

# Generación de señal PWM

Tyrone Novillo

Facultad de ingeniería

Universidad de Cuenca

Cuenca, Ecuador

tyrone.novillo@ucuenca.edu.ec

**Resumen**—The following report presents the development of a project for controlling the intensity of illumination of an LED on the Basys 3 board. The project utilizes Pulse Width Modulation (PWM) to vary the LED intensity based on the position of 8 switches on the FPGA. Each switch represents a different intensity level, allowing for 8 distinct brightness levels. The PWM functionality involves altering the pulse width of a square wave at regular intervals. For instance, a 50 % duty cycle means the LED is on for half the time and off for the other half, creating intermediate brightness. When the duty cycle is 90 %, the LED is continuously on (maximum intensity), and at 0 %, it is barely on (minimum intensity). Additionally, guard frequencies are employed to ensure the PWM signal never reaches its maximum value or goes to zero. The methodology includes utilizing a 100 MHz frequency counter to generate the PWM signal and employing a case-based method to control PWM values based on switch positions.

**Keywords**— PWM, LED intensity control, FPGA, Basys 3, frequency counter, duty cycle, case-based control.

## I. INTRODUCCIÓN

VHDL (VHSIC Hardware Description Language), es un lenguaje de descripción de hardware ampliamente utilizado en el diseño y la simulación de circuitos digitales. VHDL permite describir la estructura, el comportamiento y la funcionalidad de sistemas digitales complejos, como controladores, procesadores, y periféricos, a nivel de abstracción alto. Con VHDL, es posible modelar circuitos digitales de manera precisa y eficiente, facilitando la verificación y el desarrollo de sistemas electrónicos.

La modulación por ancho de pulsos (PWM) es una técnica que se utiliza para controlar la intensidad de una señal eléctrica, como por ejemplo el voltaje, mediante el uso de señales discretas. Consiste en variar el ancho de los pulsos de una señal cuadrada a intervalos regulares, donde la relación entre el tiempo en que la señal está en alto y en bajo determina el nivel de intensidad o voltaje aplicado a un dispositivo.

En esta práctica, se busca implementar un proyecto en la placa Basys 3 para controlar la intensidad de iluminación de un LED utilizando la técnica de PWM. Esto se logrará mediante la generación de una señal PWM que varíe en el tiempo de acuerdo a la posición de 8 switches en la FPGA, permitiendo así tener 8 niveles diferentes de brillo para el LED.

## II. DESARROLLO

Se requiere generar un proyecto para controlar la intensidad de la iluminación de un LED de la placa Basys 3.

Para ello, es necesario generar una señal PWM (pulse width modulation) que varíe en el tiempo y que esté determinada por la posición de 8 switches en la FPGA. Cada switch representará un nivel de intensidad distinto, lo que permitirá tener 8 niveles de brillo diferentes para el LED.

El funcionamiento del PWM se basa en cambiar la anchura del pulso de una señal cuadrada a intervalos regulares. En términos

simples, si se trabaja en un ciclo de trabajo del 50 %, el LED estará encendido la mitad del tiempo y apagado la otra mitad, creando así un brillo intermedio. Cuando el ciclo de trabajo es del 100 %, el LED estará siempre encendido (máxima intensidad), y cuando es del 0 %, estará siempre apagado. La Figura 1 explica visualmente como es posible obtener un voltaje variable a partir de una señal discreta.

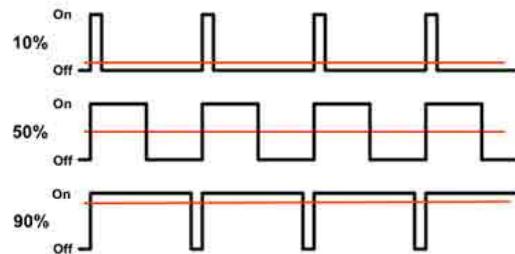


Figura 1: Señal PWM

Adicional a esto, se trabajará con frecuencias de guardia de subida y bajada. Esto significa que la señal PWM nunca alcanzará su valor máximo (el LED no estará siempre encendido) y tampoco será cero (el LED no estará siempre apagado).

Previo a la implementación, es necesario definir los puertos de entrada y salida dentro de la entidad llamada `pwm`.

### II-A. Declaración de la entidad

Para describir la funcionalidad de la creación de la señal `pwm`, se ha creado el archivo de diseño `pwm.vhd`. En este diseño, se define la entidad `pwm` que representa el componente y describe sus puertos de entrada y salida.

La entidad `pwm` tiene los siguientes puertos:

- `inp_switches`: Entradas de los switches de la placa Basys.
- `clk_in`: Entrada de reloj proporcionado por la placa Basys 3 de 100MHz.
- `voltaje_regulado`: Salida conectada al LED de la placa.

Además, se define el entero que define la frecuencia de muestreo para hacer la división de frecuencia. En este caso, se trabajará con tasa de 20ms. Por lo que el contador que se definirá posteriormente, se reseteará cuando llegue a este valor de `1_000_000`.

### II-B. Definición de la arquitectura

Previo a empezar la arquitectura, se definen las siguientes señales intermedias:

- `clk_temp_ms`: Esta señal corresponde a la de 20ms generada a partir de 100MHz.
- `pwm`: Esta es la señal controlada y que se aplicará al LED.
- `contador_ms`: Este es el contador que permite dividir la frecuencia de 100MHz.

**II-B1. Generación de señales de reloj:** Para obtener la señal de 20ms a partir de la de 100MHz, se trabajará con los contadores definidos previamente. Cuando se detecta un flanco de subida en esta señal, el contador `contador_ms` se incrementa en uno si es menor que el divisor establecido `divisor_ms`. Cuando el contador alcanza el valor del divisor, se invierte el valor de la señal `clk_temp_ms` (generando así una señal cuadrada de frecuencia reducida), y el contador se restablece a cero para comenzar de nuevo el conteo de milisegundos.

**II-B2. Generación de señal PWM:** Una vez definida la frecuencia para generar una señal PWM con frecuencias de guardia al inicio y al final se utiliza un bloque `case` para determinar el ciclo de trabajo de la señal PWM (`pwm`) según la configuración de los switches. Cada combinación de switches corresponde a un nivel de intensidad de la señal PWM. Por ejemplo, cuando todos los switches están apagados (00000000), se alternará la señal PWM cada 100\_000 ciclos de reloj hasta llegar a 900\_000 ciclos (evitando llegar a la frecuencia final de guardia), lo que representa un ciclo de trabajo del 10 %. Este proceso se repite para diferentes combinaciones de switches, ajustando el ciclo de trabajo y las frecuencias de guardia según sea necesario para obtener los niveles deseados de brillo en la señal PWM.

Finalmente, la señal PWM se asigna al puerto de salida `voltaje_regulado`, el cual está conectado al LED cuya intensidad se desea controlar.

### II-C. Constraints

El archivo `constraints.xdc` proporciona las instrucciones necesarias para establecer la conexión entre los pines físicos de la tarjeta Basys 3 y las entradas y salidas definidas en la entidad VHDL. Dentro del archivo, se realizan las siguientes configuraciones.

Primero, se define la señal de reloj (`clk_in`) especificando su ubicación física en el pin W5 y su estándar de voltaje LVCMS33. Además, se establece la frecuencia de la señal de reloj en 10.00 unidades de tiempo y su forma de onda, que inicia en 0 y alcanza 5 para representar el ciclo de la señal.

Luego, se asignan los pines correspondientes a los switches definidos en la entidad (`inp_switches`) en la FPGA Basys 3, donde cada switch está asociado a un pin específico según su posición en el arreglo (`inp_switches[0]` a `inp_switches[7]`). Estos pines se configuran con el estándar de voltaje LVCMS33 para la entrada de señales digitales.

Finalmente, se asigna el pin U16 para la salida `voltaje_regulado` conectado al primer LED de la placa Basys 3.

## III. CONCLUSIONES

Se ha empleado un método basado en un contador de frecuencia de 100 MHz para generar la señal PWM. Este contador se divide de manera que cada vez que se alcanza una frecuencia determinada (20 ms en este caso), se genera un pulso de reloj que permite controlar la anchura de los pulsos de la señal PWM, lo que a su vez regula la intensidad luminosa del LED.

Para determinar los valores de la señal PWM en función de la posición de 8 switches en la FPGA se utilizó un método basado en una estructura de control `case`. Cada combinación de switches representa un nivel de intensidad diferente para la señal PWM, lo que permite obtener 8 niveles de brillo distintos en el LED. Además, se implementan frecuencias de guardia al inicio y al final de la señal PWM para asegurar que nunca alcance su valor máximo ni sea cero.

Estos métodos combinados permiten lograr un control manual de la intensidad de iluminación del LED mediante la señal PWM.

## IV. ANEXOS

### IV-A. Archivo `contador.vhd`

---

-- Carrera: Ingeniería en Telecomunicaciones

```
-- Instituto: Universidad de Cuenca
-- Autor: Tyrone Novillo
-- Nombre del diseño: pwm
-----

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity pwm is
generic (
divisor_ms : integer := 1_000_000 --para CK 1 [ms]
);
Port (
inp_switches: in std_logic_vector(7 downto 0);
-- Entrada del clock de 100MHz
clk_in : in std_logic;
voltaje_regulado : out std_logic
);
end pwm;

architecture Behavioral of pwm is
-- Contador para dividir el clock
signal clk_temp_ms : std_logic;
signal pwm : std_logic ;
-- Señal de 1Hz para el LED
signal contador_ms : integer := 0;
begin
process (clk_in)
begin
if rising_edge(clk_in) then
if contador_ms < divisor_ms then
contador_ms <= contador_ms + 1;
else
clk_temp_ms <= not clk_temp_ms;
contador_ms <= 0;
end if;
end if;
case inp_switches is
when "00000000" =>
if contador_ms = 100_000 then
pwm <= not pwm;
elsif contador_ms = 1_000_000 then
pwm <= not pwm;
end if;

when "00000001" =>
if contador_ms = 200_000 then
pwm <= not pwm;
elsif contador_ms = 1_000_000 then
pwm <= not pwm;
end if ;

when "00000011" =>
if contador_ms = 300_000 then
pwm <= not pwm;
elsif contador_ms = 1_000_000 then
pwm <= not pwm;
end if ;

when "00000111" =>
if contador_ms = 400_000 then
pwm <= not pwm;
elsif contador_ms = 50_000 then
pwm <= not pwm;
end if;
```

```

end if;

when "00001111" =>
if contador_ms = 500_000 then
pwm <= not pwm;
elsif contador_ms = 1_000_000 then
pwm <= not pwm;
end if;

when "00011111" =>
if contador_ms = 600_000 then
pwm <= not pwm;
elsif contador_ms = 1_000_000 then
pwm <= not pwm;
end if;

when "00111111" =>
if contador_ms = 700_000 then
pwm <= not pwm;
elsif contador_ms = 1_000_000 then
pwm <= not pwm;
end if;

when "01111111" =>
if contador_ms = 800_000 then
pwm <= not pwm;
elsif contador_ms = 1_000_000 then
pwm <= not pwm;
end if;

when "11111111" =>
if contador_ms = 900_000 then
pwm <= not pwm;
elsif contador_ms = 1_000_000 then
pwm <= not pwm;
end if;

when others =>
if contador_ms = 100_000 then
pwm <= not pwm;
elsif contador_ms = 1_000_000 then
pwm <= not pwm;
end if;

end case;
end process;

voltaje_regulado <= pwm;

end Behavioral;

```

#### IV-B. Archivo de Constraints Basys-3-Master.xdc

```

# Clock signal
set_property -dict { PACKAGE_PIN W5
    IO_STANDARD LVCMOS33 } [get_ports clk_in]
#create_clock -add -name sys_clk_pin -period
    10.00 -waveform {0 5} [get_ports clk]

# Switches
set_property -dict { PACKAGE_PIN V17
    IO_STANDARD LVCMOS33 } [get_ports
    {inp_switches[0]}]

```

```

set_property -dict { PACKAGE_PIN V16
    IO_STANDARD LVCMOS33 } [get_ports
    {inp_switches[1]}]
set_property -dict { PACKAGE_PIN W16
    IO_STANDARD LVCMOS33 } [get_ports
    {inp_switches[2]}]
set_property -dict { PACKAGE_PIN W17
    IO_STANDARD LVCMOS33 } [get_ports
    {inp_switches[3]}]
set_property -dict { PACKAGE_PIN W15
    IO_STANDARD LVCMOS33 } [get_ports
    {inp_switches[4]}]
set_property -dict { PACKAGE_PIN V15
    IO_STANDARD LVCMOS33 } [get_ports
    {inp_switches[5]}]
set_property -dict { PACKAGE_PIN W14
    IO_STANDARD LVCMOS33 } [get_ports
    {inp_switches[6]}]
set_property -dict { PACKAGE_PIN W13
    IO_STANDARD LVCMOS33 } [get_ports
    {inp_switches[7]}]

# LEDs
set_property -dict { PACKAGE_PIN U16
    IO_STANDARD LVCMOS33 } [get_ports
    {voltaje_regulado}]

## Configuration options, can be used for all
## designs
set_property CONFIG_VOLTAGE 3.3
    [current_design]
set_property CFGBVS VCCO [current_design]

## SPI configuration mode options for QSPI
## boot, can be used for all designs
set_property BITSTREAM.GENERAL.COMPRESS TRUE
    [current_design]
set_property BITSTREAM.CONFIG.CONFIGRATE 33
    [current_design]
set_property CONFIG_MODE SPIx4
    [current_design]

```

#### IV-C. Funcionamiento del reloj en la Basys 3

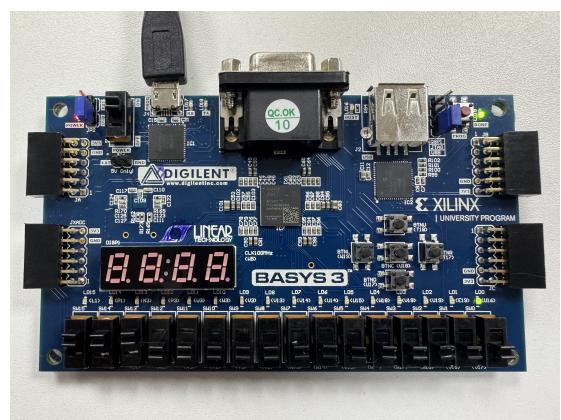


Figura 2: Intensidad mínima (0 switches)

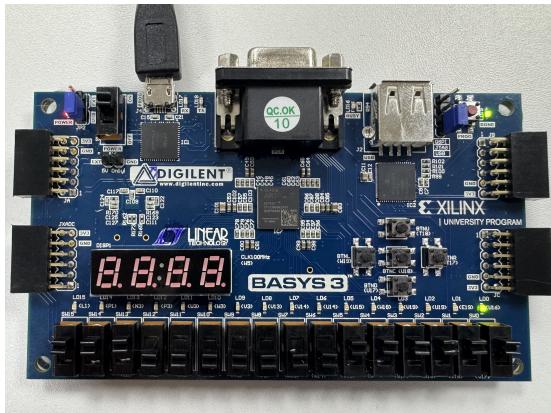


Figura 3: Intensidad media (4 switches)

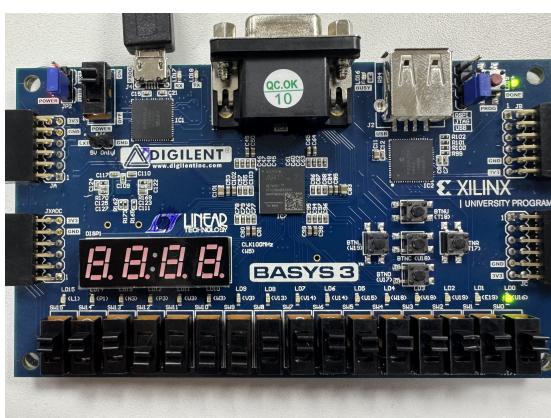


Figura 4: Intensidad máxima (8 switches)