UNIVERSIDAD DE GUADALAJARA

CUCEI

CARRERA: **INGENERIA EN COMPUTACION**

SIMULACION 9: **PROGRAMACIÓN DE SISTEMAS DIGITALES EN VHDL**

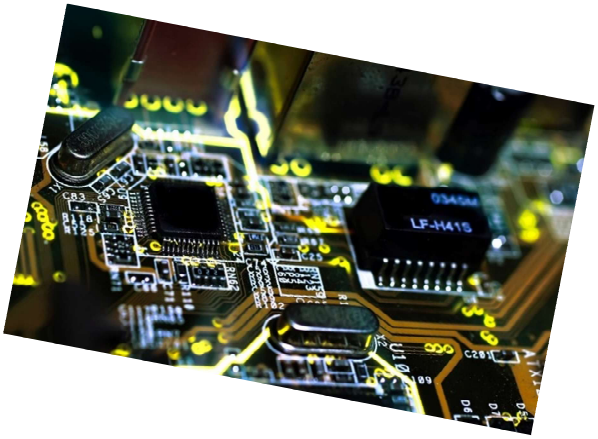
ALUMNO: **EFRAIN ROBLES PULIDO**

CODIGO: **221350095**

NOMBRE DE LA MATERIA: **ARQUITECTURA DE COMPUTADORAS**

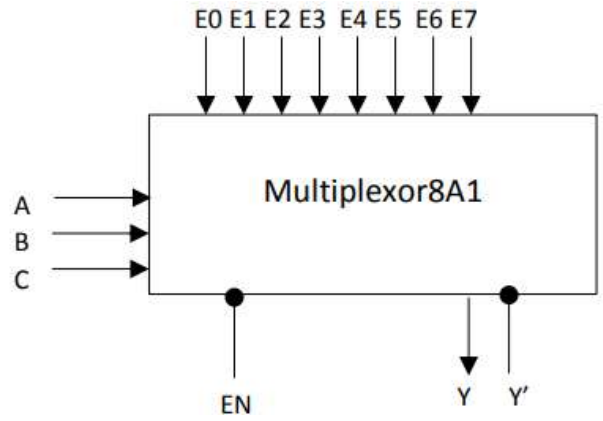
SECCIÓN: **D10** *CALENDARIO:* **2021B**

NOMBRE DE LA PROFESORA: **THELMA ISABEL MORALES RAMIREZ**



Simulación 9_Codigo VHDL

1) Realice un programa de un multiplexor de 1 bit de salida y 8 entradas, como el que se muestra en la figura. Desarrolle su código de programación con base a la tabla de verdad.



EN	C	B	A	Y	Y'
1	X	X	X	0	1
0	0	0	0	E0	E0'
0	0	0	1	E1	E1'
0	0	1	0	E2	E2'
0	0	1	1	E3	E3'
0	1	0	0	E4	E4'
0	1	0	1	E5	E5'
0	1	1	0	E6	E6'
0	1	1	1	E7	E7'

- **Ynot = ~Y**

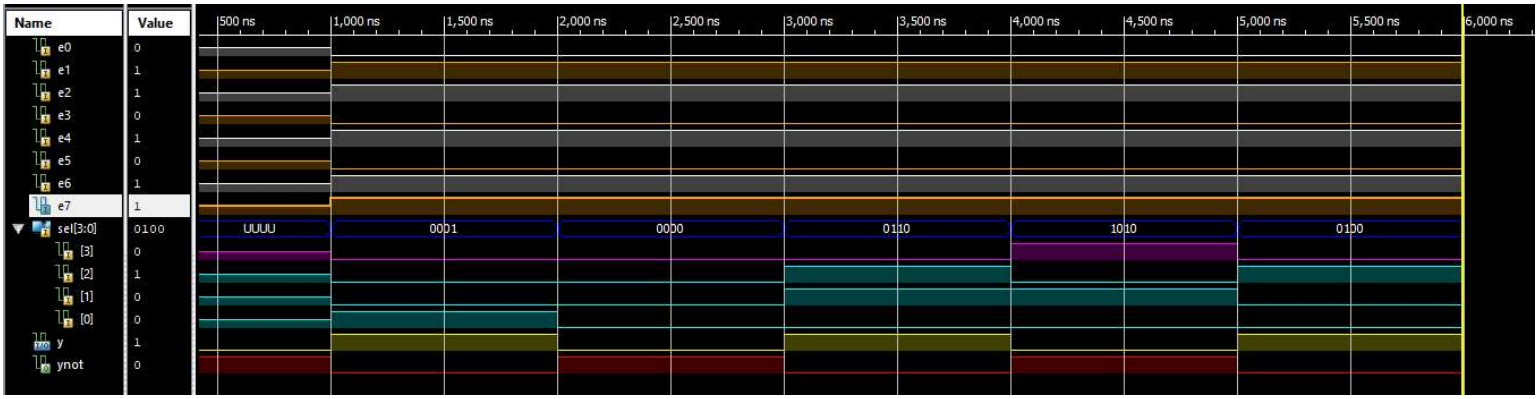
Codigo VHDL:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Multiplexor is
   Port ( E0 : in  STD_LOGIC;--Entradas
          E1 : in  STD_LOGIC;
          E2 : in  STD_LOGIC;
          E3 : in  STD_LOGIC;
          E4 : in  STD_LOGIC;
          E5 : in  STD_LOGIC;
          E6 : in  STD_LOGIC;
          E7 : in  STD_LOGIC;

        --Selectores (A,B,C, EN)
        sel: in  STD_LOGIC_VECTOR (3 downto 0);

        --Salidas
        Y : inout  STD_LOGIC;
        Ynot : out  STD_LOGIC);
end Multiplexor;
```

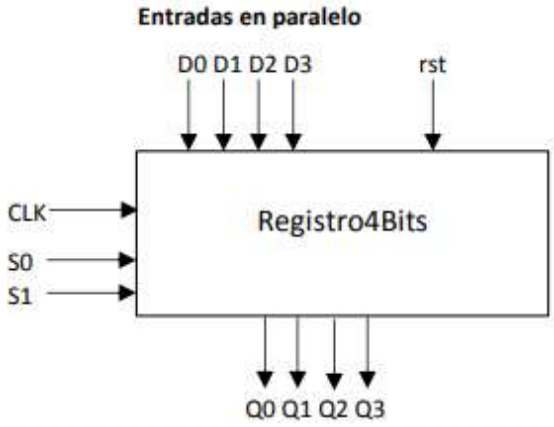
```
architecture Behavioral of Multiplexor is
begin
with sel  select
                        Y <= E0 when "0000",
                             E1 when "0001",
                             E2 when "0010",
                             E3 when "0011",
                             E4 when "0100",
                             E5 when "0101",
                             E6 when "0110",
                             E7 when "0111",
                              '0' when others;
Ynot<= not Y;
end Behavioral;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Multiplexor is
    Port ( E0 : in  STD_LOGIC;--Entradas
           E1 : in  STD_LOGIC;
           E2 : in  STD_LOGIC;
           E3 : in  STD_LOGIC;
           E4 : in  STD_LOGIC;
           E5 : in  STD_LOGIC;
           E6 : in  STD_LOGIC;
           E7 : in  STD_LOGIC;

           --Selectores (A,B,C, EN)
           sel: in  STD_LOGIC_VECTOR (3 downto 0);

           --Salidas
           Y : inout  STD_LOGIC;
           Ynot : out  STD_LOGIC);
end Multiplexor;

architecture Behavioral of Multiplexor is
begin
with sel  select
      Y <= E0 when "0000",
           E1 when "0001",
           E2 when "0010",
           E3 when "0011",
           E4 when "0100",
           E5 when "0101",
           E6 when "0110",
           E7 when "0111",
           '0' when others;

Ynot<= not Y;
end Behavioral;
```

Cronograma:



2) Diseñe un registro de 4 bits como el mostrado en la figura, cuyo funcionamiento se encuentra regulado por las señales de control S0 y S1, tal y como se muestra en la siguiente tabla.



S0	S1	Acción
0	0	Hold (Mantener)
0	1	Desplazamiento izquierda SL
1	0	Load (Carga)
1	1	Desplazamiento derecha SR

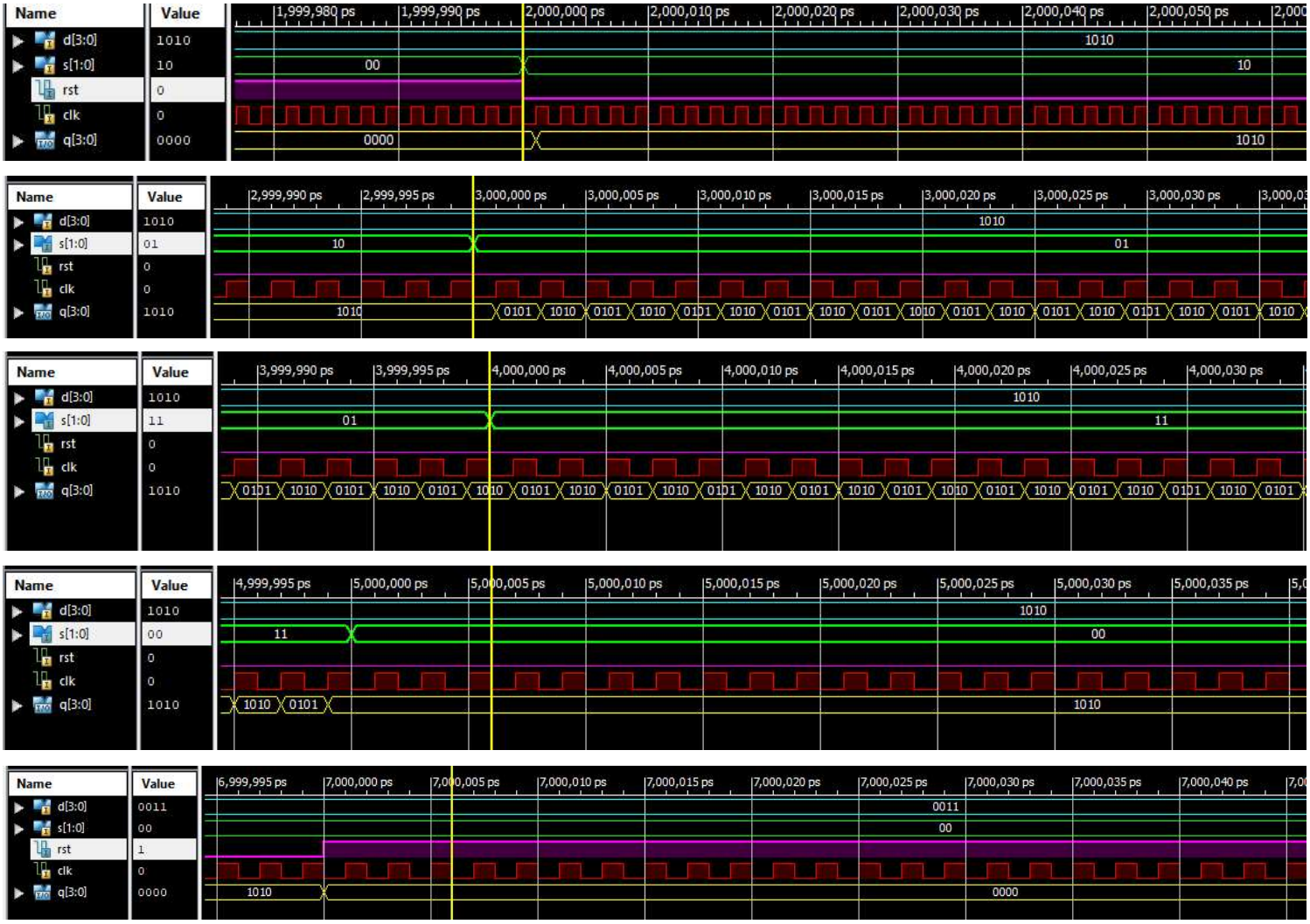
Codigo VHDL:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Registro is
   Port ( D : in  STD_LOGIC_VECTOR (3 downto 0);
          rst : in  STD_LOGIC;
          CLK : in  STD_LOGIC;
          S : in  STD_LOGIC_VECTOR (1 downto 0);
          Q : inout  STD_LOGIC_VECTOR (3 downto 0));
end Registro;
```

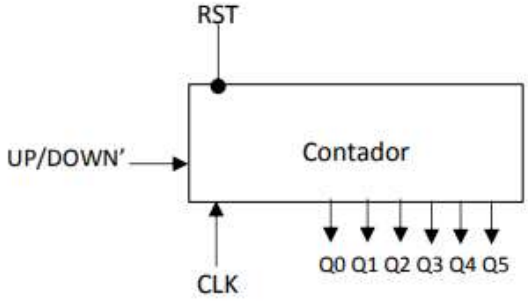
```
architecture Behavioral of Registro is
begin
                process(CLK,D,S,rst)
                begin
                  if (rst='1') then
                        Q<="0000";
                  elsif (CLK' event and CLK='1') then
                        if (S="00") then
                              Q<=Q;
                        elsif (S="01") then
                              Q(3)<=Q(2);
                              Q(2)<=Q(1);
                              Q(1)<=Q(0);
                              Q(0)<=Q(3);
                        elsif (S="10") then
                              Q<=D;
                        elsif (S="11") then
                              Q(3)<=Q(0);
                              Q(2)<=Q(3);
                              Q(1)<=Q(2);
                              Q(0)<=Q(1);
                        end if;
                  end if;
                end process;
end Behavioral;
```

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Registro is
    Port ( D : in  STD_LOGIC_VECTOR (3 downto 0);
           rst : in  STD_LOGIC;
           CLK : in  STD_LOGIC;
           S : in  STD_LOGIC_VECTOR (1 downto 0);
           Q : inout  STD_LOGIC_VECTOR (3 downto 0));
end Registro;

architecture Behavioral of Registro is
begin
   process(CLK,D,S,rst)
   begin
      if (rst='1') then
         Q<="0000";
      elsif (CLK' event and CLK='1') then
         if (S="00") then
            Q<=Q;
         elsif (S="01") then
            Q(3)<=Q(2);
            Q(2)<=Q(1);
            Q(1)<=Q(0);
            Q(0)<=Q(3);
         elsif (S="10") then
            Q<=D;
         elsif (S="11") then
            Q(3)<=Q(0);
            Q(2)<=Q(3);
            Q(1)<=Q(2);
            Q(0)<=Q(1);
         end if;
      end if;
   end process;
end Behavioral;
```

Cronograma:



3) Diseñe y programe un contador de 0 a 59, ascendente (1) y descendente (0). El circuito debe contar con una señal de reset activo en bajo, que coloca las salidas Q en estado bajo.

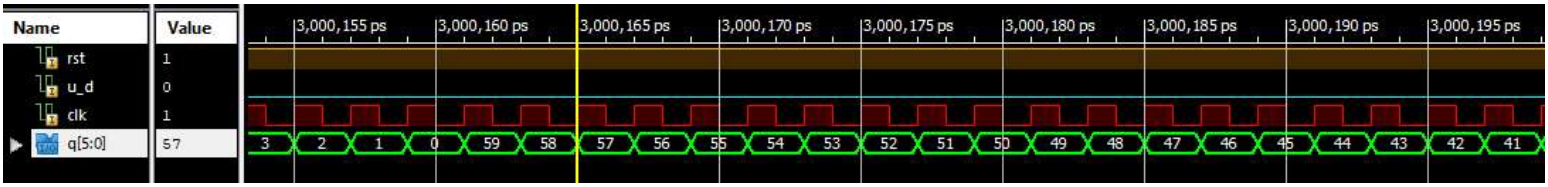


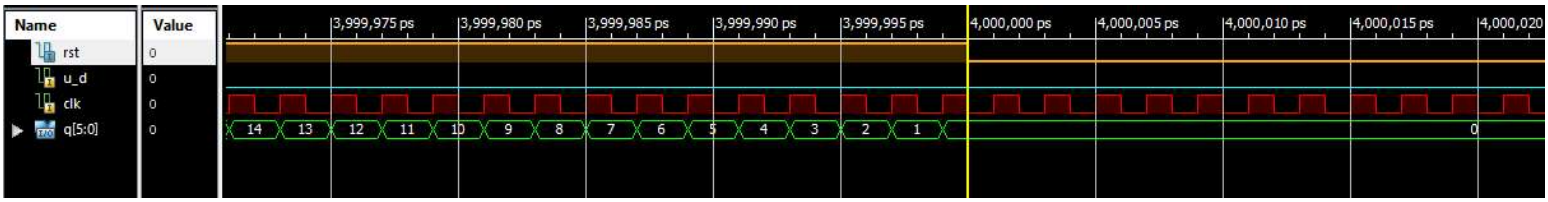
Codigo VHDL:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Contador is
   Port ( RST : in  STD_LOGIC;
          U_D : in  STD_LOGIC;
          CLK : in  STD_LOGIC;
          Q : inout  STD_LOGIC_VECTOR (5 downto 0));
end Contador;

architecture Behavioral of Contador is
begin
                                process (CLK,RST,U_D)
                                begin
                                  if(CLK'event and CLK='1') then
                                                if (RST='0') then
                                                        Q<="000000";
                                                else
                                                        if (U_D='1') then
                                                                Q<=Q+1;
                                                                if (Q="111011") then --59
                                                                        Q<="000000";--0
                                                                end if;
                                                        else
                                                                Q<=Q-1;
                                                                if (Q="000000") then --0
                                                                        Q<="111011";--59
                                                                end if;
                                                        end if;
                                                end if;
                                  end if;
                                end process;
end Behavioral;
```

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Contador is
    Port ( RST : in  STD_LOGIC;
           U_D : in  STD_LOGIC;
           CLK : in  STD_LOGIC;
           Q : inout  STD_LOGIC_VECTOR (5 downto 0));
end Contador;

architecture Behavioral of Contador is
begin
   process (CLK,RST,U_D)
   begin
      if(CLK'event and CLK='1') then
            if (RST='0') then
               Q<="000000";
            else
               if (U_D='1') then
                  Q<=Q+1;
                  if (Q="111011") then --59
                     Q<="000000";--0
                  end if;
               else
                  Q<=Q-1;
                  if (Q="000000") then --0
                     Q<="111011";--59
                  end if;
               end if;
            end if;
      end if;
   end process;
end Behavioral;
```

Cronograma:

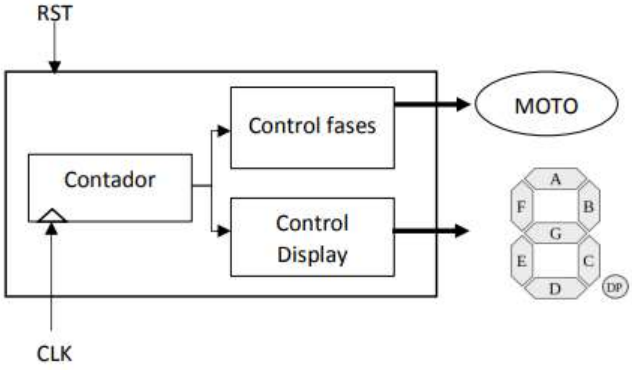




4) El siguiente circuito muestra el control de encendido de un motor a pasos en secuencia de medio tiempo.

- Tiene una señal de reset que permite restablecer la secuencia de conteo inicial.
- Independiente a su control de fases, cuenta con señales de monitoreo que indican en que paso se encuentra el proceso.

Paso	Fase A	Fase B	Fase C	Fase D
0	ON	OFF	OFF	OFF
1	ON	ON	OFF	OFF
2	OFF	ON	OFF	OFF
3	OFF	ON	ON	OFF



Codigo VHDL:

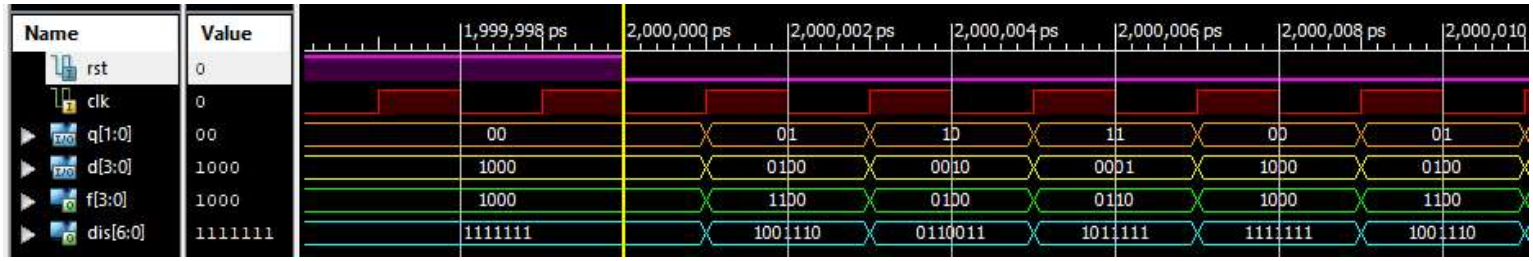
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Motor is
   Port ( RST : in  STD_LOGIC;
          CLK : in  STD_LOGIC;
          D : inout  STD_LOGIC_VECTOR (3 downto 0);
          DIS : out  STD_LOGIC_VECTOR (6 downto 0);
          Q : inout  STD_LOGIC_VECTOR (1 downto 0);
          F : out  STD_LOGIC_VECTOR (3 downto 0));
end Motor;
```

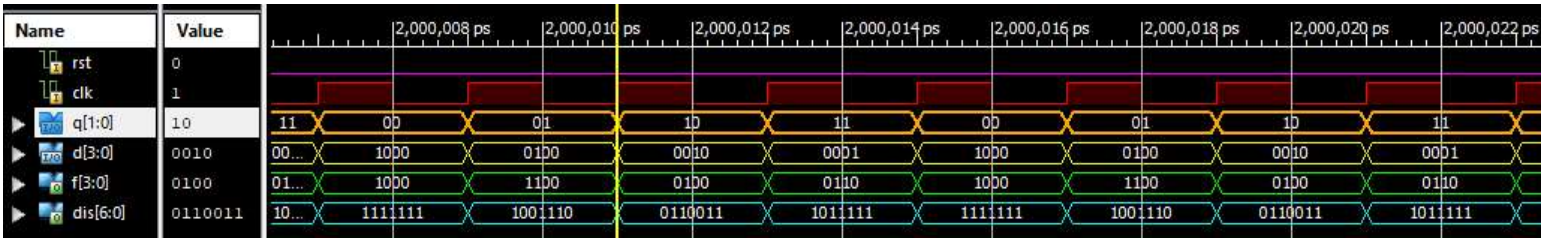
```
architecture Behavioral of Motor is
begin
                cont: process (CLK, RST, Q)--circuito secuencial
                begin
                  if (RST='1') then
                        Q<="00";
                  elsif(CLK'event and CLK='1') then
                        Q<=Q+1;
                  end if;
                end process cont;

                deco: process (Q)--circuito combinacional
                begin
                  case Q is
                        when "00" => D <= "1000";
                        when "01" => D <= "0100";
                        when "10" => D <= "0010";
                        when others => D <= "0001";
                  end case;
                end process deco;

                fase: process (D)--circuito combinacional FASES
                begin
                  case D is
                        when "1000" => F <= "1000";
                        when "0100" => F <= "1100";
                        when "0010" => F <= "0100";
                        when "0001" => F <= "0110";
                        when others => F <= "0000";
                  end case;
                end process fase;

                disp: process (D)--circuito combinacional DISPLAY
                begin
                  case D is----------------------------abcdefg
                        when "1000" => DIS <= "1111111";--8
                        when "0100" => DIS <= "1001110";--12-c
                        when "0010" => DIS <= "0110011";--4
                        when "0001" => DIS <= "1011111";--6
                        when others => DIS <= "0000000";--nada
                  end case;
                end process disp;
end Behavioral;
```

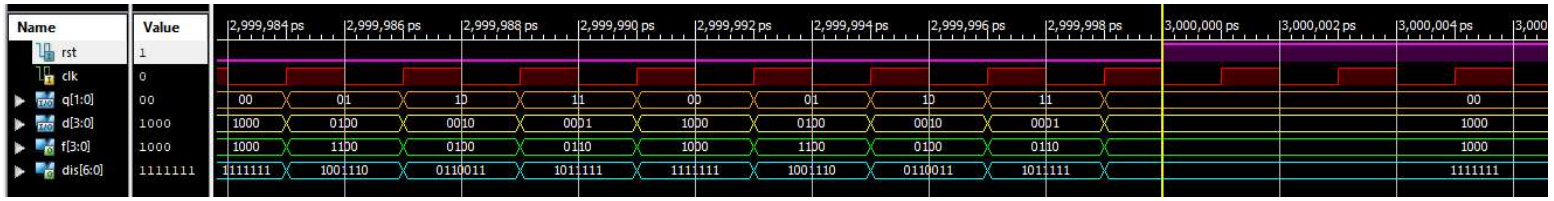
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Motor is
    Port ( RST : in  STD_LOGIC;
           CLK : in  STD_LOGIC;
           D : inout  STD_LOGIC_VECTOR (3 downto 0);
           DIS : out  STD_LOGIC_VECTOR (6 downto 0);
           Q : inout  STD_LOGIC_VECTOR (1 downto 0);
           F : out  STD_LOGIC_VECTOR (3 downto 0));
end Motor;

architecture Behavioral of Motor is
begin
   cont: process (CLK, RST, Q)--circuito secuencial
   begin
      if (RST='1') then
         Q<="00";
      elsif(CLK'event and CLK='1') then
         Q<=Q+1;
      end if;
   end process cont;

   deco: process (Q)--circuito combinacional
   begin
      case Q is
         when "00" => D <= "1000";
         when "01" => D <= "0100";
         when "10" => D <= "0010";
         when others => D <= "0001";
      end case;
   end process deco;
```

```
   fase: process (D)--circuito combinacional FASES
   begin
      case D is
         when "1000" => F <= "1000";
         when "0100" => F <= "1100";
         when "0010" => F <= "0100";
         when "0001" => F <= "0110";
         when others => F <= "0000";
      end case;
   end process fase;

   disp: process (D)--circuito combinacional DISPLAY
   begin
      case D is------------------abcdefg
         when "1000" => DIS <= "1111111";--8
         when "0100" => DIS <= "1001110";--12-c
         when "0010" => DIS <= "0110011";--4
         when "0001" => DIS <= "1011111";--6
         when others => DIS <= "0000000";--nada
      end case;
   end process disp;
end Behavioral;
```

Cronograma:


Name
Value
rst
0
clk
0
q[1:0]
00
d[3:0]
1000
f[3:0]
1000
dis[6:0]
1111111
1,999,998 ps
2,000,000 ps
2,000,002 ps
2,000,004 ps
2,000,006 ps
2,000,008 ps
2,000,010
00
01
10
11
00
01
1000
0100
0010
0001
1000
0100
1000
1100
0100
0110
1000
1100
1111111
1001110
0110011
1011111
1111111
1001110


Name
Value
rst
0
clk
1
q[1:0]
10
d[3:0]
0010
f[3:0]
0100
dis[6:0]
0110011
2,000,008 ps
2,000,010 ps
2,000,012 ps
2,000,014 ps
2,000,016 ps
2,000,018 ps
2,000,020 ps
2,000,022 ps
11
00
01
10
11
00
01
10
11
00...
1000
0100
0010
0001
1000
0100
0010
0001
01...
1000
1100
0100
0110
1000
1100
0100
0110
10...
1111111
1001110
0110011
1011111
1111111
1001110
0110011
1011111


Name
Value
rst
1
clk
0
q[1:0]
00
d[3:0]
1000
f[3:0]
1000
dis[6:0]
1111111
2,999,984 ps
2,999,986 ps
2,999,988 ps
2,999,990 ps
2,999,992 ps
2,999,994 ps
2,999,996 ps
2,999,998 ps
3,000,000 ps
3,000,002 ps
3,000,004 ps
3,000
00
01
10
11
00
01
10
11
00
1000
0100
0010
0001
1000
0100
0010
0001
1000
1000
1100
0100
0110
1000
1100
0100
0110
1000
1111111
1001110
0110011
1011111
1111111
1001110
0110011
1011111
1111111