



Universidad Nacional Autónoma de México  
Facultad de Ingeniería  
Ingeniería en Computación



Asignatura: Diseño Digital VLSI

Grupo: 4

Practica 3: Diseño Del Control De Intensidad En Leds.

Profesor: M.I. Alberto Navarrete Hernández

Nombre del alumno: Jiménez Gutiérrez Miguel

## DISEÑO DEL CONTROL DE INTENSIDAD EN LEDS

**Objetivo:** El alumno aprenderá a diseñar módulos con parámetros genéricos, lo que permitirá crear un proyecto con 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.

### Introducción

Un sistema diseñado en VHDL debe simularse y probarse para la funcionalidad antes de que se convierta en hardware. En este pase de simulación, errores de diseño e incompatibilidad de los componentes utilizados en el diseño pueden ser detectados. La simulación de un diseño requiere la generación de datos de prueba y la observación de los resultados de la simulación.

### 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 de los Leds irán cambiando siguiendo una secuencia determinada. La figura 1.1 muestra el diagrama de bloques de este sistema.

#### Diagrama de bloques:

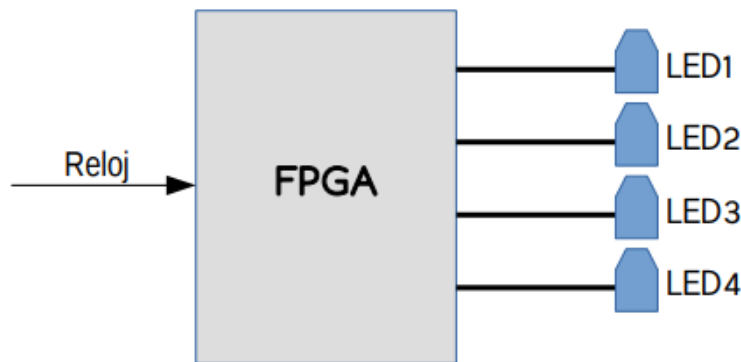


Figura 1.1 Diagrama de bloques del control de intensidad de encendido de leds.

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 prácticas anteriores, creando varias instancias de cada uno, pero se modificará uno de estos módulos para que utilice un parámetro genérico. En la figura 1.2 se muestran los bloques funcionales que integran el diseño.

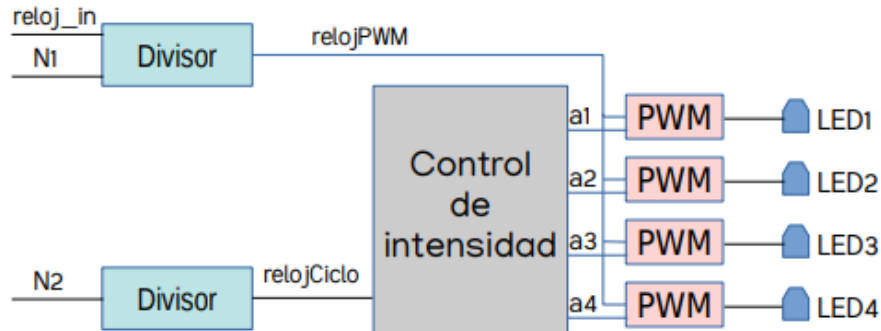


Figura 1.2 Diagrama de bloques funcionales del control de intensidad de encendido de leds.

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 de PWM que se utilizará para encender los Leds, y otro de frecuencia menor que señalará los tiempos en los que cambia el estado de la secuencia que se observará en cada Led. El siguiente código muestra lo correspondiente al módulo del divisor, que estará contenido en el archivo divisor, en el que se tiene el valor N como un parámetro genérico.

## Desarrollo

1.- Realizar una secuencia basada en la práctica realizada, con el uso de 8 leds de la tarjeta de desarrollo, de tal manera que la secuencia al llegar al final vaya de regreso, siempre con el led de mayor intensidad al inicio y el de menor al final.

Código en VHDL

```
Library IEEE;
```

```
Use IEEE.Std_logic_1164.all;
```

```
Use IEEE.Std_logic_arith.all;
```

```
Use IEEE.Std_logic_unsigned.all;
```

```
entity leds is
```

```
Port( reloj: in std_logic;
```

```
led1 : out std_logic;  
led2 : out std_logic;  
led3 : out std_logic;  
led4 : out std_logic;  
led5 : out std_logic;  
led6 : out std_logic;  
led7 : out std_logic;  
led8 : out std_logic);
```

```
end leds;
```

architecture behavioral of leds is

Component PWM is

```
Port( reloj_pwm: in std_logic;  
      D : in std_logic_vector (7 downto 0);  
      S : out std_logic);
```

```
end component;
```

Component divisor is

```
Generic (N : integer := 24);  
Port( reloj: in std_logic;  
      div_reloj : 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"10";
signal a3: std_logic_vector(7 downto 0):= X"18";
signal a4: std_logic_vector(7 downto 0):= X"20";
signal a5: std_logic_vector(7 downto 0):= X"28";
signal a6: std_logic_vector(7 downto 0):= X"30";
signal a7: std_logic_vector(7 downto 0):= X"38";
signal a8: std_logic_vector(7 downto 0):= X"F8";
```

```
begin
```

```
N1: divisor generic map (10) port map (reloj, relojPWM);
```

```
N2: divisor generic map (23) port map (reloj, 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);
```

```
P5: PWM port map (relojPWM, a5, led5);
```

```
P6: PWM port map (relojPWM, a6, led6);
```

```
P7: PWM port map (relojPWM, a7, led7);
```

```
P8: PWM port map (relojPWM, a8, led8);
```

```
process (relojCiclo)
```

```
    variable cuenta : integer range 0 to 255 := 0;
```

```
    variable light : integer range 0 to 16 := 0;
```

begin

if relojCiclo'event and relojCiclo = '1' then

cuenta:=cuenta+1;

light:= light +1;

if light >=0 and light<= 8 then

a1 <= a8;

a2 <= a1;

a3 <= a2;

a4 <= a3;

a5 <= a4;

a6 <= a5;

a7 <= a6;

a8 <= a7;

elsif light>8 and light <=16 then

a8 <= a1;

a7 <= a8;

a6 <= a7;

a5 <= a6;

a4 <= a5;

a3 <= a4;

a2 <= a3;

a1 <= a2;

a8 <= a1;

if light = 16 then

light := 0;

end if;

end if;

end if;













end process;

end behavioral;

The screenshot displays the Quartus Prime IDE interface during the compilation of a VHDL file named 'leds.vhd'. The Project Navigator on the left shows the 'Entity:Instance' hierarchy with 'MAX 10: 10M50DAF484C7G' and 'leds'. The central code editor shows the VHDL code for 'leds.vhd', which includes library declarations, port declarations, and behavioral logic. The bottom status bar displays a message log with the following entries:

Type	ID	Message
Warning	332102	Design is not fully constrained for setup requirements
Warning	332102	Design is not fully constrained for hold requirements
Information		Quartus Prime Timing Analyzer was successful. 0 errors, 5 warnings
Information	293000	Quartus Prime Full Compilation was successful. 0 errors, 13 warnings

Compilación exitosa

Named: *  Edit:  				
Node Name		Direction	Location	I/O Bank
 led1	Output	PIN_A8	7	
 led2	Output	PIN_A9	7	
 led3	Output	PIN_A10	7	
 led4	Output	PIN_B10	7	
 led5	Output	PIN_D13	7	
 led6	Output	PIN_C13	7	
 led7	Output	PIN_E14	7	
 led8	Output	PIN_D14	7	
 reloj	Input	PIN_P11	3	
<<new node>>				

## Asignación de pines

## Conclusión

En esta práctica comprobé la funcionalidad de los parámetros genéricos, sirven principalmente para modelar constantes y en este caso tiempo, el programa recorre led por led, en un ciclo que contiene los estados de encendido.

<https://drive.google.com/drive/folders/135ftdJNc8VushYewfTAdNY7yktMu978D?usp=sharing>

**Referencia:**

Navabi, Z. (2007). VHDL: Modular design and synthesis of cores and systems. USA: McGraw-Hill