PRÁCTICA 5

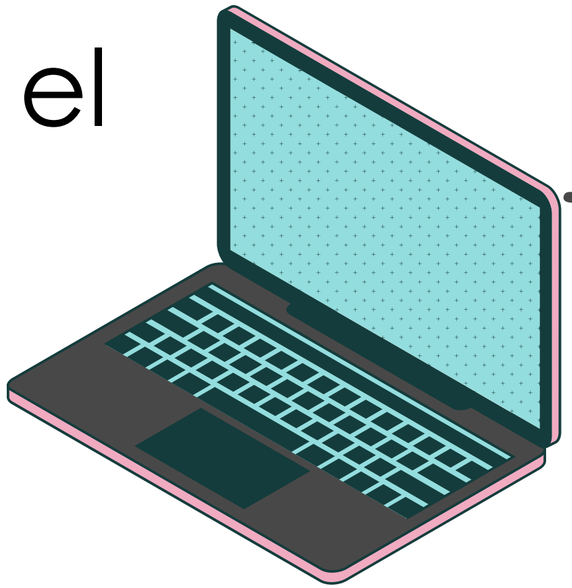
Ejercicio 5.5

Diseño Digital VLSI

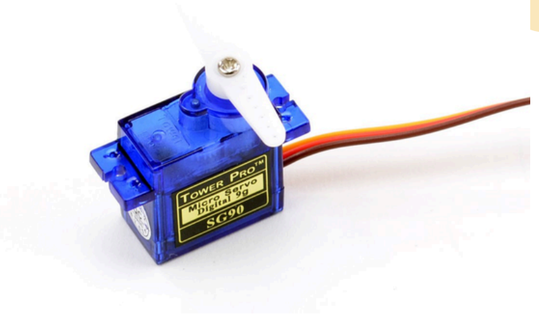
Alcalá Briseño Martha Alondra
Cadena Luna Iván Adrián
Limón Sosa Zair Odin

OBJETIVO

El alumno será capaz de comprender el funcionamiento del PWM y lo aplicará en el servomotor



INSTRUCCIONES



Realizar un proyecto que, dadas las especificaciones, señal de 50Hz y el ciclo de trabajo de 5% y 10% haga que el servomotor se mueva de 5 diferentes formas. En el caso del ejercicio 5.5 que vaya a la posición 0.

ENTIDAD DIVF

```
1   --MODULO DIVF
2   --Reduce la frecuencia de una señal de reloj (clk)
3   --generando una señal más lenta(clkl). Y esto funciona para
4   --trabajar con señales PWL de menor frecuencia o controlar
5   --dispositivos que requieren tiempos especificos
6
7   library ieee;
8   use ieee.std_logic_1164.all;
9
10  --Se define la entidad divf
11  entity divf is
12    generic (num : integer := 25000000);--Define un valor genérico
13            ---num que determina cuántos ciclos de clk se requieren
14            ----para cambiar clkl. CLKL=1Hz
15    port (
16      clk  : in std_logic; --Señal de reloj de alta frecuencia.
17      clkl : buffer std_logic := '0'--Señal de reloj reducida.
18    );
19  end entity;
```

ARQUITECTURA ARQDIVF

```
--Arquitectura "arqdivf"
architecture arqdivf of divf is
  signal conteo : integer range 0 to num;
begin
  process (clk)
  begin ---En cada flanco de subida de clk (rising_edge(clk)),
        -----el contador conteo aumenta.
    if (rising_edge(clk)) then
      if (conteo = num) then--cuandoson iguales se resetea a 0
        conteo <= 0;
        clkl   <= not clkl;
      else --Se invierte la señal clkl (not clkl), cambiándola de 0 a 1 o viceversa.
        conteo <= conteo + 1;
      end if;
    end if;
  end process;
end architecture:
```

ENTIDAD SENAL

```
1  --MODULO SENAL
2  --Genera una señal pwm con un ciclo de trabajo duty cycle
3  --configurable, este modulo es fundamental para controlar
4  --el movimiento de un servomotor de corriente continua
5  library ieee;
6  use ieee.std_logic_1164.all;
7  --Se declara la entidad senal
8  entity senal is
9    port (
10     clk  : in std_logic;  -- Entrada del reloj
11     duty : in integer;--ciclo de trabajo de 0-1000
12     snl  : out std_logic);-- Salida de la señal PWM
13 end entity;
14 --Arquitectura "argsenal"
```

ARQUITECTURA ARQSENAL

```
--Arquitectura "arqsenal"
architecture arqsenal of senal is
  signal conteo : integer range 0 to 1000;--define la resolución de pwm
begin
  process (clk)--GENERACIÓN DE LA SEÑAL PWM
  begin
    if (rising_edge(clk)) then--flaco de de subida
      if (conteo <= duty) then--se comparan
        snl        <= '1';--Si conteo ≤ duty, la salida snl es 1 (pulso alto).
      else
        snl <= '0';--Si conteo > duty, la salida snl es 0 (pulso bajo).
      end if;
      --Esto genera una señal cuadrada donde duty define la proporción del tiempo en alto.


      if (conteo = 1000) then--cuando conteo llega a 1000 se reinicia a 0 completando 1 ciclo
        conteo <= 0;
      else
        conteo <= conteo + 1;--en cada ciclo conteo se incrementa en 1 asegurando la frecuencia estable
      end if;
    end if;
  end process;
end architecture;
```

ENTIDAD PWM

```
--Se declara la entidad pwm
--este parte del código controla el movimiento de un servomotor ajustando el
--"ancho" de pulso en un rango entre 55 y 95, que corresponde a los angulos
--de rotación entre 0° y 180°
entity pwm is
  port (
    clk, rst   : in std_logic; --! Entrada del reloj de la tarjeta y el reset
    dir        : in std_logic; --! Para seleccionar la dirección (0 izquierda, 1 derecha)
    enable_in  : in std_logic; --! Para habilitar el motor
    enable_out : out std_logic; --! Salida de habilitación del L293D
    pwm, npwm  : out std_logic --! Salidas para la señal pwm del L293D
  );
end entity;
```

ARQUITECTURA AQRPWM

```
22    --Arquitectura arqpwm
23    architecture arqpwm of pwm is
31      begin
32      dvfff : entity work.divf(arqdivf) generic map (2500) port map(clk, clkl);
33      -- Entidad para el control del pwm, corresponde a 1/5 del ciclo de dvfff
34      epwm : entity work.senal(arqsenal) port map(clkl, 500, pwm_out);
35    --el 500 representa el ciclo de trabajo de la señal pwm
36
37      -- PROCESO DE ACTUALIZACIÓN DEL ESTADO
38      fsm_update : process (rst, clkl)
39      begin
40        if rst = '0' then --Si rst = 0, se reinicia el estado a STP
41          PS <= STP;
42        elsif rising_edge(clkl) then--En cada flanco de subida del reloj clkl, el estado actual
43       -- (PS) toma el valor del estado siguiente (NS).
44          PS <= NS;
45        end if;
46      end process;
```

LÓGICA PARA EL CAMBIO DE ESTADO

Estado STP

```
49    --PROCESO PARA LA LOGICA DEL CAMBIO DE ESTADO
50    fsm : process (PS, enable_in, dir, pwm_out)
51    begin
52      case PS is
53          -- Estado STP (Stop), el motor no está encendido
54          -- Si enable_in se habilita:
55          --      Si dir es 1, gira a la derecha.
56          --      Si dir es 0, gira a la izquierda.
57        when STP =>
58          if enable_in = '1' and dir = '1' then
59            NS <= RIGHT;
60          elsif enable_in = '1' and dir = '0' then
61            NS <= LEFT;
62          else
63            NS <= STP;
64          end if;
65          pwm        <= '0';
66          npwm       <= '0';
67          enable_out <= '0';
```

Estado LEFT

```
    -- Estado LEFT, el motor gira a la izquierda
    -- Si cambia la dirección, para a RIGHT
    -- Si se deshabilita, regresa a STP
when LEFT =>
    if enable_in = '0' then
        NS <= STP;
    elsif enable_in = '1' and dir = '1' then
        NS <= RIGHT;
    else
        NS <= LEFT;
    end if;

    pwm        <= '0';
    npwm       <= pwm_out;
    enable_out <= '1';
```

Estado RIGHT

```
        -- Estado RIGHT, el motor gira a la derecha
        -- Si cambia la dirección, para a LEFT
        -- Si se deshabilita, regresa a STP
      when RIGHT =>
        if enable_in = '0' then
          NS <= STP;
        elsif enable_in = '1' and dir = '0' then
          NS <= LEFT;
        else
          NS <= RIGHT;
        end if;
        pwm        <= pwm_out;
        npwm       <= '0';
        enable_out <= '1';

      when others =>
        pwm        <= '0';
        npwm       <= '0';
        enable_out <= '0';
        NS         <= STP;
    end case;
  end process;
end architecture;
```

REFERENCIAS

[1] S. autor, "Prácticas de diseño digital VLSI 2022 para tarjetas FPGA Altera-Intel (DE10 LITE)" Facultad de Ingenieria, 29 12 2021. [En línea]. [Último acceso: 03 03 2025]. Utilizado: para recordar objetivo e introducción [pg 34 - 35]

[2] “DE-10 Lite User Manual”, Intel, Sep. 7, 2016. Available: https://ftp.intel.com/Public/Pub/fpgaup/pub/Intel_Material/Boards/DE10 Lite/DE10_Lite_User_Manual.pdf [Último acceso: 03 03 2025] Utilizado: para el uso del GPIO y switch de la tarjeta [pg 4 - 12]

[3] O. Gazi, «Entity, Architecture and VHDL Operators,» de A Tutorial Introduction to VHDL Programming, Turkey, Springer, 2019, p. 254. [Último acceso: 03 03 2025]. Utilizado: en la definicion de entidad, arquitectura y operadores [pg 1 - 25]

THANK YOU