Práctica 4.

DISEÑO DEL CONTROL DE SERVOMOTORES

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:

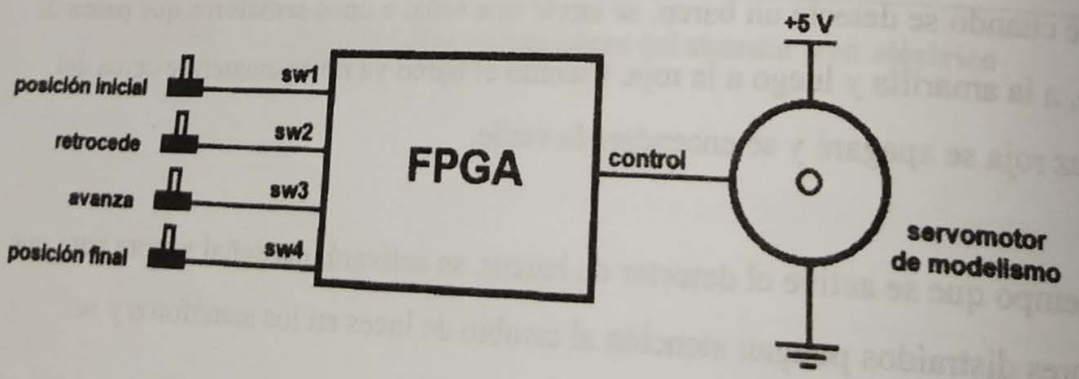

Figura 4.1. Diagrama de bloques del control de un servomotor de modelismo

En la elaboración de un proyecto basado en un FPGA, comúnmente se desarrollan gran cantidad de módulos funcionales para manejar las tareas necesarias en esa aplicación. Una buena práctica de diseño es la de utilizar cada uno de esos módulos de manera

independiente, ya que esto simplifica el proceso de diseño y permite distribuir las diferentes tareas entre varios grupos de trabajo. Además, si se hace una buena división de tareas, al final se contará con un conjunto de módulos funcionales que eventualmente podrán ser reutilizados en otros proyectos. De esta manera, al aplicar esta metodología de diseño, el alumno podrá ir construyendo su propia biblioteca de módulos funcionales, lo que en el futuro le permitirá reducir los tiempos de diseño al reutilizar estos módulos. Esto implica que cada módulo funcional debe estar contenido en un archivo diferente.

Para el desarrollo de esta práctica se aplicará este concepto de división en módulos funcionales, cada uno de ellos contenidos en un archivo diferente, que posteriormente son integrados en un solo proyecto al ser instanciados en el módulo principal. La figura 4.2 muestra los bloques funcionales que componen al control de servomotor.

BLOQUES FUNCIONALES:

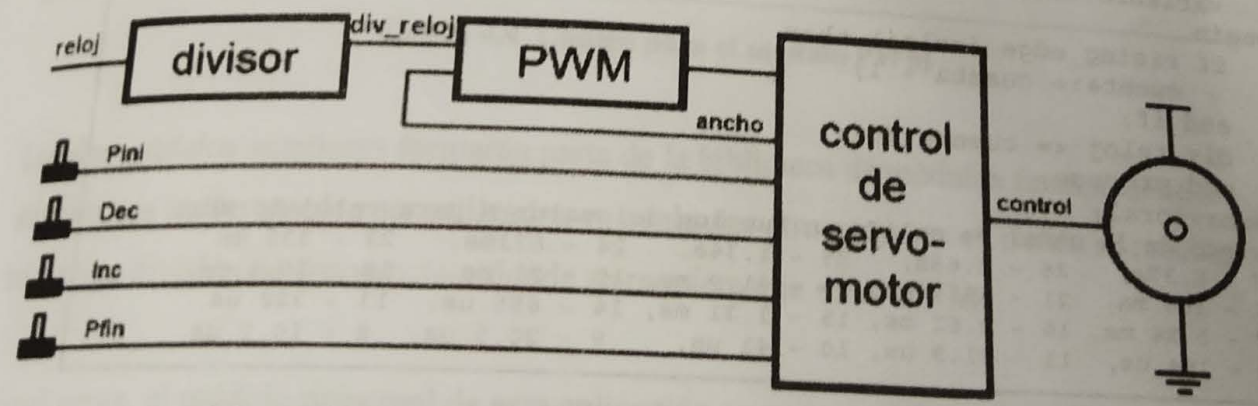

Figura 4.2. Bloques funcionales del control de servomotor

Para la elaboración de este proyecto, se diseñarán dos módulos funcionales de aplicación genérica, el módulo Divisor y el módulo PWM, que podrán ser los dos primeros módulos funcionales de la biblioteca del alumno, además del módulo principal dedicado a la aplicación específica del control del servomotor controlado por cuatro interruptores, en donde se instanciarán los dos módulos de uso general.

ACTIVIDADES COMPLEMENTARIAS:

1. Siguiendo la metodología de diseño presentada, el alumno elaborará un módulo funcional genérico para controlar un servomotor de modelismo, que complementará la biblioteca de módulos del alumno.

2. Utilizando el módulo genérico para controlar un servomotor diseñado en el punto anterior, construir un sistema que haga que dos servomotores de modelismo se muevan de forma complementaria, es decir, se moverán de la misma forma, pero girando en la dirección opuesta.

3. Utilizando el módulo genérico para controlar un servomotor diseñado en el primer punto, construir un sistema que haga que dos servomotores de modelismo se muevan independientemente, cada uno de ellos controlado por dos interruptores de presión tipo *push-boton*, que al presionarlos harán avanzar o retroceder al eje del motor.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity PWM is
    Port ( reloj_pwm : in  STD_LOGIC;
           D : in  STD_LOGIC_VECTOR (7 downto 0);
           S : out  STD_LOGIC);

end PWM;

architecture Behavioral of PWM is
begin
    process (reloj_pwm)
        variable cuenta : integer range 0 to 255 := 0;

    begin
        if reloj_pwm ='1' and reloj_pwm 'event then
            cuenta := (cuenta + 1) mod 256;
            if cuenta < D then
                S <= '1';
            else
                S <= '0';
            end if;
        end if;
    end process;
end behavioral;
```

Figura 4.4. Código para el módulo PWM

Los dos módulos anteriores formarán parte de la biblioteca de módulos funcionales del alumno, los cuales pueden ser utilizados en cualquier otro proyecto en donde se requiera hacer una división de frecuencia o donde se requiera una señal PWM.

Finalmente, el módulo principal de esta aplicación se encargará de detectar la actividad en los interruptores y a partir de ello definir el ciclo de trabajo de la señal PWM. Hay que recordar que en un servomotor de modelismo típico se requiere que la señal de control tenga un período de 20 ms, y que el ancho del pulso varíe en el rango de 1 a 2 ms, en donde el ancho del pulso determina la posición del eje del servomotor; este módulo debe asegurar que esto se cumpla. Por ello se eligió el bit 11 en el divisor de frecuencia, para tener en 256 ciclos aproximadamente los 20 ms. El ancho del pulso de salida variará en el rango de 13 a 24 ciclos para tener el rango de 1 a 2 ms. Con esto el servomotor tendrá 12 posiciones que podrá adoptar en su recorrido. La figura 4.5 muestra el código para el módulo Servomotor, el cual estará contenido en el archivo servomotor.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity servomotor is
    Port ( reloj_sv : in  STD_LOGIC;
           Pini : in  STD_LOGIC;
           Pfin : in  STD_LOGIC;
           Inc : in  STD_LOGIC;
           Dec : in  STD_LOGIC;
           control : out  STD_LOGIC);
end Servomotor;

architecture Behavioral of Servomotor is
  component divisor is
    Port ( reloj : in std_logic;
           div_reloj : out std_logic);
  end component;
  component PWM is
    Port ( reloj_pwm : in  STD_LOGIC;
           D : in  STD_LOGIC_VECTOR (7 downto 0);
           S : out  STD_LOGIC);
  end component;
  signal reloj_serv : STD_LOGIC;
  signal ancho : STD_LOGIC_VECTOR (7 downto 0) := X"0F";
begin
  U1: divisor port map (reloj_sv, reloj_serv);
  U2: PWM port map (reloj_serv, ancho, control);

 process (reloj_serv, Pini, Pfin, Inc, Dec)
   variable valor : STD_LOGIC_VECTOR (7 downto 0) := X"0F";
   variable cuenta : integer range 0 to 1023 := 0;
begin
  if reloj_serv='1' and reloj_serv'event then
    if cuenta>0 then
      cuenta := cuenta -1;
    else
      if Pini='1' then
        valor := X"0D";
      elsif Pfin='1' then
        valor := X"18";
      elsif Inc='1' and valor<X"18" then
        valor := valor + 1;
      elsif Dec='1' and valor>X"0D" then
        valor := valor - 1;
      end if;
      cuenta := 1023;
      ancho <= valor;
    end if;
  end if;
 end process;
end Behavioral;
```

Figura 4.5. Código para el módulo servomotor.

ACTIVIDADES COMPLEMENTARIAS:

1. Siguiendo la metodología de diseño presentada, el alumno elaborará un módulo funcional genérico para controlar un servomotor de modelismo, que complementará la biblioteca de módulos del alumno.

2. Utilizando el módulo genérico para controlar un servomotor diseñado en el punto anterior, construir un sistema que haga que dos servomotores de modelismo se muevan de forma complementaria, es decir, se moverán de la misma forma, pero girando en la dirección opuesta.

3. Utilizando el módulo genérico para controlar un servomotor diseñado en el primer punto, construir un sistema que haga que dos servomotores de modelismo se muevan independientemente, cada uno de ellos controlado por dos interruptores de presión tipo *push-boton*, que al presionarlos harán avanzar o retroceder al eje del motor.