Práctica 5.

DISEÑO DEL CONTROL DE INTENSIDAD EN LEDS

OBJETIVO:

El alumno aprenderá a diseñar módulos con parámetros genéricos, lo que permitirá crear en un proyecto varias instancias de un mismo elemento, pero con diferentes características de operación, con el fin de dar una mayor versatilidad a los módulos diseñados por el alumno.

ESPECIFICACIONES:

Diseñar un circuito utilizando un FPGA que se encargue de controlar el encendido de cuatro LEDs, cada uno de los cuales encenderá con diferente intensidad. La intensidad de cada LED será controlada por medio del ciclo de trabajo de una señal PWM. Las luces en los LEDs irán cambiando siguiendo una secuencia determinada. La figura 5.1 muestra el diagrama del bloque de este sistema.

DIAGRAMA DE BLOQUES:

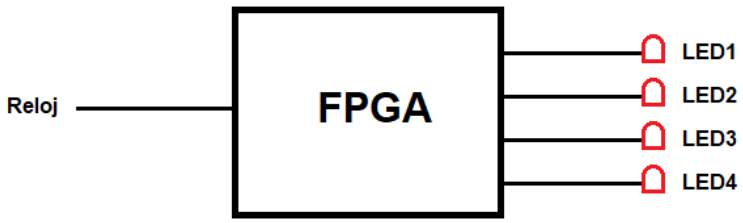

Figura 5.1. Diagrama de bloques del control de intensidad de encendido de LEDs

Al realizar un diseño utilizando un FPGA es común que se requiera tener funcionando concurrentemente varias copias de un mismo objeto, y en muchas ocasiones cada copia del objeto deberá tener características de operación diferente. Por ejemplo, en un mismo diseño se puede requerir utilizar varios registros similares, pero de diferente tamaño. No sería práctico tener en la biblioteca una versión del mismo registro para cada tamaño posible. Lo conveniente en este caso, es tener una sola definición del registro en el que se pueda definir

el tamaño del mismo cuando sea creada una instancia de él. Esto se puede lograr con el uso de parámetros genéricos.

En esta práctica se utilizarán los bloques funcionales diseñados en la práctica anterior, creando varias instancias de cada uno, pero se modificará uno de estos módulos para que utilice un parámetro genérico. En la figura 5.2 muestran los bloques funcionales que integran este diseño.

BLOQUES FUNCIONALES:

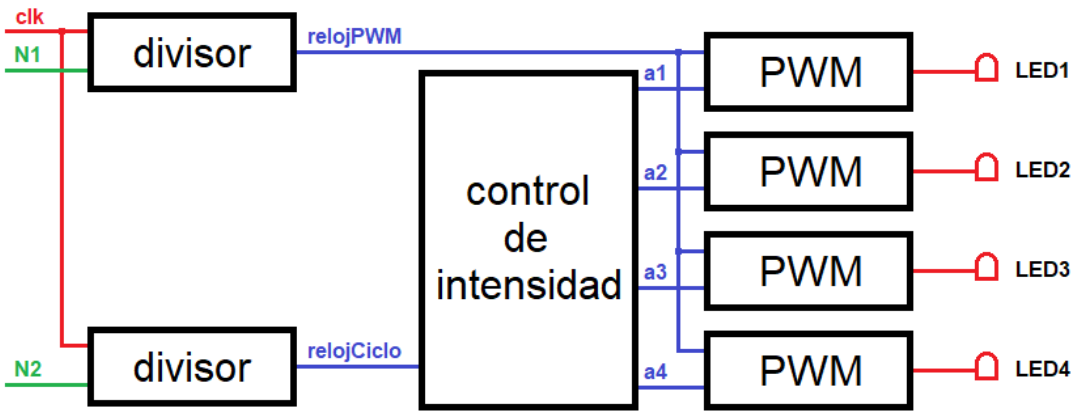

Figura 5.2. Bloques funcionales del control de intensidad de encendido de LEDs

Como se observa en el diagrama, se utilizarán cuatro instancias del módulo PWM y dos del módulo Divisor. Hay que notar que se requiere utilizar dos divisores de frecuencia, ya que se tienen dos procesos que utilizan relojes con frecuencia diferente, uno de frecuencia alta para generar la señal PWM que se utilizará para encender los LEDs, y otro de frecuencia menor que señalará los tiempos en que cambia el estado de la secuencia que se observará en cada LED. La figura 5.3 muestra el código para el módulo Divisor, que estará contenido en el archivo **divisor.vhd**, en el que ha cambiado la definición del valor N, siendo ahora un parámetro genérico en lugar de una constante.

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

entity Divisor is
   Generic ( N : integer := 24);
   Port    ( clk : in std_logic;
             div_clk : out std_logic);
end Divisor;

architecture Behavioral of Divisor is
begin
   process (clk)
      variable cuenta: std_logic_vector (27 downto 0) := X"0000000";
   begin
      if rising_edge (clk) then
         cuenta := cuenta + 1;
      end if;
      div_clk <= cuenta (N);
   end process;
end Behavioral;
-- Periodo de la señal de salida en funcion del valor N para clk=50 MHz:
--  27 ~ 5.37s,   26 ~ 2.68s,   25 ~ 1.34s,   24 ~ 671ms,   23 ~ 336 ms
--  22 ~ 168 ms,  21 ~ 83.9 ms, 20 ~ 41.9 ms, 19 ~ 21 ms,   18 ~ 10.5 ms
--  17 ~ 5.24 ms, 16 ~ 2.62 ms, 15 ~ 1.31 ms, 14 ~ 655 us,  13 ~ 328 us
--  12 ~ 164 us,  11 ~ 81.9 us, 10 ~ 41 us,    9 ~ 20.5 us,  8 ~ 10.2 us
```

Figura 5.3. Código para el módulo divisor.vhd

Para el caso del módulo PWM no es necesario realizar modificación alguna al código de la práctica anterior, por lo que aquí se reutilizará directamente el módulo previamente construido y que forma parte de la biblioteca de módulos del alumno.

Ahora sólo falta construir el módulo principal, el cual se encargará de generar la secuencia que se observará en los LEDs. La figura 5.4 muestra el código para el módulo Leds, el cual irá contenido en el archivo **leds.vhd**. Para probar el funcionamiento de esta práctica se utilizarán los LEDs de la tarjeta de desarrollo.

Es importante notar en el código que para cambiar la asignación de intensidades no es necesario utilizar una variable auxiliar para evitar la pérdida de los valores, ya que aquí se está describiendo hardware.

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

entity Leds is
    Port ( clk : in  STD_LOGIC;
           led1 : out  STD_LOGIC;
           led2 : out  STD_LOGIC;
           led3 : out  STD_LOGIC;
           led4 : out  STD_LOGIC);
end Leds;

architecture Behavioral of Leds is
      component divisor is
            Generic ( N : integer := 24);
            Port ( reloj : in std_logic;
                        div_clk : out std_logic);
      end component;
      component PWM is
            Port ( Reloj : in  STD_LOGIC;
                        D : in  STD_LOGIC_VECTOR (7 downto 0);
                        S : out  STD_LOGIC);
      end component;
      signal relojPWM : STD_LOGIC;
      signal relojCiclo : STD_LOGIC;
      signal a1 : STD_LOGIC_VECTOR (7 downto 0) := X"08";
      signal a2 : STD_LOGIC_VECTOR (7 downto 0) := X"20";
      signal a3 : STD_LOGIC_VECTOR (7 downto 0) := X"60";
      signal a4 : STD_LOGIC_VECTOR (7 downto 0) := X"F8";
begin
      R1: divisor generic map (10) port map (clk, relojPWM);
      R2: divisor generic map (23) port map (clk, relojCiclo);
      P1: PWM port map (relojPWM, a1, led1);
      P2: PWM port map (relojPWM, a2, led2);
      P3: PWM port map (relojPWM, a3, led3);
      P4: PWM port map (relojPWM, a4, led4);

      process (relojCiclo)
            variable Cuenta : integer range 0 to 255 := 0;
      begin
            if relojCiclo='1' and relojCiclo'event then
                  a1 <= a4;
                  a2 <= a1;
                  a3 <= a2;
                  a4 <= a3;
            end if;
      end process;
end Behavioral;
```

Figura 5.4. Código para el módulo leds.vhd

ACTIVIDADES COMPLEMENTARIAS:

1. Hacer que la misma secuencia de 4 LEDs encendidos usada en esta práctica ahora recorra los 8 LEDs de la tarjeta de desarrollo, y al llegar al final ahora vaya de regreso, siempre con el LED de mayor intensidad al inicio y el de menor intensidad al final.

2. Utilizando un LED RGB, controlar la intensidad de encendido de cada color de tal manera que la luz resultante vaya mostrando los colores del arcoíris.