ALU (Unidad Aritmética Lógica).

Facultad: Ingeniería.
Escuela: Electrónica.
Asignatura: Sistemas Digitales II.
Lugar de Ejecución: Microprocesadores (3.23).
Instituto de Investigación e Innovación en Electrónica.
Docente: Kelman Belloso.

Objetivo General.

- Que el alumno diseñe e implemente circuitos simples basados en arreglos de compuertas básicas por medio de VHDL.

Objetivo Específicos.

- Diseñar y simular una ALU.

Material y equipo.

- 1 Computadora con ISE 14.7 instalado.
- 1tarjeta Spartan 6 - LX9-MicroBoard.
- 1 tarjeta de entradas / salidas (E/S o I/O).
- 1 tarjeta con display.
- 18 cables (jumpers) macho – macho.

Tarea previa.

1. Ver los videos: ISE Circuit: https://www.youtube.com/watch?v=kx9ql74iWME
   Intro ISE: https://www.youtube.com/watch?v=fnWVHz488Mw
   ISE TESTBENCH: https://www.youtube.com/watch?v=Ww6lO_mUVEI

Introducción teórica

- **Números no signados (todos los números son positivos).**

$0111_2 = 7_{10}$
$1111_2 = 15_{10}$

- **Números signados (el primer bit define el signo).**

$0111_2 = 7_{10}$
$1111_2 = -7_{10}$

- **Suma binaria.**

Ci			**1**	**1**		1		1
a	0	0	0	1	1	0		6
b	0	0	0	0	1	0		2
	0	0	1	0	0	1		9

- **Resta binario.**

a		0	0	0	1	1	0	6
b		0	0	0	0	1	0	- 2
		0	0	0	1	1	0	4
		1	1	1	1	0	1	A1
	1	1	1	1	1	1	1	A2
	1	0	0	0	1	0	0	4

+

	0	0	0	0	1	0		2
	0	0	0	1	1	0	-	6
	0	0	0	0	1	0	-	4
	1	1	1	0	0	1		A1
				1	1	1		A2
0	1	1	1	1	0	0	-	4

−

	0	0	0	0	1	1	A1
				1	1	1	A2
	0	0	0	1	0	0	4

- **Multiplicación binaria (no signada y signada).**

a	b
6	2

1	1	0	X	0	1	0	6	x 2
			0	0	0			1 2
		1	1	0				
	0	0	0					
	0	1	1	0	0			1 2

a	b
- 2	2

1	1	0	X	0	1	0	-2	x 2
	1	0	X		1	0	A2	x 2
				0	0			- 4
			1	0				
1		1	0	0				- 4

- **División binaria (no signada y signada).**

a	b
6	2

1	1	0	1	0	6	/ 2
1	0		1	1		3
	1	0				
	1	0				
		0				

a	b
- 2	2

1	1	0	0	1	0	-2 / 2
	1	0	1		1	- 1
1	0	0				

+

- **ALU**

La unidad aritmética lógica o unidad aritmético-lógica, también conocida como ALU (siglas en inglés de arithmetic logic unit), es un circuito digital que calcula operaciones aritméticas (como suma, resta, multiplicación, etc.) y operaciones lógicas (si, y, o, no), entre valores (generalmente uno o dos) de los argumentos.

Muchos tipos de circuitos electrónicos necesitan realizar algún tipo de operación aritmética, así que incluso el circuito dentro de un reloj digital tendrá una ALU minúscula que se mantiene sumando 1 al tiempo actual, y se mantiene comprobando si debe activar el sonido de la alarma, etc.

Por mucho, los circuitos electrónicos más complejos son los que están construidos dentro de los chips de microprocesadores modernos. Por lo tanto, estos procesadores tienen dentro de ellos un ALU muy complejo y potente. De hecho, un microprocesador moderno (y los mainframes) puede tener múltiples núcleos, cada núcleo con múltiples unidades de ejecución, cada una de ellas con múltiples ALU.

Muchos otros circuitos pueden contener en el interior una unidad aritmético lógica: unidades de procesamiento gráfico como las que están en las GPU modernas, FPU como el viejo coprocesador matemático 80387, y procesadores digitales de señales como los que se encuentran en tarjetas de sonido, lectoras de CD y los televisores de alta definición. Todos éstos tienen en su interior varias ALU potentes y complejas.

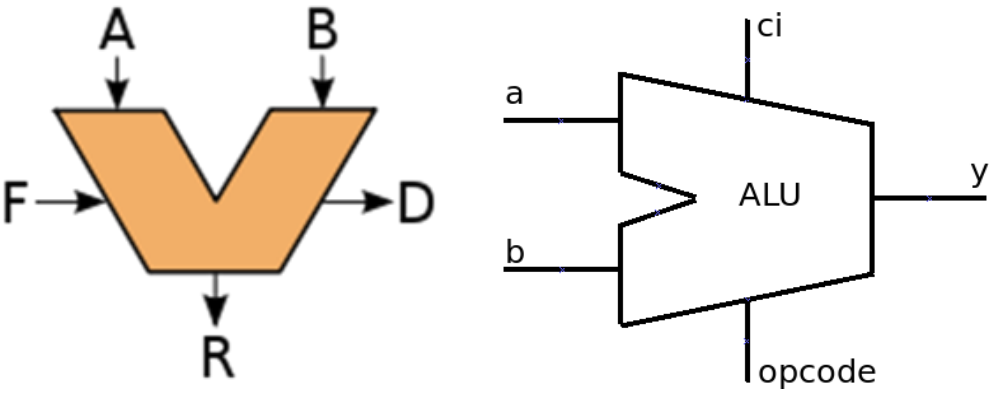

Figura 1: Símbolo general de una ALU.

Selección (opcode)				Instrucción	Operación	Unidad
op3	op2	op1	op0			
0	0	0	0	Comp. A	y = NOT(a)	Lógica
0	0	0	1	Comp. B	y = NOT(b)	
0	0	1	0	AND	y = a AND b	
0	0	1	1	NAND	y = a NAND b	
0	1	0	0	OR	y = a OR b	
0	1	0	1	NOR	y = a NOR b	
0	1	1	0	XOR	y = a XOR b	
0	1	1	1	XNOR	y = a XNOR b	
1	0	0	0	mov a	y = a	Aritmética
1	0	0	1	mov b	y = b	
1	0	1	0	incr. a	y = a + 1	
1	0	1	1	decr. a	y = a - 1	
1	1	0	0	Suma	y = a + b + Ci	
1	1	0	1	Resta	y = a - b	
1	1	1	0	Multiplicación	y = a * b	
1	1	1	1	División	y = a / b	

Tabla 1: Funciones de la ALU para las diferentes operaciones de selección.

Procedimiento.

Parte I: Creación de proyectos con ISE para Spartan-6 LX9 CSG324.

1) Desarrolle un nuevo proyecto como **ALU**, para probarse en la tarjeta Spartan 6 LX9 Microboard.
2) Cree el archivo que contenga el código VHDL propuesto.

```
----------------------------------------------------------------------
-- Ejemplo 5.3: Página 127-129, Circuit Design and
-- Simulation with VHDL, Volnei A. Pedroni
----------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity ALU is
   generic (N      : integer := 2);              --tamaño de las entradas
   Port ( a        : in STD_LOGIC_VECTOR (N downto 0);
          b        : in STD_LOGIC_VECTOR (N downto 0);
          Ci       : in STD_LOGIC;
          Opcode   : in STD_LOGIC_VECTOR (N+1 downto 0);
          y        : out STD_LOGIC_VECTOR (((2*N)+1) downto 0));
end ALU;

architecture Behavioral of ALU is
   signal a_sig, b_sig: signed (N downto 0);    -- Declaración de señales con signo
   signal y_sig: signed (((2*N)+1) downto 0);
   signal y_unsig: STD_LOGIC_VECTOR (((2*N)+1) downto 0);
   signal c_int: integer range 0 to 1;
begin
-------------------------------------------------------
------------------- Unidad lógica --------------------
-------------------------------------------------------
   with opcode (N downto 0) select              -- El símbolo & sirve para concatenar bits
      y_unsig <=  "111" & (not (a)) when "000",
                  "111" & (not (b)) when "001",
                  "000" & (a and b) when "010",
                  "111" & (a nand b) when "011",
                  "000" & (a or b) when "100",
                  "111" & (a nor b) when "101",
                  "000" & (a xor b) when "110",
                  "111" & (a xnor b) when others;

-------------------------------------------------------
----------------- Unidad aritmética -------------------
-------------------------------------------------------
   a_sig <= signed(a);                          -- Pasar entradas a entradas con signo
   b_sig <= signed(b);
   c_int <= 1 when Ci='1' else 0;

   with opcode (N downto 0) select
      y_sig <= "000" & a_sig when "000",
               "000" & b_sig when "001",
               ("000" & a_sig) + 1 when "010",
               ("000" & a_sig) - 1 when "011",
               ("000" & a_sig) + ("000" & b_sig) + c_int when "100",
               ("000" & a_sig) - ("000" & b_sig) when "101",
               a_sig * b_sig when "110",
               "00" & (('0' & a_sig) / ('0' & b_sig)) when others;
```

```
-------------------------------------------------------
-------------------- Multiplexor ----------------------
-------------------------------------------------------
   with opcode(N+1) select
      y <=  y_unsig when '0',
            STD_LOGIC_VECTOR (y_sig) when others;
end Behavioral;
```

Código 1: Código VHDL para ALU descrita en la tabla 1.

3) Verifique el funcionamiento de la ALU con el Test Bench siguiente:

```
LIBRARYieee;
USEieee.std_logic_1164.ALL;

ENTITY ALU_tb IS
END ALU_tb;

ARCHITECTURE behavior OF ALU_tb IS
    -- Component Declaration for the Unit Under Test (UUT)
  COMPONENT ALU
  Port ( a  : in STD_LOGIC_VECTOR (2 downto 0);
         b  : in STD_LOGIC_VECTOR (2 downto 0);
         Ci : in STD_LOGIC;
         Opcode : in STD_LOGIC_VECTOR (3 downto 0);
         y  : out STD_LOGIC_VECTOR (5 downto 0));
        );
  END COMPONENT;

   --Inputs
   signal a : std_logic_vector (2 downto 0) := (others => '0');
   signal b : std_logic_vector (2 downto 0) := (others => '0');
   signal Ci : std_logic := '0';
   signal opcode : std_logic_vector (3 downto 0) := (others => '0');

    --Outputs
   signal y : std_logic_vector (5 downto 0);

BEGIN
    -- Instantiate the Unit Under Test (UUT)
    uut: ALU PORT MAP (
         a => a,
         b => b,
         Ci => Ci,
         opcode => opcode,
         y => y,
    );

    -- Stimulus process
    stim_proc: process
    begin
    -- hold reset state for 10 ns.
       a  <= "110";
       b  <= "010";
       Ci <= '0';
       opcode <= "0000";
       wait for 10 ns;
       Ci <= '1';
       wait for 10 ns;
       opcode <= "0001";
       wait for 10 ns;
       opcode <= "0010";
       wait for 10 ns;
       opcode <= "0011";
       wait for 10 ns;
       opcode <= "0100";
       wait for 10 ns;
```

```
            opcode <= "0101";
            wait for 10 ns;
            opcode <= "0110";
            wait for 10 ns;
            opcode <= "0111";
            wait for 10 ns;
            Ci <= '0';
            opcode <= "1000";
            wait for 10 ns;
            opcode <= "1001";
            wait for 10 ns;
            opcode <= "1010";
            wait for 10 ns;
            opcode <= "1011";
            wait for 10 ns;
            opcode <= "1100";
            wait for 10 ns;
            Ci <= '1';
            wait for 10 ns;
            opcode <= "1101";
            wait for 10 ns;
            a  <= "010";
            b  <= "110";
            wait for 10 ns;
            opcode <= "1110";
            wait for 10 ns;
            a  <= "110";
            b  <= "010";
            opcode <= "1111";
            wait for 10 ns;
        wait;
        end process;
    END;
```

Código 2: Código VHDL del Test Bench, archivo ALU_tb.vhd.

4) Observe si el diagrama de tiempos es tal como el mostrado a continuación:

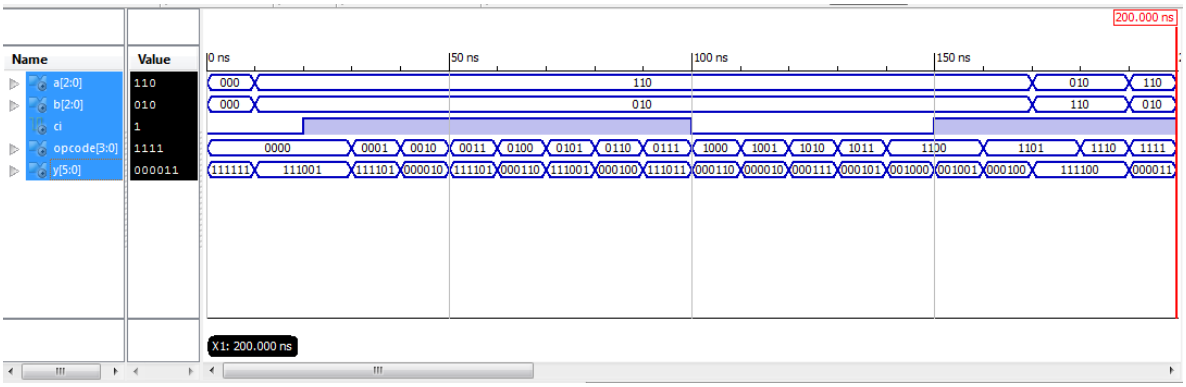

Figura 2: Diagrama de tiempos esperado para el test bench código 2.

Parte II: Utilizar PlanAhead.

5) Dé clic izquierdo en el menú: **Tools >> PlanAhead >> I/O Pin Planning (PlanAhead) – Pre-Synthesis**.
6) Se desplegará un cuadro de dialogo, elija la opción Yes.

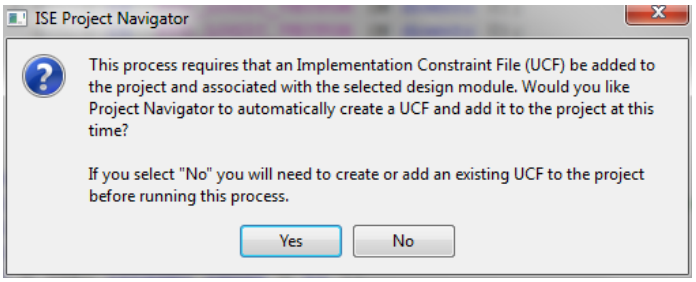

Figura 3: Cuadro de dialogo, creación de archivo User constraint.

7) Le preguntará si desea abrir el **PlanAhead** a lo que debe responder **Yes**.
8) Espere a que cargue la aplicación **PlanAhead** y cierre el cuadro de dialogo de bienvenida (clic izquierdo en el botón **Close**).
9) En el recuadro inferior, seleccione el estándar 3.3V para tecnología CMOS en el bloque de la variable **y (salidas)** tal como se muestra en la figura 4.

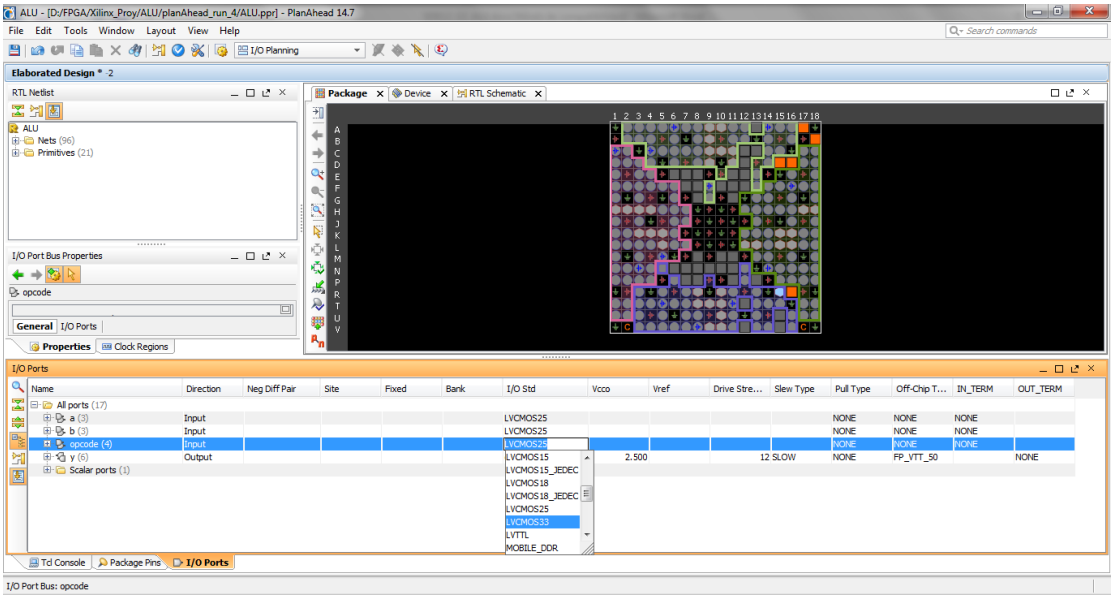

Figura 4: Aplicación PlanAhead, creación de archivo User constraint.

10) Sobre la sección **Site** de cada una de las entradas elija los valores de la tabla 2, según se muestran las conexiones en la figura 5.

Variables del código VHDL	Nombre de los pines de la FPGA
a[2]	F18
a[1]	F17
a[0]	K13
b[2]	K12
b[1]	E18
b[0]	E16

Variables del código VHDL	Nombre de los pines de la FPGA
Ci	G13
opcode[3]	A4
opcode[2]	B4
opcode[1]	A3
opcode[0]	B3

Variables del código VHDL	Nombre de los pines de la FPGA
y[5]	D18
y[4]	D17
y[3]	G14
y[2]	F14
y[1]	F15
y[0]	F16

Tabla 2: Tabla de asignaciones de pines conforme al circuito de la figura 5.

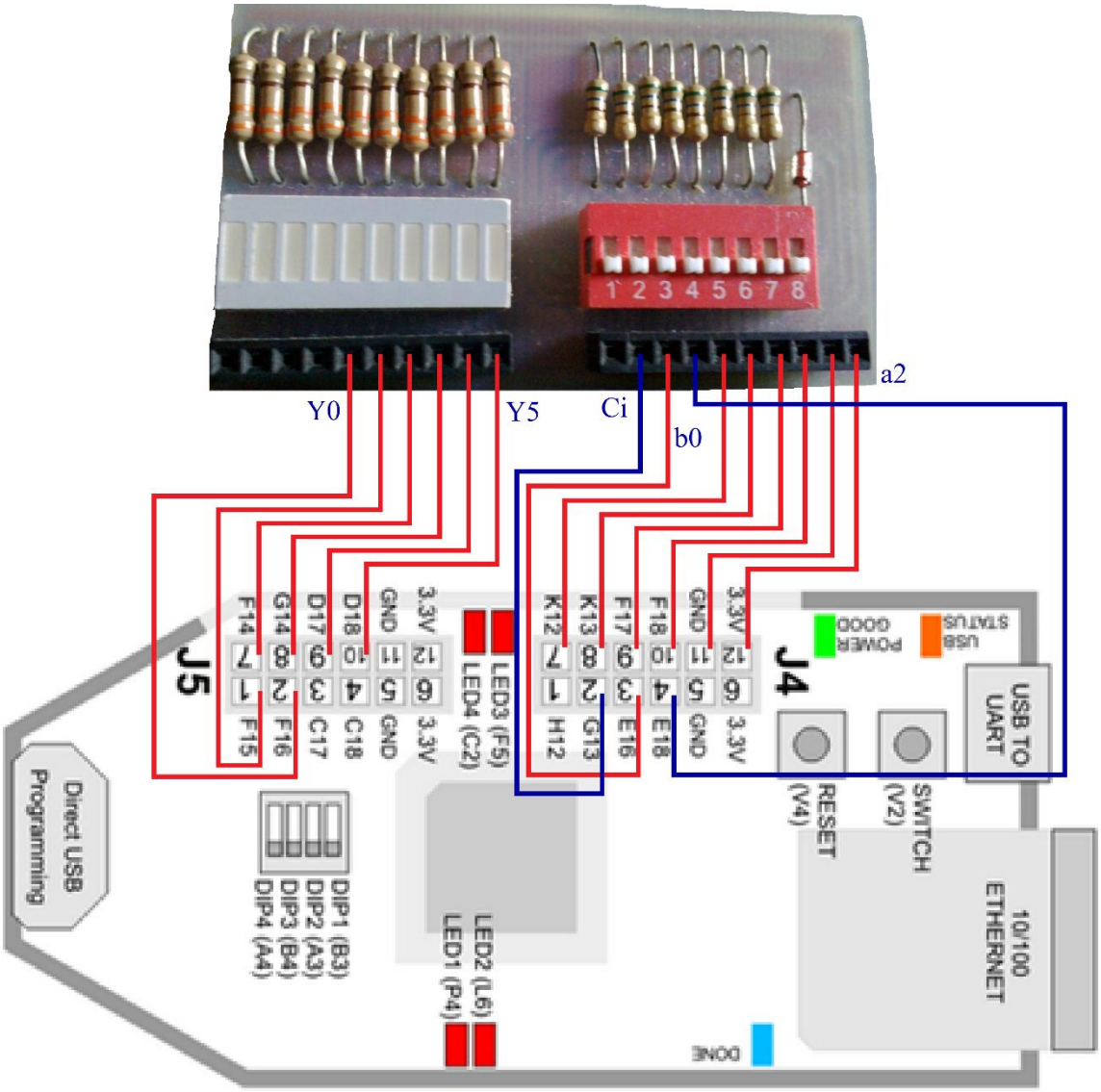

Figura 5: Conexión de pruebas para circuito ALU.

Nota:

*Note que a medida que vaya agregando las relaciones en el recuadro de color de fondo negro, con nombre **Package**, se irán habilitando los nombres asignados conforme a la cuadricula donde las filas representan las letras y las columnas los números similar a la cuadricula de una hoja de cálculo. Esa es la distribución de los pines en el interior de la FPGA. Así los pines de **y**, **a**, **b** y **Ci** se encuentran en el **banco 0** y los de **opcode** en el **banco 0**.*

11) Guarde el archivo en la ventana **PlanAhead**.
12) Siempre en la ventana **PlanAhead**, dé clic en el menú **File >> Export >> Export I/O ports…**
13) En la ventana desplegable de la figura 6 seleccione solamente el tipo de archivo **UCF**, seleccione una ruta donde guardar el archivo y de clic en el botón **OK**.
14) Cierre la aplicación **PlanAhead**.

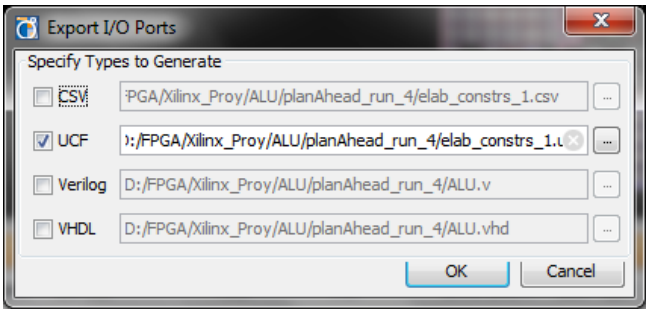

Figura 6: Verificación de archivo KBH004_LX9_LUT.ucf.

15) Verifique en la ventana de **ISE Project Navigator**, que el archivo ALU.ucf ya se encuentre debajo del archivo **ALU.vhd** en la sección **Hierarchy**.
16) Si el archivo **ALU.ucf** no aparece entonces, dé clic derecho sobre el nombre del archivo con extensión **.vhd** y elija la opción **Add Source..**, busque la carpeta **planAhead_run_4**[1] y dentro de ella el nombre del archivo **elab_constrs_1.ucf**, de clic en el botón **Open**.
17) En la ventana desplegable de clic en el botón **OK**. Deberá observar un archivo similar al del Código 3.

```
1  # PlanAhead Generated IO constraints
2  NET "y[5]" IOSTANDARD = LVCMOS33;
3  NET "y[4]" IOSTANDARD = LVCMOS33;
4  NET "y[3]" IOSTANDARD = LVCMOS33;
5  NET "y[2]" IOSTANDARD = LVCMOS33;
6  NET "y[1]" IOSTANDARD = LVCMOS33;
7  NET "y[0]" IOSTANDARD = LVCMOS33;
8
9  # PlanAhead Generated physical constraints
10 NET "Ci" LOC = G13;
11 NET "a[2]" LOC = F18;
12 NET "a[1]" LOC = F17;
13 NET "a[0]" LOC = K13;
14 NET "b[2]" LOC = K12;
15 NET "b[1]" LOC = E18;
16 NET "b[0]" LOC = E16;
17 NET "opcode[3]" LOC = A4;
18 NET "opcode[2]" LOC = B4;
19 NET "opcode[1]" LOC = A3;
20 NET "opcode[0]" LOC = B3;
21 NET "y[5]" LOC = D18;
22 NET "y[4]" LOC = D17;
23 NET "y[3]" LOC = G14;
24 NET "y[2]" LOC = F14;
25 NET "y[1]" LOC = F15;
26 NET "y[0]" LOC = F16;
```

Código 3: Archivo descriptor de pines de la tarjeta Spartan para la aplicación de la figura 5.

18) Conecte la tarjeta Spartan tal como lo muestra la figura 6.
19) Genere el archivo .bit y programe la tarjeta.
20) Verifique el funcionamiento del VHDL. Accione combinaciones con los **minidip** y vea el resultado en el led de barra.
21) Desconecte la tarjeta Spartan de la computadora.
22) Desconecte la tarjeta de I/O's de la tarjeta Spartan.
23) Cierre las aplicaciones abiertas y apague la computadora.

[1] Según imagen de la figura 6.

Investigación complementaria

1. Probar la ALU para operaciones: mayor que, menor que, igual que, concatenar, división signada, multiplicación no signada, residuo, valor absoluto, exponenciación (a**b) signada y no signada para números de 3 bits. Revise el capítulo 4 de Circuit Design and Simulation with VHDL.
2. Desarrollar un convertidor de binario (de 4 bits) a Gray, binario a BCD exceso tres, binario a ASCII.
3. Desarrollar un convertidor de binario (6 bits) a BCD de (8 bits).
4. Realice un intérprete del set de instrucciones del microcontrolador PIC16F84A, los códigos se introducirán en números hexadecimales multiplexados, las operaciones se realizarán con números de 4 bits, omita por el momento las instrucciones relacionadas a guardar datos en memoria.

Bibliografía

1. http://www.erikavilches.com/Anterior/TC1004.01.200811/diapositivas/Aritmetica%20Numeros%20con%20Signo%203.pdf
2. http://www2.dis.ulpgc.es/~itis-sd/Transparencias0607/Tema03.pdf
3. https://eciencia.urjc.es/bitstream/handle/10115/4045/diseno_de_circuitos_digitales_con_vhdl_v1.01.pdf?sequence=3&isAllowed=y
4. Pedroni, Volnei A. Circuit Design and Simulation with VHDL (2nd Edition). (2010). MIT Press: http://app.knovel.com/hotlink/toc/id:kpCDSVHDLN/circuit-design-simulation/circuit-design-simulation
5. Ashenden, Peter J. Designer's Guide to VHDL (3rd Edition). (2008). Elsevier.: http://app.knovel.com/hotlink/toc/id:kpDGVHDLEO/designer-s-guide-vhdl/designer-s-guide-vhdl