Práctica 2.

DISEÑO DE REGISTROS DE CORRIMIENTO EN CASCADA

OBJETIVO:

Demostrar a los estudiantes mediante el diseño de registros de corrimiento en cascada, que las declaraciones secuenciales requieren de un orden para ser ejecutadas, utilizando las estructuras de control *if-then-else* o *case* dentro de un proceso.

ESPECIFICACIONES:

Utilizando un FPGA y 8 displays de 7 segmentos, diseñar un sistema digital que despliegue un mensaje que se vea recorrer en los displays.

La figura 2.1 muestra el diagrama de bloques del sistema Registros de Corrimiento en Cascada.

DIAGRAMA DE BLOQUES:

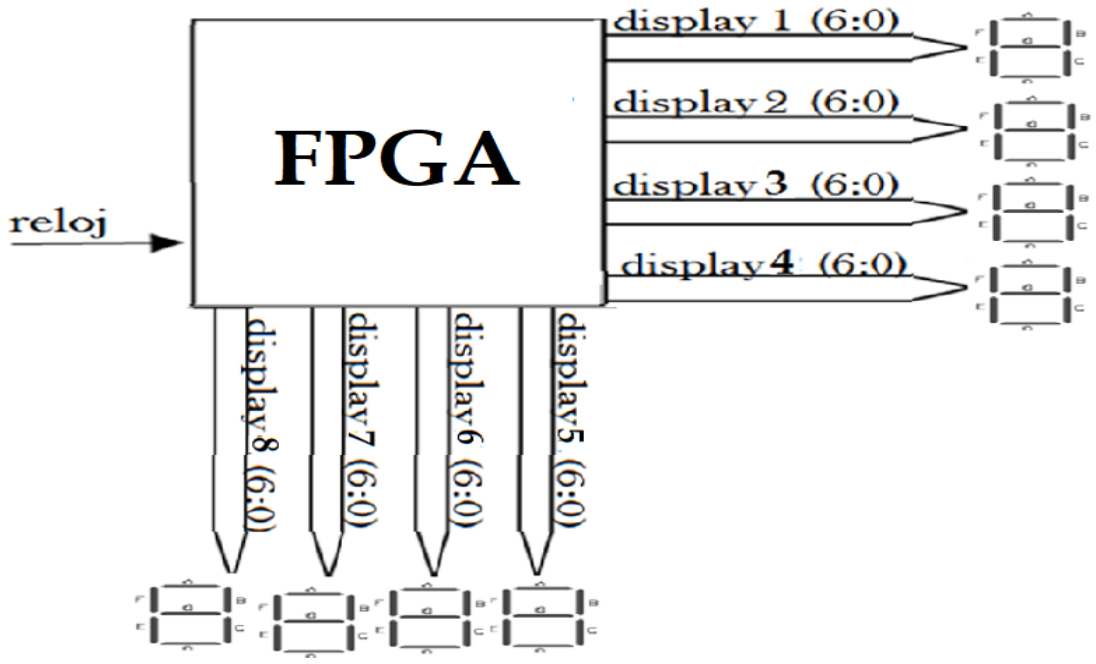

Figura 2.1. Diagrama de bloques del sistema Registros de Corrimiento en Cascada.

Dentro del sistema digital Registros de Corrimiento en Cascada, se tienen varios bloques funcionales, los cuales internamente ejecutan instrucciones en forma secuencial. La figura 2.2 muestra los bloques funcionales del sistema.

BLOQUES FUNCIONALES:

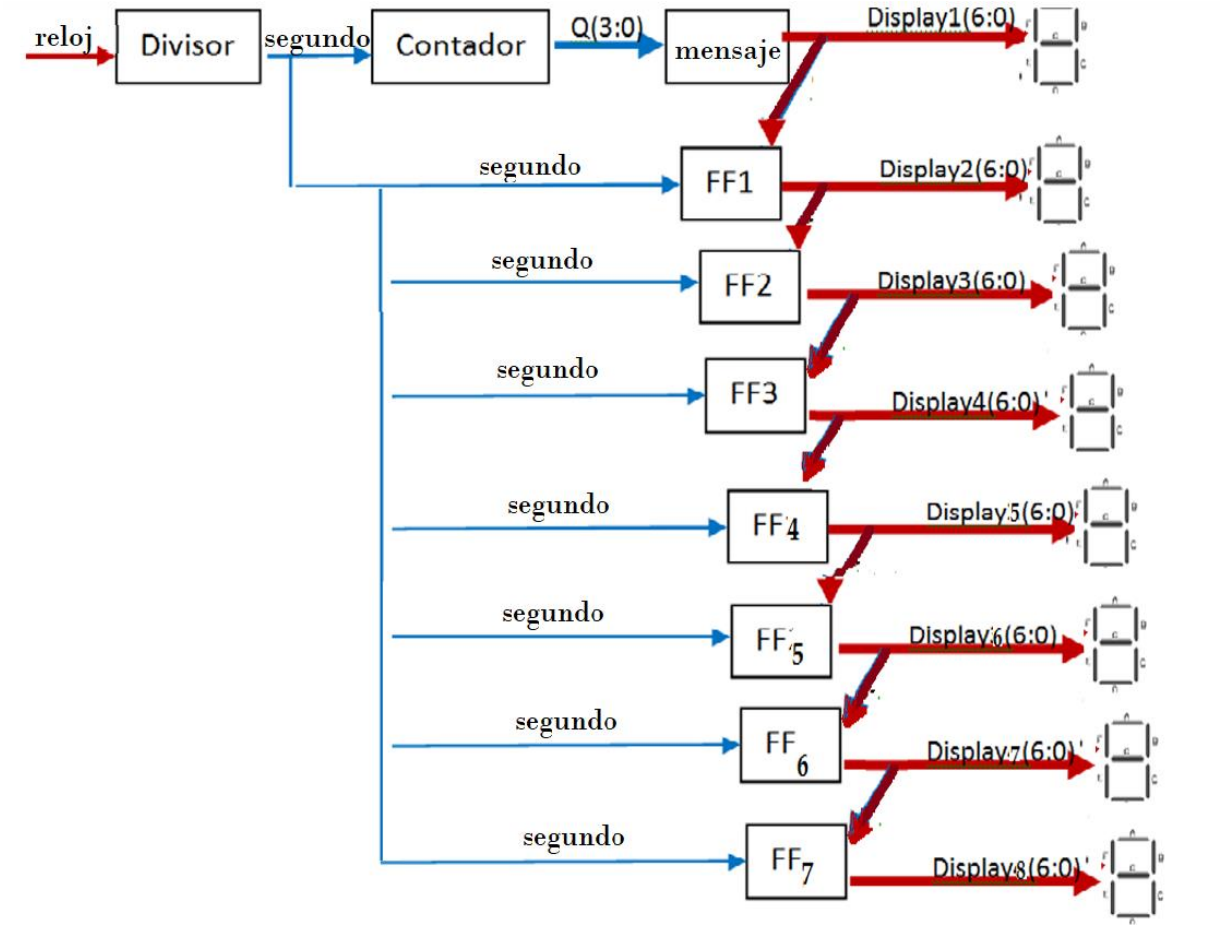

Figura 2.2. Diagrama de bloques funcionales del sistema Registros de Corrimiento en Cascada

La figura 2.3 muestra la entidad del sistema Registros de Corrimiento en Cascada.

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

entity corri is
   Port ( reloj : in  std_logic;
          display1, display2, display3, display4, display5, display6,
          display7, display8 : buffer std_logic_vector (6 downto 0));
end corri;
```

Figura 2.3. Entidad del sistema Registros de Corrimiento en Cascada

La figura 2.4 muestra la parte declaratoria de la arquitectura del sistema Registros de Corrimiento en Cascada.

```
architecture Behavioral of corri is
   signal segundo : std_logic;
   signal Q : std_logic_vector(3 downto 0):="0000";
```

Figura 2.4. Parte declaratoria de la arquitectura del sistema Registros de Corrimiento en Cascada

La figura 2.5 muestra la parte operatoria de la arquitectura del sistema Registros de Corrimiento en Cascada.

```
begin
   divisor : process (reloj)
      variable CUENTA: std_logic_vector(27 downto 0) :=  X"0000000";
   begin
      if rising_edge (reloj) then
         if CUENTA =X"48009E0" then
            cuenta :=  X"0000000";
         else
            cuenta  := cuenta+1;
         end if;
      end if;
      segundo  <=CUENTA(22);
   end process;

   contador : process (segundo)
   begin
      if rising_edge (segundo) then
         Q  <= Q +1;
      end if;
   end process;
```

Figura 2.5. Parte operatoria de la arquitectura del sistema Registros de Corrimiento en Cascada

```
   with Q select
      display1 <= "0000110" when "0000",  -- E
                  "0101011" when "0001",  -- n
                  "1111111" when "0010",  -- espacio
                  "1000111" when "0011",  -- L
                  "0001000" when "0100",  -- A
                  "1111111" when "0101",  -- espacio
                  "1000000" when "0110",  -- O
                  "1000111" when "0111",  -- L
                  "0001000" when "1000",  -- A
                  "1111111" when others;  -- espacios

   FF1 : process (segundo)
   begin
      if rising_edge (segundo) then
         display2 <= display1;
      end if;
   end process;

   FF2 : process (segundo)
   begin
      if rising_edge (segundo) then
         display3 <= display2;
      end if;
   end process;

   FF3 : process (segundo)
   begin
      if rising_edge (segundo) then
         display4 <= display3;
      end if;
   end process;

   FF4 : process (segundo)
   begin
      if rising_edge (segundo) then
         display5 <= display4;
      end if;
   end process;

   FF5 : process (segundo)
   begin
      if rising_edge (segundo) then
         display6 <= display5;
      end if;
   end process;

   FF6 : process (segundo)
   begin
      if rising_edge (segundo) then
         display7 <= display6;
      end if;
   end process;
```

Figura 2.5. (continuación) Parte operatoria de la arquitectura del sistema Registros de Corrimiento en Cascada

```
   FF7 : process (segundo)
   begin
      if rising_edge (segundo) then
         display8 <= display7;
      end if;
   end process;
end Behavioral;
```

Figura 2.5. (continuación) Parte operatoria de la arquitectura del sistema Registros de Corrimiento en Cascada

NOTA IMPORTANTE: El código mostrado fue realizado considerando que se tiene una tarjeta de desarrollo con cada display de 7 segmentos conectado directamente al FPGA. En caso de que en su tarjeta los displays de 7 segmentos estén multiplexados, hay que realizar los ajustes necesarios, partiendo del bloque multiplexor creado en la práctica 1, para crear el bloque Mux4disp.

ACTIVIDADES COMPLEMENTARIAS:

1.- A partir del bloque multiplexor creado en la práctica 1, el alumno diseñará un bloque funcional que contenga los elementos para multiplexar las señales para 4 displays de 7 segmentos, de tal manera que este bloque pueda ser utilizado más adelante para otras aplicaciones. Este bloque tendrá la estructura mostrada en la figura 2.6, estará contenida en un archivo denominado Mux4disp.vhd, y pasará a ser parte de la biblioteca de módulos funcionales del alumno.

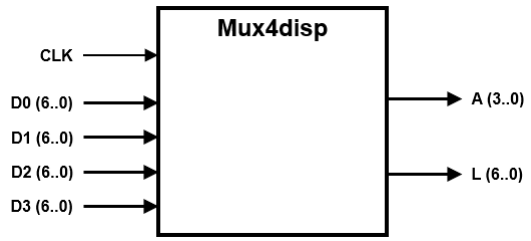

Figura 2.6. Diagrama de bloque del módulo Mux4disp

2.- Las demás actividades complementarias serán presentadas el día de la práctica.