Universidad Nacional Autonoma de Mexico Facultad de ingeniería

Diseño Digital VLSI

Grupo : 4

Práctica: 4

DISEÑO DE CONTROL DE SERVOMOTORES

Edgar Daniel Barcenas Martinez

OBJETIVO:

El alumno aprenderá la manera de organizar un proyecto de manera modular y separarlo en diferentes archivos, con la finalidad de que vaya construyendo su propia biblioteca de módulos funcionales, y pueda reutilizar los módulos generados en otros proyectos.

ESPECIFICACIONES:

Diseñar el control de un servomotor de modelismo utilizando en un FPGA, en el cual, por medio de cuatro interruptores de presión tipo push-boton, se pueda controlar la posición del eje del motor. Dos de los interruptores permitirán llevar al eje a cada una de las posiciones extremas, mientras que los otros permitirán que el motor gire en cada dirección avanzando paso a paso a través de 12 posiciones definidas cada vez que el interruptor es presionado. La determinación de la posición se hará por medio de una señal PWM. La figura 4.1 muestra el diagrama del bloque de este sistema.

DIAGRAMA DE BLOQUES:

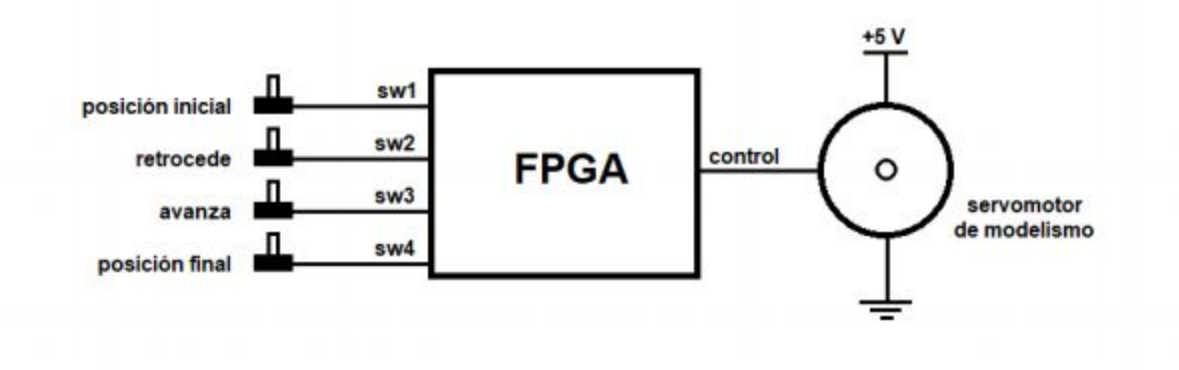


Figura 4.1. Diagrama de bloques del control de un servomotor de modelismo

BLOQUES FUNCIONALES:

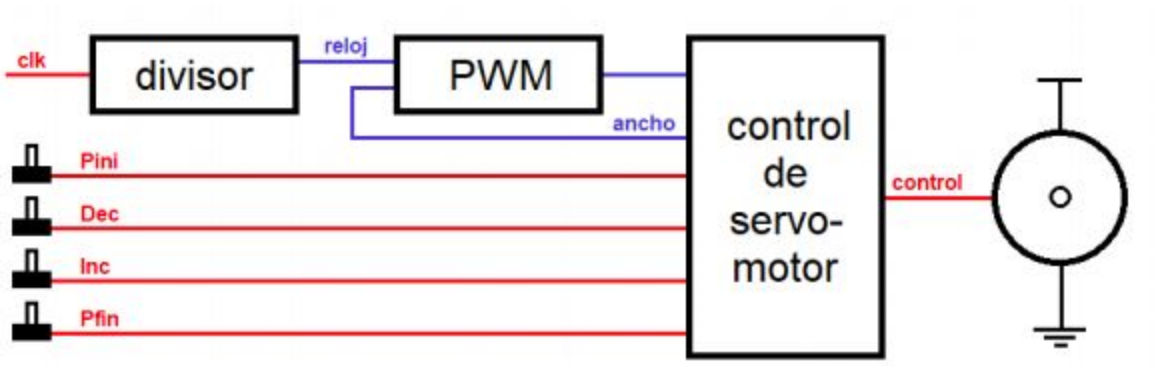


Figura 4.2. Bloques funcionales del control de servomotor

Actividad 1

Se programaron los componentes necesarios para desarrollar el correcto funcionamiento del servomotor, el código quedó de la siguiente manera:

Servomotor.vhd

Programa principal

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.STD_LOGIC_ARITH.ALL;
4   use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6   entity Servomotor is
7   Port ( clk : in   STD_LOGIC;
8   Pini : in      STD_LOGIC;
9   Pfin : in STD_LOGIC;
10  Inc : in STD_LOGIC;
11  Dec : in STD_LOGIC;
12  control : out  STD_LOGIC);
13  end Servomotor;
14
15  architecture arqServomotor of Servomotor is
16
17     component Prac4 is
18        Port ( clk : in std_logic;
19        div_clk : out std_logic);
20     end component;
21
22     component PWM is
23        Port ( Reloj : in STD_LOGIC;
24        D : in   STD_LOGIC_VECTOR (7 downto 0);
25        S : out     STD_LOGIC);
26     end component;
27
28     signal reloj : STD_LOGIC;
29     signal ancho : STD_LOGIC_VECTOR (7 downto 0) := X"0F";
30        begin
31
32        U1: Prac4 port map (clk, reloj);
33        U2: PWM port map (reloj, ancho, control);
34
35     process (reloj, Pini, Pfin, Inc, Dec)
36        variable valor : STD_LOGIC_VECTOR (7 downto 0) := X"0F";
37        variable cuenta : integer range 0 to 1023 := 0;
38        begin
39           if reloj='1' and reloj'event then
40              if cuenta>0 then
41                 cuenta := cuenta -1;
42              else
43              if Pini='1' then
44                 valor := X"0D";
45                 elsif Pfin='1' then
46                    valor := X"18";
47                 elsif Inc='1' and valor<X"18" then
48                    valor := valor + 1;
49                 elsif Dec='1' and valor>X"0D" then
50                    valor := valor - 1;
51              end if;
52                 cuenta := 1023; ancho <= valor;
53              end if;
54           end if;
55     end process;
   end arqServomotor;
```

Se define un reloj, dirección Izq, dirección Der como entradas para el circuito, y como salida se nombra un control para llevar un orden en el servomotor.

Para Prac4.vhd, se define un divisor de frecuencia.

Para PWM.vhd, se define una máquina de estados para determinar la dirección del servomotor.

Se llaman a los componentes definidos en cada programa.

Se define cada una de las acciones dentro del servomotor (direcciones).

PWM.vhd

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.STD_LOGIC_ARITH.ALL;
4   use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6   entity PWM is
7   Port ( Reloj : in STD_LOGIC;
8   D : in   STD_LOGIC_VECTOR (7 downto 0);
9   S : out  STD_LOGIC);
10  end PWM;
11
12  architecture arqPWM of PWM is
13  begin
14     process (Reloj)
15     variable Cuenta : integer range 0 to 255 := 0;
16        begin
17        if Reloj='1' and Reloj'event then
18        Cuenta := (Cuenta + 1) mod 256;
19           if Cuenta < D then
20              S <= '1';
21           else
22              S <= '0';
23           end if;
24        end if;
25     end process;
26  end arqPWM;
27
```

Se definen las entradas y salidas

Dentro del divisor de frecuencia, se define la dirección que tomará el servomotor.

Prac4.vhd

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.STD_LOGIC_ARITH.ALL;
4   use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6   entity Prac4 is
7   Port ( clk : in std_logic; div_clk : out std_logic);
8   end Prac4;
9
10  architecture arqPrac4 of Prac4 is begin
11
12     process (clk)
13        constant N : integer := 11;
14        variable cuenta: std_logic_vector (27 downto 0) := x"0000000";
15        begin
16        if rising_edge (clk) then
17           cuenta := cuenta + 1;
18        end if;
19        div_clk <= cuenta (N);
20     end process;
21  end arqPrac4;
```

Se define el proceso para el divisor de frecuencia.

Para el funcionamiento adecuado del prorama, se define como Top-Level entity al programa "Servomotor.vhd". Una vez hecho lo anterior, se compila y se definen los pines correspondientes para las entradas y salidas.

in clk	Input	PIN_N14
out control	Output	PIN_W5
in Dec	Input	PIN_C10
in Inc	Input	PIN_C11
in Pfin	Input	PIN_B14
in Pini	Input	PIN_F15

Actividad 2

Para esta actividad, se implementaron dos servomotores que se movieran en sentido opuesto al mismo tiempo. Para ello, se desarrolló un nuevo programa:

CtrlDobl.vhd

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.STD_LOGIC_ARITH.ALL;
4   use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6   entity CtrlDobl is
7   Port ( clk : in   STD_LOGIC;
8    PiniD : in     STD_LOGIC;
9    PfinD: in STD_LOGIC;
10   IncD : in   STD_LOGIC;
11   DecD : in   STD_LOGIC;
12   control1 : out STD_LOGIC;
13  control2 : out STD_LOGIC);
14   end CtrlDobl;
15
16  architecture arqCtrlDobl of CtrlDobl is
17
18
19      component Servomotor is
20      Port ( clk : in   STD_LOGIC;
21             Pini : in  STD_LOGIC;
22             Pfin : in STD_LOGIC;
23             Inc : in   STD_LOGIC;
24             Dec : in   STD_LOGIC;
25             control: out STD_LOGIC);
26      end component;
27
28      component Prac4 is
29      Port ( clk : in std_logic;
30      div_clk : out std_logic);
```

Se implementan las mismas variables que el programa Servomotor.vhd, donde habrá dos señales de control para cada uno de los servomotores.

Se llama al programa Servomotor donde se extraen sus variables definidas en su entidad.

```
31      end component;
32
33      component PWM is
34      Port ( Reloj : in STD_LOGIC;
35         D : in   STD_LOGIC_VECTOR (7 downto 0);
36         S : out     STD_LOGIC);
37      end component;
38
39      signal reloj : STD_LOGIC;
40      signal ancho1 : STD_LOGIC_VECTOR (7 downto 0) := X"0F";
41      signal ancho2 : STD_LOGIC_VECTOR (7 downto 0) := X"0F";
42         begin
43         --U1: Prac4 port map (clk, reloj);
44         --U2: PWM port map (reloj, ancho1, control1);
45         --U3: PWM port map (reloj, ancho2, control2);
46           S1: Servomotor port map (clk, PiniD,PfinD,IncD,DecD,control1);
47           S2: Servomotor port map (clk, PfinD,PiniD,DecD,IncD,control2);
48  end arqCtrlDobl;
```

Las señales de salidas, para que se muevan simultáneamente y en sentido contrario, se define de la siguiente manera.

Actividad 3

Ahora, ambos motores se deberán mover simultáneamente y en el mismo sentido, para ello, se realiza los cambios del código anterior:

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.STD_LOGIC_ARITH.ALL;
4   use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6   entity CtrlDobl is
7   Port ( clk : in   STD_LOGIC;
8   PiniD : in     STD_LOGIC;
9   PfinD: in STD_LOGIC;
10  IncD : in    STD_LOGIC;
11  DecD : in    STD_LOGIC;
12  PiniD2 : in    STD_LOGIC;
13  PfinD2: in STD_LOGIC;
14  IncD2 : in   STD_LOGIC;
15  DecD2 : in   STD_LOGIC;
16  control1 : out STD_LOGIC;
17  control2 : out STD_LOGIC);
18  end CtrlDobl;
19
20  architecture arqCtrlDobl of CtrlDobl is
21
22
23      component Servomotor is
24         Port ( clk : in   STD_LOGIC;
25                Pini : in  STD_LOGIC;
26                Pfin : in STD_LOGIC;
27                Inc : in   STD_LOGIC;
28                Dec : in   STD_LOGIC;
29                control: out STD_LOGIC);
30      end component;
```

Se implementan las mismas variables que el programa Servomotor.vhd, donde habrá dos señales de control para cada uno de los servomotores.

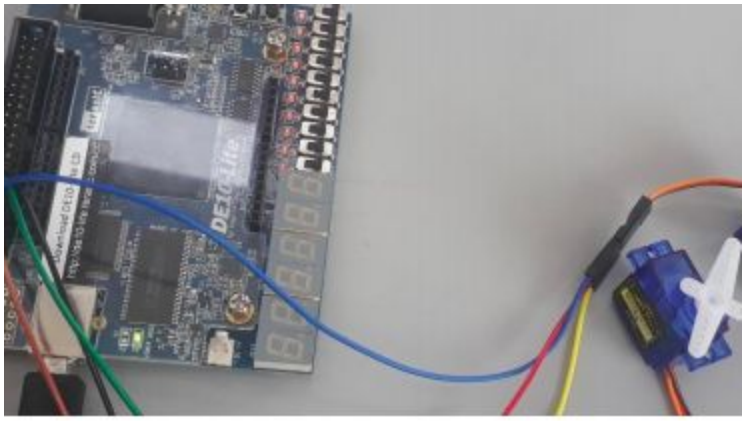
Se llama al programa Servomotor donde se extraen sus variables definidas en su entidad.

```
32      component Prac4 is
33         Port ( clk : in std_logic;
34            div_clk : out std_logic);
35      end component;
36
37      component PWM is
38         Port ( Reloj : in STD_LOGIC;
39            D : in   STD_LOGIC_VECTOR (7 downto 0);
40            S : out     STD_LOGIC);
41      end component;
42
43      signal reloj : STD_LOGIC;
44      signal ancho1 : STD_LOGIC_VECTOR (7 downto 0) := X"0F";
45      signal ancho2 : STD_LOGIC_VECTOR (7 downto 0) := X"0F";
46         begin
47          --U1: Prac4 port map (clk, reloj);
48          --U2: PWM port map (reloj, ancho1, control1);
49          --U3: PWM port map (reloj, ancho2, control2);
50          --S1: Servomotor port map (clk, PiniD,PfinD,IncD,DecD,control1);
51          --S2: Servomotor port map (clk, PfinD,PiniD,DecD,IncD,control2);
52          S1: Servomotor port map (clk, PiniD,PfinD,IncD,DecD,control1);
53          S2: Servomotor port map (clk, PfinD2,PiniD2,IncD2,DecD2,control2);
54  end arqCtrlDobl;
```

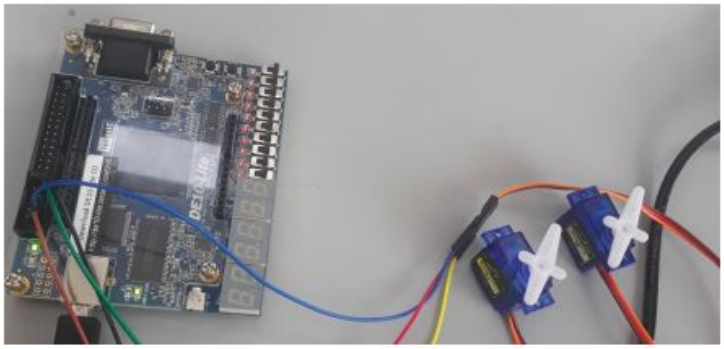
Las señales de salidas, para que se muevan simultáneamente y en sentido contrario, se define de la siguiente manera.

Salidas de las Actividades:

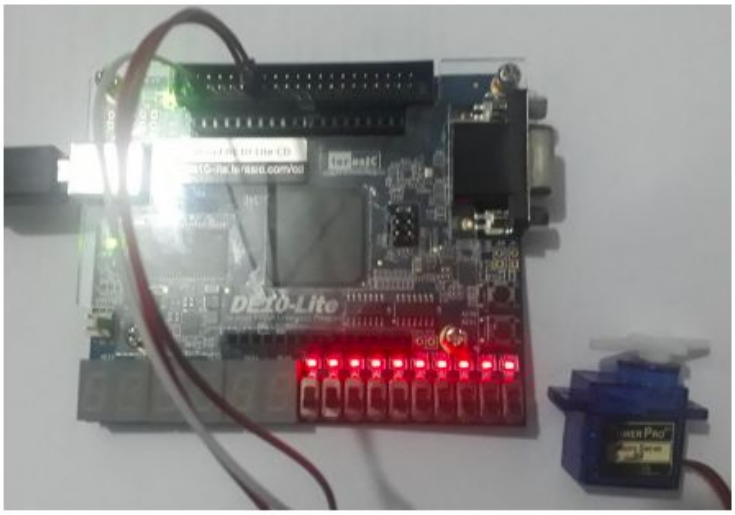
Actividad 1:



Actividad 2:



Actividad 3:



Conclusiones:

En esta práctica aprendí a manejar de manera modular y a separar en diferentes archivos mi programa en VHDL, aprendi a generar mi propia biblioteca de módulos funcionales para posteriormente reutilizar mis modulos.