 ps					
	∢ Ra	dix							$\overline{\mathbf{v}}$



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:

	P 🕅 🏓 🖉	9 🔊 🥕 🗟) 🗠 🖭 🕇 👔	► → ₁ 🖬 🕨 🖡	X 1.00us 💌 😼	🗔 Re-launch		
P				1.000000 us				<u>^</u>
2	Name	Value	0 us	1us	2 us	3 us	4 us	5 us
2	1 <mark>6</mark> A 10 B	Z						
<u>@</u>	l <mark>la</mark> c	z						
9	l <mark>l</mark> D	Z						
t£ ≁r	ц <u>ю</u> х Ц <mark>ы</mark> х	x x						
	lla z	х						
÷								
<u></u>								
			X1: 1.000000 us					

Name		Value	0 us	
	Cut		Ctrl+X	
	Сору		Ctrl+C	
1, 0	Paste		Ctrl+V	
1 <mark>1.</mark> x 🗙	Delete	1	Del	
Ц <mark>ь</mark> Y	Renan	ne		
Ъ z м	Find		Ctrl+F	
	Select	All	Ctrl+A	
	Expan	d		
	Collap	se		
	Ungro	up		
	Name		•	
	Radix		×.	
	Signal	Color	۱.	
	Divide	r Color	E.	
	Revers	se Bit Order		
	Go To	Source Code		
	Show	Drivers)0000 i
-	Force	Constant		
	Force	Clock		
	Remo	ve Force		
	New G	Group		
difference	New D	Divider		
IX III	New V	/irtual Bus		
_				

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:

U B .	^ 🥟					
	1 🦾	Name	Value		0 us 1 u	ıs
	- ~	Ц 🔒 🗛	Z			
		Ца в	Z			
		lll c	Z			
	9	l <mark>l</mark> d	Z			
100	12	lla x	х			
ſ	For	ce Selected Signal	-		? <mark>×</mark>	Ŋ
	Enter value previo overri	parameters below t Assignments made busly applied consta idden.	to force the from with nt or clock	e sig nin H fore	gnal to a constant IDL code or any ce will be	
	Signal I	Name:		/circ	uitico/B	
	Value F	Radix	1	Binar	y 💌	
	Force t	to Value:		0		
	Startin	g at Time Offset:		0		
	Cancel	after Time Offset:	[
		OK Cancel	A	pply	Help	
			+	•	•	

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.**

ר 📰 xc3s700a-4fg484	Ξ
🖨 🔽 top_fpga (top_fpga.v)	10
🦢 💟 ejercicio1 - circuitico (circuitico	
	1
	×
	74
	×
4	3
	6
No Processes Running	\leq
rocesses: top_fpga	*
🗁 🎾 🛛 ISim Simulator	
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.

Þ					2.000000 us	
2	Name	Value	0 us	1us	2 us	3 us
2	🔻 📷 SW[3:0]	1001	ZZZZ	1001		
~	[3] ال	1				
6	[2] الم	0				
9	լը [1]	0				
1	נס]	1				
≁r	🔻 📑 LED[7:0]	00000110	00000XXX	00000110		
<u> </u>	رم 🔓	0				
Ť	[6] الم	0				
j 🕋	تو [5]	0				
\sim	[4]	0				
岡	Ц [3]	0				
พ	l <mark>o</mark> [2]	1				
CUU CUU	l <mark>la</mark> [1]	1				
. 🛄	[0]	0				
			X1: 2.000000 us			





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.

Welcome	S
Welcome to PlanAhead	1
ISE Integration Flow	
The PlanAhead [™] software provides an environment to help improve your design results throughout the design flow. Integrated within the Project Navigator environment, PlanAhead is automatically launched at four different design process steps. These include VO Pin planning (Pre-synthesis), VO Pin planning (Pre-synthesis), Floorplan Area/OLogic (Post-synthesis), Analyze Timing and Floorplan Design (Post-implementation). Each of these steps offers unique and powerful capabilities previously only available in the standalone PlanAhead environment. The PlanAhead software replaces PACE and Floorplanner for all pin planning, design viewing, and floorplanning flows.	
When PlanAhead is invoked from Project Navigator, the interface provides access only to the PlanAhead features specific to the selected task. The standalone PlanAhead environment offers the full range of PlanAhead RTL to bitstream design and analysis capabilities. For more information about the standalone PlanAhead environment, please refer to the PlanAhead User Guide, Tutorial or visit <u>www.xilinx.com/planahead</u> .	
For more information, please review the various documents below.	
D Sourcing Tcl script 'V:/FPGAs/Semillero ADT/Practica_1/my_first_time/pa.fromHdl.tcl'	ח
Image: Second	
Chapter 8 - I/O Pin Planning	
Chapter 7 - Design Analysis and Constraint Definition	
Chapter 10 - Analyzing Implementation Results	
Chapter 11 - Floorplanning	
These chapters are all available with the ISE Help documentation as well.	
Video Demonstrations	
Video Demonstrations Visit <u>www.xilinx.com/design</u> for a variety of self running video demonstrations describing ISE Design Suite [™] software features and methodology.	
Video Demonstrations Visit <u>www.xilinx.com/design</u> for a variety of self running video demonstrations describing ISE Design Suite [™] software features and methodology. Tutorial Select <i>Help > Tutorials</i> for a list of available PlanAhead Tutorials	

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).

Name	Direction	Neg Diff Pair	Site		Fixed	Bank	I/O Std	Vcco	Vref	Drive Stre	Slew Type	Pull Type
E- 🗁 All ports (12)												
🖹 🚱 LED (8)	Output						default (LVCMOS25)	2,500		12	2 SLOW	NONE
- 🖸 LED[7]	Output						default (LVCMOS25)	2.500		12	2 SLOW	NONE
LED[6]	Output		A2				default (LVCMOS25)	2.500		12	2 SLOW	NONE
	Output		A3				default (LVCMOS25)	2,500		12	2 SLOW	NONE
	Output		A4				default (LVCMOS25)	2.500		12	2 SLOW	NONE
	Output		A5				default (LVCMOS25)	2.500		12	2 SLOW	NONE
	Output		A6				default (LVCMOS25)	2.500		12	2 SLOW	NONE
	Output		A7				default (LVCMOS25)	2.500		12	2 SLOW	NONE
(IED[0]	Output		A8	Ŧ			default (LVCMOS25)	2.500		12	2 SLOW	NONE
🖻 👺 SW (4)	Input		A9				default (LVCMOS25)					NONE
D→ SW[3]	Input						default (LVCMOS25)					NONE

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

Name	Direction	Neg Diff Pair	A2	W	Fixed	Bank	I/O Std	Vcco	Vref	Drive Stre	Slew Type	Pull Type
	Output		A3		V	1	default (LVCMOS25)	2.500		12	SLOW	NONE
	Output		A4		V	1	default (LVCMOS25)	2.500		12	SLOW	NONE
	Output		A5		V	1	default (LVCMOS25)	2.500		12	SLOW	NONE
	Output		A6		V	1	default (LVCMOS25)	2.500		12	SLOW	NONE
	Output		A7		V	1	default (LVCMOS25)	2.500		12	SLOW	NONE
	Output		A 8		V	1	default (LVCMOS25)	2.500		12	SLOW	NONE
🖻 🕒 SW (4)	Input		A9	-			default (LVCMOS25)					NONE
- 🕞 SW[3]	Input						default (LVCMOS25)					NONE
- D SW[2]	Input						default (LVCMOS25)					NONE
- D SW[1]	Input						default (LVCMOS25)					NONE
	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*	3.3	00	12	2 SLOW	NONE
	Output		W21	V		1 LVCMOS33	3.3	00	12	2 SLOW	NONE
	Output		Y22	V		1 LVCMOS12 🔺	3.3	00	12	2 SLOW	NONE
	Output		V20	1		1 LVCMOS15	3.3	00	12	2 SLOW	NONE
	Output		V19	V		1 LVCMOS18	3.3	00	12	2 SLOW	NONE
	Output		U19	V		1 LVCMOS25	3.3	00	12	2 SLOW	NONE
	Output		U20	V		1 LVCMOS33	3.3	00	12	2 SLOW	NONE
	Output		T19	V		1 LVTTL	3.3	00	12	2 SLOW	NONE
	Output		R20	V		1 PCI33_3 👻	3.3	00	12	2 SLOW	NONE
🖃 👺 SW (4)	Input					2 PCI66_3 25	5)				NONE
- 🐼 SW[3]	Input		Т9			2 default (LVCMOS2	5)				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
E MI ports (12)											
📮 🔞 LED (8)	Output					1 LVCMOS33*	3.300		12 🗸	SLOW	NONE
	Output		W21	V		1 LVCMOS33*	3.300		2	SLOW	NONE
	Output		Y22	V		1 LVCMOS33*	3.300		4	SLOW	NONE
	Output		V20	V		1 LVCMOS33*	3.300		6	SLOW	NONE
	Output		V19	V		1 LVCMOS33*	3.300		8	SLOW	NONE
	Output		U 19	V		1 LVCMOS33*	3.300		12	SLOW	NONE
	Output		U20	V		1 LVCMOS33*	3.300		16	SLOW	NONE
	Output		Т19	V		1 LVCMOS33*	3.300		24	SLOW	NONE
	Output		R20	V		1 LVCMOS33*	3.300		12	SLOW	NONE
🗐 🤒 SW (4)	Input					2 LVCMOS33*					NONE
- 🖾 SW[3]	Input		Т9	V		2 LVCMOS33*					NONE
🔜 Td Console 🔎 Package Pins	D→ 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
	Output		W21	V		1 LVCMOS33*	3.300		8*	* SLOW	NONE
	Output		Y22	v		1 LVCMOS33*	3.300		8	* SLOW	NONE
	Output		V20	V		1 LVCMOS33*	3.300		8'	* SLOW	NONE
	Output		V19	V		1 LVCMOS33*	3.300		81	SLOW	NONE
	Output		U19	1		1 LVCMOS33*	3.300		8'	* SLOW	NONE
	Output		U20	1		1 LVCMOS33*	3.300		8*	* SLOW	NONE
	Output		T19	1		1 LVCMOS33*	3.300		8	* SLOW	NONE
	Output		R20	v		1 LVCMOS33*	3.300		8	* SLOW	NONE
🖃 🤒 SW (4)	Input					2 LVCMOS33*					NONE
🛛 🕑 SW[3]	Input		Т9	v		2 LVCMOS33*					NONE
- 🐼 SW[2]	Input		U8	1		2 LVCMOS33*					NONE
- 🐼 SW[1]	Input		U10	v		2 LVCMOS33*					NONE
	Input		V8	V		2 LVCMOS33*					NONE
🛄 🔂 Scalar ports (0)											

Processes: top_fpga							
···· Σ	Design Summary/Reports						
🛓 - 🏄	Design Utilities						
÷. 🏄	User Constraints						
🗄 🔁 🐴	Synthesize - XST						
⊕ ? 2	Implement Design						
- 62	Generate Programming File						
🗄 - 🛞 -	Configure Target Device						
ш. <u>е́н</u>	Analyze Design Using ChipScope						

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.

🚱 ISE IMPACT (P.20131013)						
File Edit View Operations Output Debug Window Help						
MPACT Flows ↔ □ ♂ ×						
– ∰ Boundary Scan – ■ SystemACE – ■ Create DPOM File Format						
WebTalk Data						
MPACT Processes ++ C & X						
Console	⇔⊡₽×					
	A					

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.

≽ ISE P	- ISE Project Navigator (P-20131013) - V/FPGAs/Semillero ADTr/Practica_1/my_first_time/my_first_time/size - [top_fogax] - 0 ×							
📄 File		dit View Project Source Process Too	Window Layout Help	- & ×				
i 🗋 🖻		ISE iMPACT (P.20131013) - [Boundary Scan]		1				
Design	1	File Edit View Operations Output	Debug Window Help					
Vier	1	🗋 🎓 🛃 🔓 📰 🗱 🖨 🖷	<i>₽</i> k ?					
		MPACT Flows ↔ □	Sight click device to select operations					
		● ■ Boundary Scan ■ SystemACE ■ Create PROM File (PROM File Format ● ■ WebTalk Data ● ■ WebTalk Data ● ■ WebTalk Data ● ■ ■ <td< td=""><td>TDI Example xc3a700a xcf04s bypass bypass TDO Do you want to continue and assign configuration files(s)? Do Do you want to continue and assign save the setting in preference. Yez No</td><td>. 3</td></td<>	TDI Example xc3a700a xcf04s bypass bypass TDO Do you want to continue and assign configuration files(s)? Do Do you want to continue and assign save the setting in preference. Yez No	. 3				
				ь				
Errors			Identify Succeeded	+ □ ♂ ×				
	П		Boundary Scan					
•	l	Console	++□∂×					
Co		PROGRESS_END - End Operation. Elapsed time = 0 sec.	^	Ln 21 Col 1 Verilog				
1		🥭 📋 🖸 📀	E .	In the second secon				

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**.

≽ ISE	Pro	oject Navigator (P.20131013) - V:\FPGAs\Semille	ro ADT\Practica_1\my_first_time\my_first_time.xise - (top_fpga.v)		_ 0 ×
📄 File	e	Edit View Project Source Process To	ols Window Layout Help		_ 6 ×
1	d	Boundary Scan			
Design		File Edit View Operations Output	Debug Window Help	×	A
III Vi	iew	Assign New Co	nfiguration File		
		MPACT Flows Look in: ■ SystemACE My Compute ■ SystemACE My Compute ■ WebTalk Data CiroGanutr	<pre>V:PPGAS/Semilero ADT/Practica_1/my_frst_time</pre>		F
		MPACT Processes Available Operations are: File name: top. Files of type: All D	fpga.bit cancel Bypass esign Files (*.bit *.rbit *.rkiy *.isc *.bsd)	L	3
> S Errors	itar		Identify Succeeded		↔□♂×
4		Console	Boundary Scan	×	
	Con	PROGRESS END - End Operation Elapsed time = 0 sec. // *** BATCH CMD : identifyM	94	^	Ln 21 Col 1 Verilog
7		🥖 📋 🖸 🧿	ES ES	۵ I	★ II:08 20/02/2015

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.

Device Programming Properties - Device Category	1 Programming Properties	×				
Boundary-Scan Device 1 (FPGA xc3s700a)	Property Name	Value				
Device 2 (PROM xcf04s)	Verify					
	FPGA Device Specific Programming Properties					
	; Pulse PROG					
	ОК	Cancel Apply Help				

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.

🍃 ISE I	Project Navigator (P.20131013) - V:\FPGAs\Semiller	o ADT\Practica_1\my_first_time\my_first_time.xise - [top_fpga.v]	
📄 File	Edit View Project Source Process Too	ls Window Layout Help	- 6 ×
2	SE ISE IMPACT (P.20131013) - [Boundary Scan]		1
Design	File Edit View Operations Output	Debug Window Help - 6 ×	×
iii Vie		x i # 🔍 🗖 🔎 🖗	
61 H	MPACT Flows ++ D R X		
8	B 22 Boundary Scan	19861902	
	SystemACE		
_61	Create PROM File (PROM File Format	TDI EXUMP EVENT	
-	and the last bata		
		xc3s700a xcf04s	1
<u> </u>		top_fpgs.bit bypess	
		TD0	
1*			
5 1			
TE Pr			
20			
9	MPACT Processes ++		
-	Available Operations are:		
m 🖷	Get Device ID		
4	Get Device Signature/Usercode		3
1	Read Device Status		
	One Step SVF One Step XSVF		
			· · · ·
		Identify Succeeded	
Errors		Identity Succeeded	+08×
			Î Î
		Reader Cran	
	Consta		
	UTNEO: IMPACT: 501 - 111: Added	+U & X	· · · ·
	A THEFT AND A THE AND A THE ADDRESS AND A THE AD	verze wooloom pwoeooxwarzi.	In 21 Cold Medice
			Ln ZI Col I Venlog
1		🥕 😻 👘 -	20/02/2015

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.

