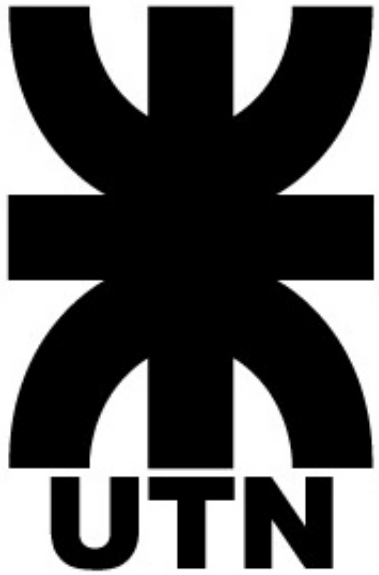

Universidad Tecnológica Nacional
Facultad Regional de Córdoba

Trabajo Práctico De Laboratorio $N°1$

Navarro, Facundo 63809

Curso: 6r4
Grupo $N°5$

Técnicas Digitales IV

Docentes:
Ing. Cayuela, Pablo
Ing. Olmedo, Sergio

6 de junio de 2020

Índice

1. Introducción

El siguiente informe documenta los procedimientos realizados para el diseño digital de un controlador de un motor mediante la lectura de un encoder, se utilizar parte de las prácticas adquiridas en los prácticos de entrenamiento y se profundiza en los conocimientos sobre encoders o codificadores rotativos.

2. Marco Teórico

2.1. Encoder

Un "encoder rotatorio", también llamado codificador del eje o generador de pulsos, es un dispositivo electromecánico capaz de convertir la posición angular de un eje a un código digital. Estos dispositivos suelen ser añadidos a motores de corriente continua para convertir el movimiento mecánico en pulsos digitales que pueden ser interpretado por un sistema electrónico de control.

Encuentran aplicación dentro de los campos de la robótica, lentes fotográficas, aplicaciones industriales que requieran medición angular, etc.

Los motores DC tienen un comportamiento complejo en cuanto a control de posición y de la velocidad, el cual no es lineal y depende mucho de la carga que soporten, es por este motivo que surge la necesidad de la aplicación de un encoder que permita conocer y asegurar la correcta posición del eje.

Según su diseño y funcionalidad existen distintos tipos de encoders. Los tipos más comunes se pueden clasificar en absolutos y relativos.

2.1.1. Encoder Absoluto

Un encoder absoluto es un dispositivo que mide la posición absoluta angular. Esta diseñado para proporcionar un código digital de acuerdo a la posición angular de la flecha, como se ve en la imagen 1, una vuelta esta dividida en un número especifico de divisiones o marcas y cada una de ellas se les asigna físicamente un código digital único, es decir, que tiene un código único para cada posición.

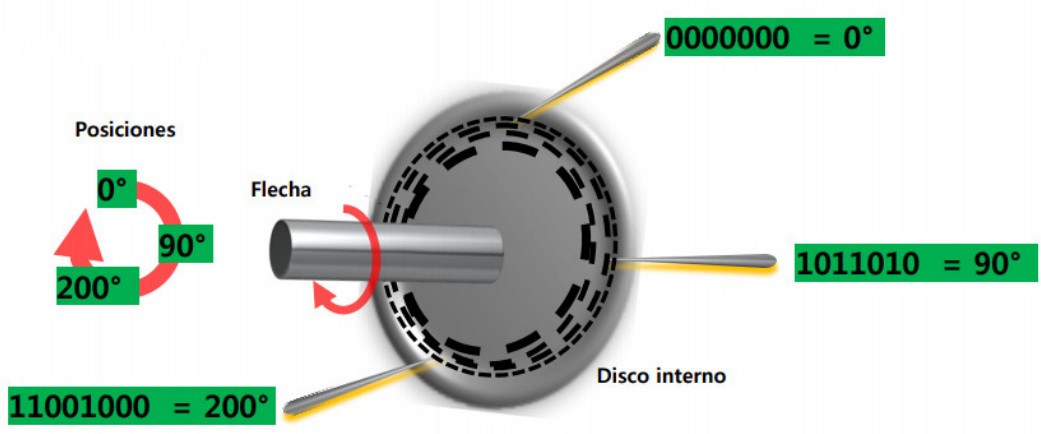

Figura 1: Estructura interna de un encoder absoluto.

2.1.2. Encoder Incremental

El tipo común de encoder incremental consiste de un disco solidario al eje del motor que contiene un patrón de marcas o ranuras que son codificados por un interruptor óptico generando pulsos eléctricos cada vez que el patrón del disco interrumpe y luego permite el paso de luz hacia el interruptor óptico a medida que el disco gira.

Este tipo de encoders determina la posición de rotación de acuerdo a un número específico de pulsos por vuelta (PPV), mediante el conteo de esos pulsos a medida que el encoder gira, es entonces que la PPV es un factor de resolución y es el aspecto más importante a la hora de seleccionar un encoder incremental. La resolución de un encoder típico es del orden de 1000 pulsos por revolución.

Desde un encoder incremental no se puede determinar la posición angular absoluta del eje. Para poder determinar la posición relativa a un punto de referencia (cero), el encoder debe incluir una señal adicional que genera un pulso por revolución, denominada "Indice"(Z).

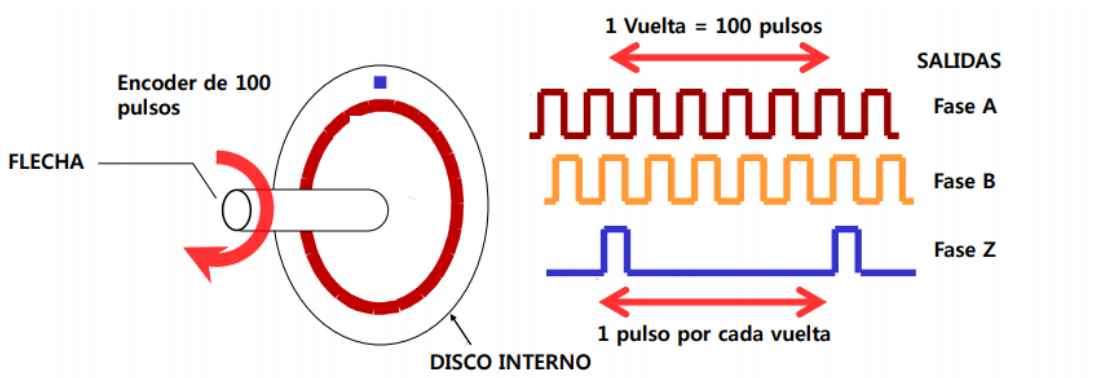

Figura 2: Estructura interna de un encoder incremental.

2.1.3. Encoder en cuadratura

Corresponde a un tipo de encoder incremental que utiliza dos sensores ópticos posicionados con un desplazamiento de 1/4 de ranura el uno del otro, generando dos señales de pulsos digitales desfasadas en 90 grados o en "cuadratura". Estas señales se llaman comúnmente A y B, mediante ellas es posible suministrar los datos de posición, velocidad y dirección de rotación del eje.

Usualmente, si la señal A adelanta a B (la señal A toma un valor lógico de "1.ªntes que B) se establece por convenio que el eje esta rotando en sentido horario, mientras que si B adelanta a A, el sentido es anti horario, como se ve en la figura **3**.

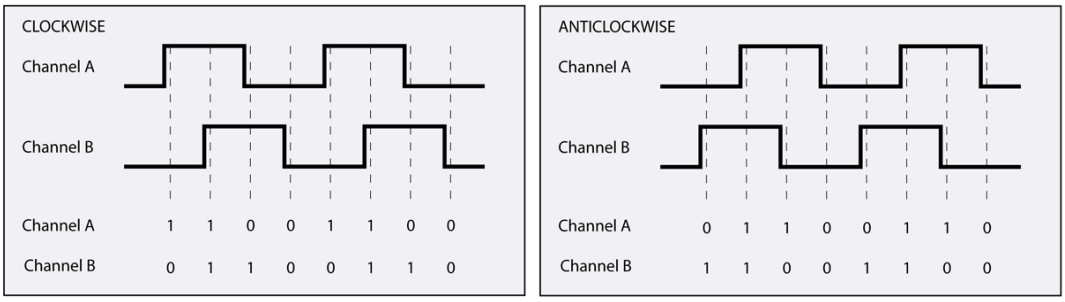

Figura 3: Sentido de giro del encoder en cuadratura

3. Control de lazo cerrado en posición de un motor C.C.

3.1. Consigna

Realizar la descripción de un decodificador de encoder incremental visualizando en un display los pulsos en n_contador_up_down.

Con la cuenta se debe controlar una señal PWM. Esta señal, el ciclo de trabajo debe variar entre 5 % y 95 %. La frecuencia debe ser de 400 Hz.

El ciclo de trabajo al 50 % del PWM, debe suceder cuando la comparación de la cuenta del contador_up_down es igual a la entrada de referencia.

La entrada de referencia, nos dice en que lugar se quiere posicionar el motor, dentro de una revolución indicada por la cuenta desde 0 a 1999 pulsos de encoder.

La entrada debe ser de 11bit, como tambien el contador_up_down.

La señal PWM debe ser maxima para una diferencia ±100 entre la cuenta de los pulsos (contador_up_down) y la de referencia.

3.2. Desarrollo

3.2.1. Estructura del proyecto

El proyecto fue desarrollado en el SW Vivado 2019.2.1, ya que se utilizo la plataforma de entrenamiento de Digilent Basys3.

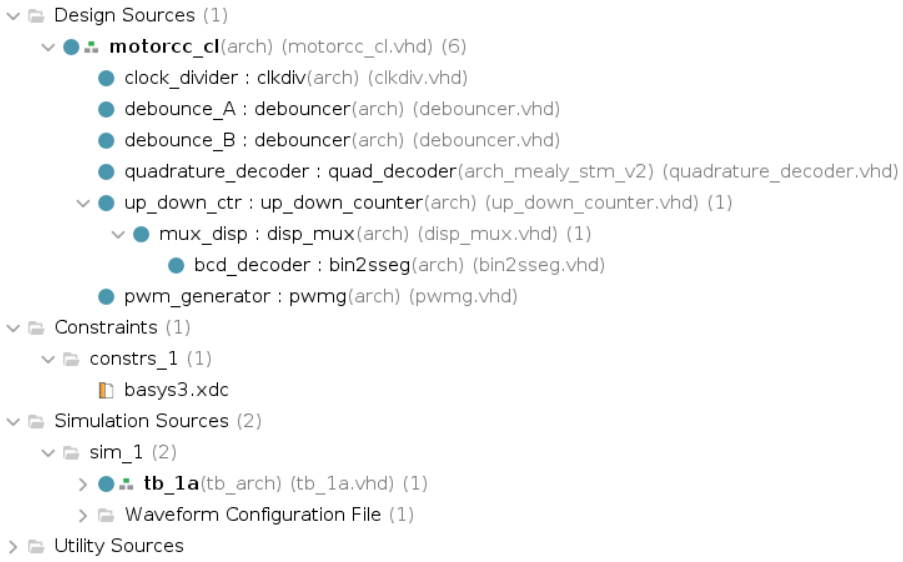

Figura 4: Árbol del proyecto

3.2.2. Debouncer

Siempre existe la posibilidad de ruido externo que se puede introducir dentro de la señal, en este caso se plantea un filtro digital a través de un registro, el cual podría ser parametrizado en caso que se requiera un tiempo mayor para estabilizar la señal.

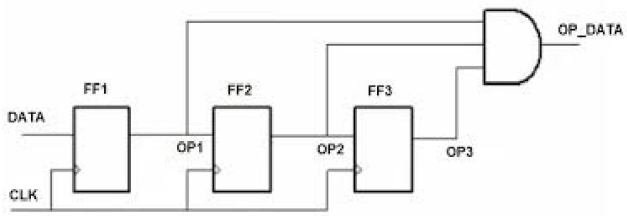

Figura 5: Estructura de filtro digital.

El filtro se utiliza sobre las señales que provienen del encoder, A y B como se aprecia en la imagen **6**.

```
debounce_A : entity work.debouncer(arch)
    port map(
        clk => clk,
        reset => reset,
        data_in => A,
        f_data => A_s
        );

debounce_B : entity work.debouncer(arch)
    port map(
        clk => clk,
        reset => reset,
        data_in => B,
        f_data => B_s
        );
```

Figura 6: Señales filtradas A y B.

3.2.3. Divisor de frecuencia y multiplexado de displays

A partir de la frecuencia de la placa f_1 en 100 MHz, se pretende utilizarla para realizar el multiplexado de los 4 displays necesarios para visualizar la cuenta hasta 5759, como el ojo no es capaz de percibir el encendido y apagado a alta velocidad, fenómeno conocido como persistencia de la visión, engañando al espectador en creer que el los displays se encuentran encendido en simultáneo.

$$f_{mux} = \frac{f_{in}}{2^n} = \frac{100Mhz}{2^{17}} \approx 800Hz \tag{1}$$

Con la frecuencia de 100 Mhz, es necesario un registro de 19 bits de longitud, con los 2 MSB se multiplexa la salida de los display a una velocidad aproximada de 800 Hz.

```
-- 2MSBs controlar el multipleado de los 4 display
sel <= std_logic_vector(q_reg(N-1 downto N-2));
process(sel, bin0, bin1, bin2, bin3)
begin
    case sel is
        when "00" =>
            an <= "1110";
            bin_reg <= bin0;
        when "01" =>
            an <= "1101";
            bin_reg <= bin1;
        when "10" =>
            an <= "1011";
            bin_reg <= bin2;
        when others =>
            an <= "0111";
            bin_reg <= bin3;
```

Figura 7: Multiplexado de displays de 7 segmentos.

3.2.4. Decodificador de cuadratura

El decodificador del encoder en cuadratura se plantea como una máquina de estado que detecta el sentido de acuerdo si la señal A se adelanta a señal B o su inversa. Por cada transición del nivel bajo hacia alto o bajo se produce un pulso. Como se dijo previamente, una características principal de este tipo de encoder es su resolución, el encoder utilizado tiene una resolución de 1440 ppv, la cual se multiplica por 4, correspondiendo a un ciclo entero entre las señales en cuadratura A y B, otorgando una mayor resolución de 5760 ppv, haciendo necesario un contador de módulo 5760.

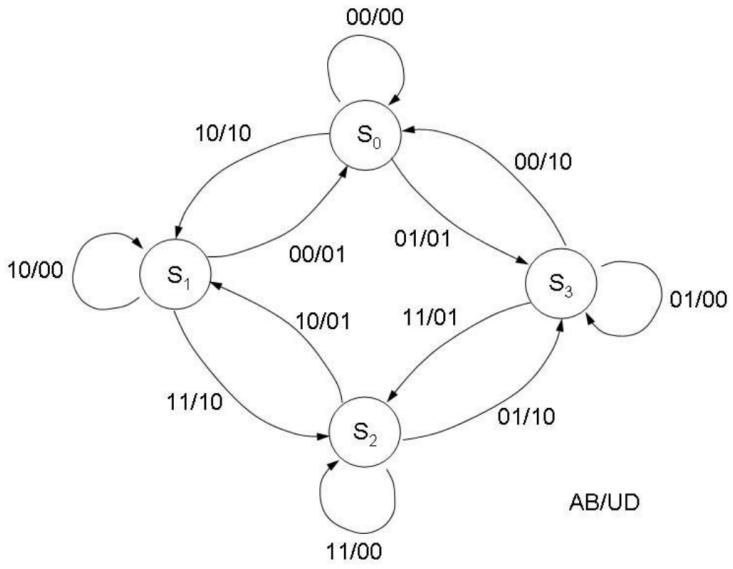

Figura 8: Máquina de Mealy

Los resultados de la máquina de Mealy se observan en la imagen **9**, aclarando que se probaron 3 arquitecturas distintas en la descripción, dos mealy y un combinacional puro.

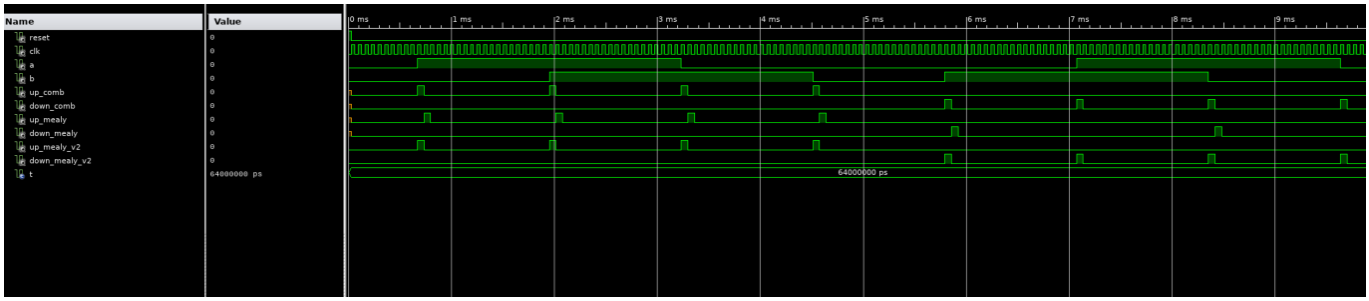

Figura 9: Simulación de detector de cuadratura.

3.2.5. Contador 0 - 5759

El encoder incremental usado se ve en la imagen **10**

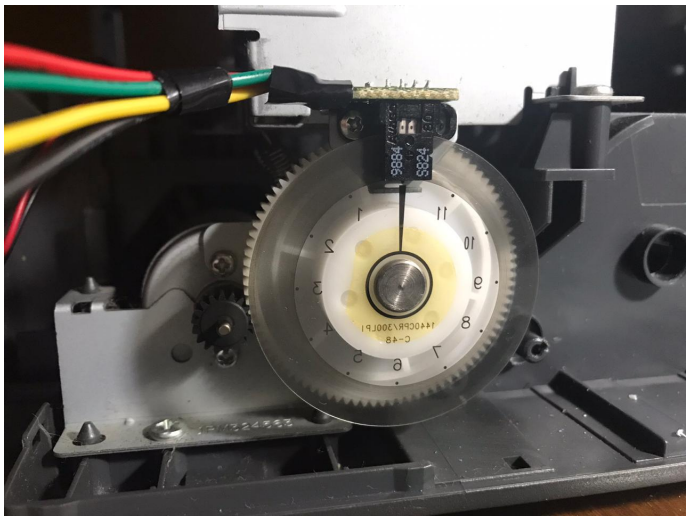

Figura 10: Encoder de 1440 ppv.

Para el contador de módulo 5760, se usan los pulsos que se generan en el detector de cuadratura, de acuerdo si la señal es *up* o *down*, el valor de contador asciende o desciende respectivamente. Como se ve en la figura **11**, si en algún momento se acierta uno de los pulsos y además concuerda con el valor parametrizable M o M_n, el valor de registro se setea a 0.

```
r_next <= (others => '0') when (up='1' and r_reg = (M-1)) or (down='1' and r_reg = M_n+1) else
          r_reg + 1          when up='1'          else
          r_reg - 1          when down='1'        else
          r_reg;
```

Figura 11: Contador módulo 5760.

Para los displays en un principio se orientó la solución utilizando un conversor binario a BCD, el cual consiste en un registro de desplazamiento con una adición de un entero de valor 6 de acuerdo si se cumplen ciertas condiciones, el mismo se implementó y se vio que tenia el resultado correcto, sin embargo se trataba de una estructura que consumía muchos recursos y necesita de muchos pulsos de clock para convertir una entrada serie, a una salida paralela de 12 bit para los 4 displays, caso por el cual se implementa un contador similar al planteado en los ejercicios de entrenamiento (10).

```
if (up='1') then
  if (d3_reg=5) and (d2_reg=7) and (d1_reg = 5) and (d0_reg = 9) then
      d3_next <= "0000";
      d2_next <= "0000";
      d1_next <= "0000";
      d0_next <= "0000";
  else
        if (d0_reg/=9) then
            d0_next <= d0_reg + 1;
        else
            d0_next <= "0000";
            if (d1_reg/=9) then
                d1_next <= d1_reg + 1;
            else
                d1_next <= "0000";
                if (d2_reg/=9) then
                    d2_next <= d2_reg + 1;
                else
                    d2_next <= "0000";
                    if (d3_reg/=9) then
                        d3_next <= d3_reg + 1;
                    else
                        d3_next <= "0000";
                    end if;
                end if;
            end if;
        end if;
  end if;
```

```
elsif (down='1') then
  if (d3_reg=0) and (d2_reg=0) and (d1_reg = 0) and (d0_reg = 0) then
      d3_next <= "0101";
      d2_next <= "0111";
      d1_next <= "0101";
      d0_next <= "1001";
  else
        if (d0_reg/=0) then
            d0_next <= d0_reg - 1;
        else
            d0_next <= "1001";
            if (d1_reg/=0) then
                d1_next <= d1_reg - 1;
            else
                d1_next <= "1001";
                if (d2_reg/=0) then
                    d2_next <= d2_reg - 1;
                else
                    d2_next <= "1001";
                    if (d3_reg/=0) then
                        d3_next <= d3_reg - 1;
                    else
                        d3_next <= "1001";
                    end if;
                end if;
            end if;
        end if;
  end if;
```

Figura 12: Descripción de contador para displays 7 segmentos.

En la imagen **12** se puede apreciar que se utilizan 4 registros de 4 bits (dn_reg) para cada uno de los displays, en el caso de que se acierte un pulso *up*, se incrementa los registros de los displays, con la condición que si el valor previo era 9, el próximo valor posible es 0. Para el caso inverso, con el pulso *down* la lógica es similar, pero en este caso los registros se decrementan, con la excepción que si el valor es 0 el siguiente valor sera 9. Por ultimo se debe notar la condición que si el contador llega a un valor equivalente 5759 y se acierta un pulso up, los registros se reinician, caso contrario si se encuentra en 0 y se acierta un pulso down, el los registros se establecen a 5759.

3.2.6. PWM

Para el control del motor se genera dos señales de PWM en contra fase, las cuales son utilizadas para controlar tanto la velocidad como el sentido de giro del motor. Estas señales son llevadas a una llave H, formada por un integrado L9110S, el comportamiento de la llave con las señales es la siguiente:

- Duty = 50 % : Motor parado en torque.
- Duty > 50 %: Motor gira en sentido horario.
- Duty < 50 %: Motor gira en sentido anti horario.

3.2.6.1. Frecuencia del PWM Como requisito se pedía una frecuencia de 400 Hz. Para ello se debe tener en cuenta la resolución del PWM, como así también la frecuencia del clock. Se elige una resolución de 7 bit para satisfacer la condición de un mínimo de 5 % y un máximo de 95 %.

$$f_{out} = \frac{f_{clk}}{2^{(n+1)}} = \frac{100\ Mhz}{2^18} \approx 381Hz \tag{2}$$

Restando los 7 bit de resolución del PWM se debe elegir el divisor de frecuencia correspondiente a Q(10), como se muestra en la imagen **13**.

```
entity clkdiv is
    Port (
        mclk, reset :   in  std_logic;
        clk_b10     :   out std_logic
     );
end clkdiv;

architecture arch of clkdiv is
    signal  q :   std_logic_vector(16 downto 0);
begin

    process(mclk, reset)
    begin
        if (reset = '1') then
            q <= (others => '0');
        elsif (rising_edge(mclk)) then
            q <= q + 1;
        end if;
    end process;

    clk_b10 <=  q(10);
end arch;
```

Figura 13: Divisor de frecuencia

De esta manera se tiene el clock del PWM, falta ajustar el periodo necesario para llegar a 400 Hz

$$T = \frac{T_{buscada}}{T_{clk\_div}} * 2^n = \frac{2,5\ ms}{2,62\ ms} * 128 \approx 122 \tag{3}$$

```
process(clk, reset)
begin
    if(reset = '1') then
        count <= (others => '0');
    elsif (rising_edge(clk)) then
        if (count = period - 1) then
            count <= (others => '0');
        else
            count <= count + 1;
        end if;
    end if;
end process;

process(clk, count, duty)
begin
    if(rising_edge(clk)) then
        if(count < lim_inf) then
            pwm_next <= '1';
        elsif (count >= lim_sup) then
            pwm_next <= '0';
        else
            if (count < duty) then
                pwm_next <= '1';
            else
                pwm_next <= '0';
            end if;
        end if;
    end if;
end process;

pwm   <= pwm_reg      when reset = '0' else '0';
pwm_n <= not(pwm_reg) when reset = '0' else '0';
```

Figura 14: Descripción de PWM

Finalmente, se pide que si los pasos entre la posición actual del motor y la de referencia difieren en ±100, la velocidad es máxima, esto se consigue con la siguiente descripción.

```
ref_next <=  signed(ref);
pos_next <=  signed(pos);

step_next <=  pos_reg - ref_reg when pos_reg > ref_reg else
              ref_reg - pos_reg when ref_reg > pos_reg else
              (others => '0');

cw <= '1' when pos_reg > ref_reg else
      '0';

acw <= '1' when ref_reg > pos_reg else
      '0';

duty <= "1111010" when cw =  '1' and step_reg > 100 else    -- 100%
        "1011011" when cw =  '1' and step_reg > 50  else    --  75%
        "1000011" when cw =  '1'                    else    --  55%
        "0000000" when acw = '1' and step_reg > 100 else    --   0%
        "0011111" when acw = '1' and step_reg > 50  else    --  25%
        "0110111" when acw = '1'                    else    --  45%
        "0111101";                                          --  50%
```

Figura 15: Seleccion de sentido de giro y duty cycle.

El funcionamiento del código en **15** es el siguiente, *pos* es derivado del módulo contador y *ref* de los switchs y es variable con estas. Dependiendo cual sea mayor a la otra determinara el sentido de giro, ya sea clockwise(cw) o anticlockwise(acw). Por ultimo se busca la magnitud de separación entre ambas posiciones, si es mayor de 100 se elegirá un pwm mas cercano a la máxima velocidad respetando el sentido de giro, caso que la posición actual sea igual a la referencia, tanto cw y acw valen '0' y el duty queda al 50 % (torque).

3.2.7. Simulaciones

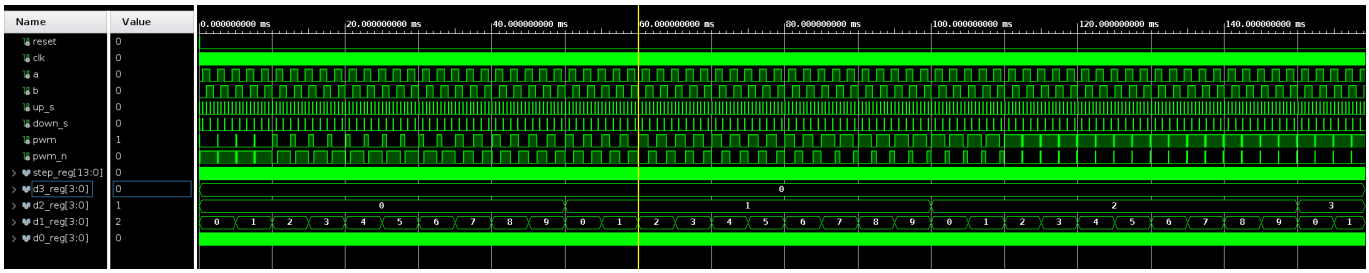

Figura 16: Simulación behavioral, con referencia puesta en 120.

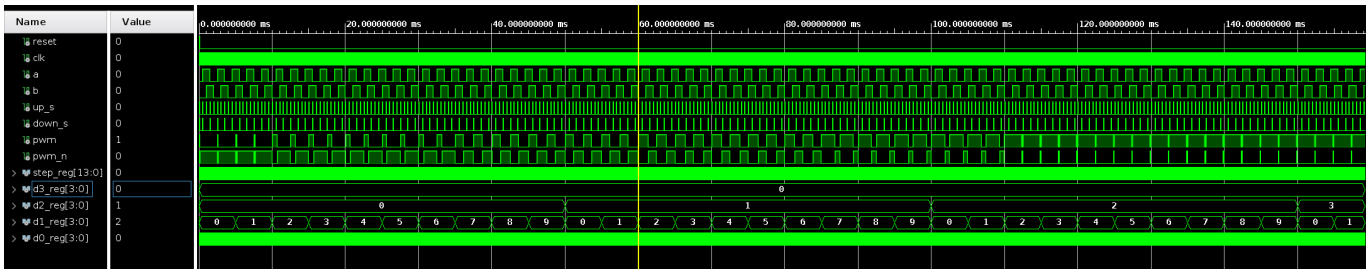

Figura 17: Simulación postfit, con referencia puesta en 120.

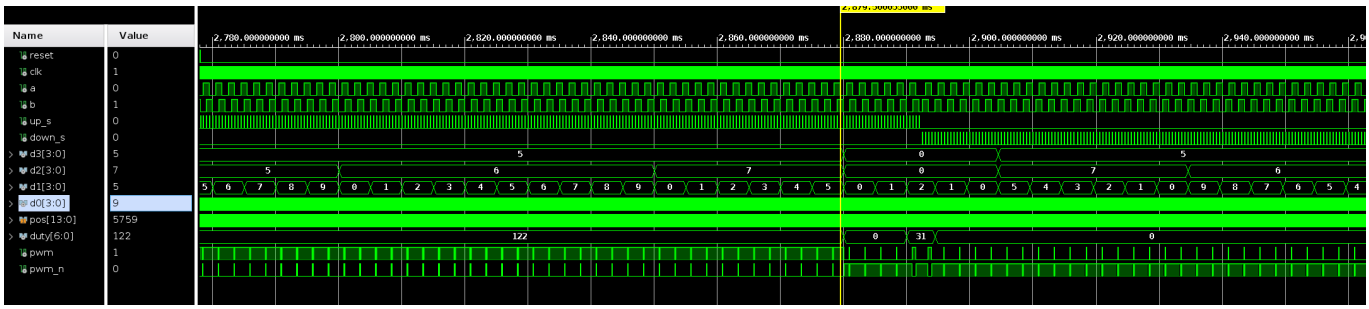

Figura 18: Reinicio contador ascendente modulo 5760

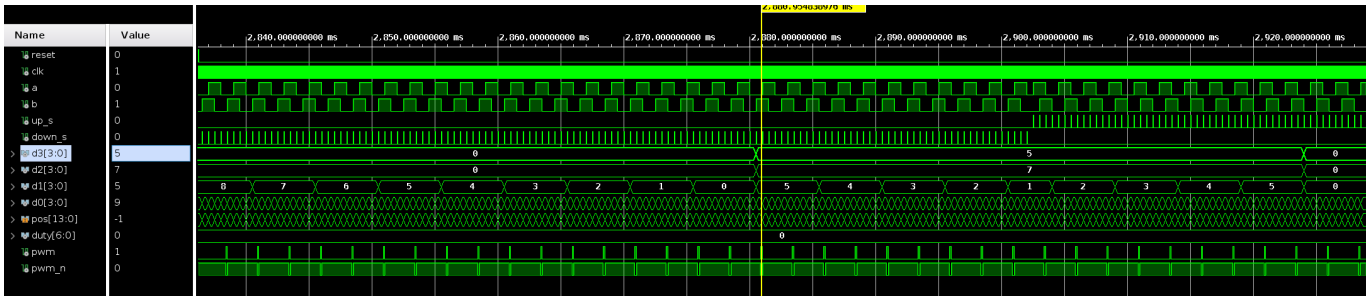

Figura 19: Reinicio contador descendente modulo 5760

3.2.8. Multimedia

Se adjunta un dos links de unos vídeos demostrativos:

- Presentación: `https://youtu.be/4qJg4IgcLNk`
- Demo: `https://youtu.be/wE91018Mobk`

4. Control de lazo de velocidad de un motor C.C.

5. Control de motor paso a paso de velocidad a lazo abierto con rampa de aceleración y desale ración parametrizable