Práctica 6.

DISEÑO DEL CONTROL DE MOTORES A PASOS

OBJETIVO:

El alumno aprenderá a diseñar el controlador de un motor a pasos mediante el uso e implantación de máquinas de estado.

ESPECIFICACIONES:

Diseñar el circuito de control utilizando un FPGA, el cual se encargue de activar un motor a pasos bipolar con 4 líneas de control. Los movimientos que debe realizar el motor son en sentido a las manecillas del reloj, viceversa y detenido por medio de tres botones que controlan estos movimientos.

La figura 6.1 muestra el diagrama a bloques del sistema.

DIAGRAMA DE BLOQUES:

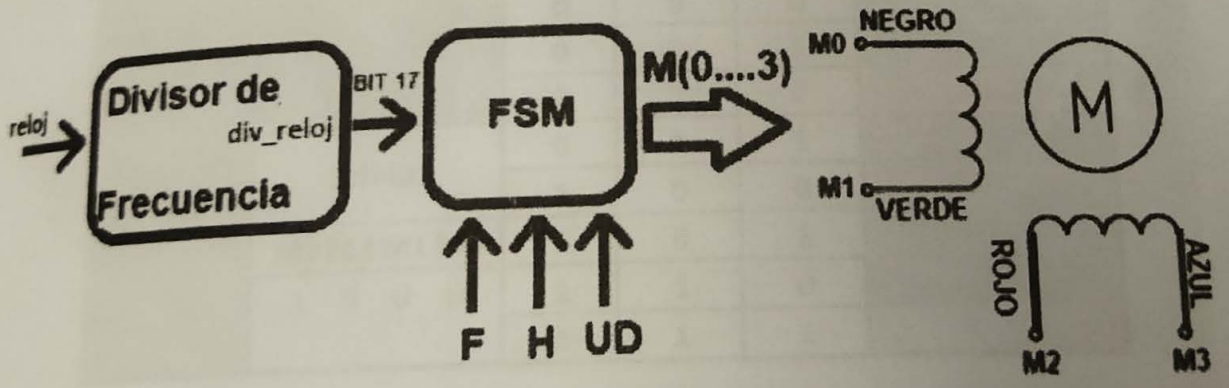

Figura 6.1. Diagrama a bloques del control de motor a pasos

TABLA DE ESTADOS:

La figura 6.2, muestra la tabla de estados, la cual está diseñada con 11 estados que inician en el estado SM0 hasta el estado SM10. Por la cantidad de condiciones de entrada y estados está expresada por colores para cada estado, para una mejor comprensión. En la figura 6.2.A se observa el Estado 0, Estado 1, Estado 2 y Estado 3. En la figura 6.2.B se observa

el Estado 4, Estado 5, Estado 6 y Estado 7. En la figura 6.2.C se observa el Estado 8, Estado 9 y Estado 10.

Estado Presente		Condiciones			
		F	H	UD	Estado Siguente
Estado 0	SM0	0	0	0	SM0
		0	0	1	SM0
		0	1	0	SM0
	Salidas	0	1	1	SM0
		1	0	0	SM0
	M3 M2 M1 M0	1	0	1	SM0
	0 0 0 0	1	1	0	SM0
		1	1	1	SM0
Estado 1	SM1	0	0	0	SM7
		0	0	1	SM3
		0	1	0	SM8
	Salidas	0	1	1	SM2
		1	0	0	SM8
	M3 M2 M1 M0	1	0	1	SM2
	1 0 0 0	1	1	0	SM4
		1	1	1	SM1
Estado 2	SM2	0	0	0	SM7
		0	0	1	SM1
		0	1	0	SM8
	Salidas	0	1	1	SM4
		1	0	0	SM1
	M3 M2 M1 M0	1	0	1	SM8
	1 0 0 0	1	1	0	SM4
		1	1	1	SM9
Estado 3	SM3	0	0	0	SM1
		0	0	1	
		0	1	0	SM8
	Salidas	0	1	1	SM2
		1	0	0	SM2
	M3 M2 M1 M0	1	0	1	SM4
	0 1 0 0	1	1	0	SM4
		1	1	1	SM9

Figura 6.2.A. Estado 0, Estado 1, Estado 2 y Estado 3

Estado Presente	Condiciones			
SM4	F	H	UD	Estado Siguente
	0	0	0	SM7
	0	0	1	SM1
	0	1	0	SM2
	0	1	1	SM6
Salidas	1	0	0	SM3
M3 M2 M1 M0	1	0	1	
0 1 1 0	1	1	0	SM10
	1	1	1	SM9

Estado 4

	F	H	UD	Estado Siguente
	0	0	0	SM5
	0	0	1	SM7
	0	1	0	SM8
	0	1	1	SM2
Salidas	1	0	0	SM6
M3 M2 M1 M0	1	0	1	SM4
0 0 1 0	1	1	0	SM4
	1	1	1	SM9

Estado 6	F	H	UD	Estado Siguente
SM5	0	0	0	SM7
	0	0	1	SM1
	0	1	0	SM4
	0	1	1	SM8
Salidas	1	0	0	
M3 M2 M1 M0	1	0	1	SM7
0 0 1 1	1	1	0	SM4
	1	1	1	SM9

Estado 7	F	H	UD	Estado Siguente
SM7	0	0	0	
	0	0	1	SM1
	0	1	0	SM8
Salidas	0	1	1	SM2
M3 M2 M1 M0	1	0	0	SM6
0 0 0 1	1	0	1	SM8
	1	1	0	SM4
	1	1	1	SM9

Figura 6.2.B. Estado 4, Estado 5, Estado 6 y Estado 7

Estado Presente		Condiciones			
Estado 8	SM8	F	H	UD	Estado Siguente
		0	0	0	SM7
		0	0	1	SM1
		0	1	0	SM6
	Salidas	0	1	1	SM2
		1	0	0	SM7
	M3 M2 M1 M0	1	0	1	SM1
	1 0 0 1	1	1	0	SM9
		1	1	1	SM10

		F	H	UD	Estado Siguente
Estado 9	SM9	0	0	0	SM7
		0	0	1	SM1
		0	1	0	SM8
	Salidas	0	1	1	SM2
		1	0	0	SM8
	M3 M2 M1 M0	1	0	1	SM1
	1 0 1 0	1	1	0	SM4
		1	1	1	SM8

		F	H	UD	Estado Siguente
Estado 10	SM10	0	0	0	SM7
		0	0	1	SM1
		0	1	0	SM8
	Salidas	0	1	1	SM2
		1	0	0	SM8
	M3 M2 M1 M0	1	0	1	SM1
	0 1 0 1	1	1	0	SM8
		1	1	1	SM4

Figura 6.2.C. Estado 8, Estado 9 y Estado 10

Las siguientes figuras muestran el código del control para el motor a pasos, el cual estará contenido en el archivo llamado MotPasos. En la figura 6.3 se observa el código de la entidad y las señales dentro de la arquitectura.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity MotPasos is
   port ( reloj  : in STD_LOGIC;
          UD : in STD_LOGIC;
          rst : in STD_LOGIC;
          FH : in STD_LOGIC_VECTOR(1 downto 0);
          led : out STD_LOGIC_VECTOR(3 downto 0);
          MOT : out STD_LOGIC_VECTOR(3 downto 0) );
end MotPasos;

architecture behavioral of MotPasos is
   signal div : std_logic_vector(17 downto 0);
   signal clks : std_logic;
   type estado is(sm0,sm1, sm2, sm3, sm4, sm5, sm6, sm7, sm8,
                  sm9, sm10);
   signal pres_S, next_s : estado;
   signal motor : std_logic_vector(3 downto 0);
begin
```

Figura 6.3. Código para la entidad y parte de la arquitectura de MotPasos

En la Figura 6.4 se observa el código del bloque Divisor de Frecuencia.

```
process (reloj,rst)
begin
   if rst='0' then
      div <= (others=>'0');
   elsif reloj'event and reloj='1' then
      div <= div + 1;
   end if;
end process;
clks <= div(17);
```

Figura 6.4. Código del bloque Divisor de Frecuencia

En la Figura 6.5 se observa el código de las transiciones de estados.

```
process (clks,rst)
begin
   if rst='0' then
      pres_S <= sm0;
   elsif clks'event and clks='1' then
      pres_S <= next_s;
   end if;
end process;

process (pres_S,UD,rst,FH)
begin
   case(pres_S) is
      when sm0 =>
         next_s <= sm1;                          -- Estado 0
      when sm1 =>                                -- Estado 1
         if FH="00" then ---motor bipolar
            if UD='1' then
               next_s <= sm3;
            else
               next_s <= sm7;
            end if;
      elsif FH="01" then
         if UD='1' then
            next_s <= sm2;
         else
            next_s <= sm8;
         end if;
      elsif FH="10" then
         if UD='1' then
            next_s <= sm2;
         else
            next_s <= sm8;
         end if;
      elsif FH="11" then
         if UD='1' then
            next_s <= sm9;
         else
            next_s <= sm4;
         end if;
      else
         next_s <= sm1;
      end if;
   when sm2 =>                                   -- Estado 2
      if FH="00" then
         if UD='1' then
            next_s <= sm1;
   else
      next s <= sm7;
```

Figura 6.5. Transiciones de estados

```
elsif FH="01" then
   if UD='1' then
      next_s <= sm4;
   else
      next_s <= sm8;
   end if;
elsif FH="10" then
   if UD='1' then
      next_s <= sm3;
   else
      next_s <= sm1;
   end if;
elsif FH="11" then
   if UD='1' then
      next_s <= sm9;
   else
      next_s <= sm4;
   end if;
else
   next_s <= sm2;
end if;
when sm3 =>                                    -- Estado 3
   if FH="00" then
      if UD='1' then
         next_s <= sm5;
      else
         next_s <= sm1;
      end if;
   elsif FH="01" then
      if UD='1' then
         next_s <= sm2;
      else
         next_s <= sm8;
      end if;
   elsif FH="10" then
      if UD='1' then
         next_s <= sm4;
      else
         next_s <= sm2;
      end if;
   elsif FH="11" then
      if UD='1' then
         next_s <= sm9;
      else
         next_s <= sm4;
      end if;
   else
      next_s <= sm3;
end if;
```

Figura 6.5. (continuación) Transiciones de estados

```
        when sm4 =>
          if FH="00" then                    -- Estado 4
            if UD='1' then
              next_s <= sm1;
            else
              next_s <= sm7;
            end if;
          elsif FH="01" then
            if UD='1' then
              next_s <= sm6;
            else
              next_s <= sm2;
            end if;
          elsif FH="10" then
            if UD='1' then
              next_s <= sm5;
            else
              next_s <= sm3;
            end if;
          elsif FH="11" then
            if UD='1' then
              next_s <= sm9;
            else
              next_s <= sm10;
            end if;
        else
          next_s <= sm4;
      end if;
    when sm5 =>
      if FH="00" then                        -- Estado 5
        if UD='1' then
          next_s <= sm7;
        else
          next_s <= sm3;
        end if;
      elsif FH="01" then
        if UD='1' then
          next_s <= sm2;
        else
          next_s <= sm8;
        end if;
      elsif FH="10" then
        if UD='1' then
          next_s <= sm6;
        else
          next_s <= sm4;
        end if;
      elsif FH="11" then
        if UD='1' then
          next_s <= sm9;
        else
          next_s <= sm4;
        end if;
```

Figura 6.5. (continuación) Transiciones de estados

```
      else
         next_s <= sm3;                          -- Estado 6
      end if;
   when sm6 =>
      if FH="00" then
         if UD='1' then
            next_s <= sm1;
         else
            next_s <= sm7;
         end if;
      elsif FH="01" then
         if UD='1' then
            next_s <= sm8;
         else
            next_s <= sm4;
         end if;
      elsif FH="10" then
         if UD='1' then
            next_s <= sm7;
         else
            next_s <= sm5;
         end if;
      elsif FH="11" then
         if UD='1' then
            next_s <= sm9;
         else
            next_s <= sm4;
         end if;
      else
         next_s <= sm7;
      end if;
   when sm7 =>                                   -- Estado 7
      if FH="00" then
         if UD='1' then
            next_s <= sm1;
         else
            next_s <= sm5;
         end if;
      elsif FH="01" then
         if UD='1' then
            next_s <= sm2;
         else
            next_s <= sm8;
         end if;
      elsif FH="10" then
         if UD='1' then
            next_s <= sm8;
         else
            next_s <= sm6;
         end if;
```

Figura 6.5. (continuación) Transiciones de estados

```
        elsif FH="11" then
          if UD='1' then
            next_s <= sm9;
          else
            next_s <= sm4;
          end if;
        else
          next_s <= sm7;
        end if;
      when sm8 =>
        if FH="00" then
          if UD='1' then                    -- Estado 8
            next_s <= sm1;
          else
            next_s <= sm7;
          end if;
        elsif FH="01" then
          if UD='1' then
            next_s <= sm2;
          else
            next_s <= sm6;
          end if;
        elsif FH="10" then
          if UD='1' then
            next_s <= sm1;
          else
            next_s <= sm7;
          end if;
        elsif FH="11" then
          if UD='1' then
            next_s <= sm10;
          else
            next_s <= sm9;
          end if;
        else
          next_s <= sm8;
        end if;
      when sm9=>
        if FH="00" then                     -- Estado 9
          if UD='1' then
            next_s <= sm1;
          else
            next_s <= sm7;
          end if;
        elsif FH="01" then
          if UD='1' then
            next_s <= sm2;
          else
            next_s <= sm8;
          end if;
```

Figura 6.5. (continuación) Transiciones de estados

```
        elsif FH="10" then
          if UD='1' then
            next_s <= sm1;
          else
            next_s <= sm8;
          end if;
        elsif FH="11" then
          if UD='1' then
            next_s <= sm8;
          else
            next_s <= sm4;
          end if;
        else
          next_s <= sm9;
        end if;
      when sm10 =>                                  -- Estado 10
        if FH="00" then
          if UD='1' then
            next_s <= sm1;
          else
            next_s <= sm7;
          end if;
        elsif FH="01" then
          if UD='1' then
            next_s <= sm2;
          else
            next_s <= sm8;
          end if;
        elsif FH="10" then
          if UD='1' then
            next_s <= sm1;
          else
            next_s <= sm8;
          end if;
        elsif FH="11" then
          if UD='1' then
            next_s <= sm4;
          else
            next_s <= sm8;
          end if;
        else
          next_s <= sm10;
        end if;
      when others => next_s <= sm0;
    end case;
end process;
```

Figura 6.5. (continuación) Transiciones de estados

En la Figura 6.6 se observa el código de las salidas de estados al motor.

```
process(pres_S)
begin
  case pres_S is
    when sm0 => motor <= "0000";
    when sm1 => motor <= "1000";
    when sm2 => motor <= "1100";
    when sm3 => motor <= "0100";
    when sm4 => motor <= "0110";
    when sm5 => motor <= "0010";
    when sm6 => motor <= "0011";
    when sm7 => motor <= "0001";
    when sm8 => motor <= "1001";
    when sm9 => motor <= "1010";
    when sm10 => motor <= "0101";
    when others => motor <= "0000";
  end case;
end process;

MOT<=motor;
led<=motor;

end behavioral;
```

Figura 6.6. Salidas de estados

ACTIVIDAD COMPLEMENTARIA:

El alumno deberá realizar las modificaciones pertinentes para poder girar el motor las vueltas necesarias que representen los dígitos de su número de cuenta, se deben combinar los giros horario, anti horario y detenido.