



# Universidad Nacional Autónoma de México

FACULTAD DE INGENIERÍA

Diseño Digital VLSI

Práctica 2. Diseño de registros de corrimientos en cascada

*Semestre: 2025-1 Brigada 03*

**Integrantes:**

**Martínez Jiménez Israel**

**Pérez González Sharon Leslie**

**Gómez Urbano Mariana**

**Hernández Ramírez Miguel Ángel**



# Índice

<b>1. Ejercicio 2.3</b>	<b>3</b>
1.1. Módulo conta . . . . .	3
1.2. Módulo display . . . . .	6
1.3. Módulo divfreq . . . . .	7
1.4. Módulo ss7 . . . . .	9
1.5. Ejercicio 2.3 . . . . .	11



## 1. Ejercicio 2.3

### 1.1. Módulo conta

Este programa implementa un contador secuencial utilizando una máquina de estados finitos (FSM). El contador avanza a través de una serie de estados predefinidos en cada ciclo de reloj, reiniciándose al llegar al último estado o cuando se activa la señal de reinicio. La salida del contador refleja el estado actual, que es utilizado por otros módulos para coordinar las operaciones.

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3
4 -- Entidad 'conta' define un contador secuencial que transita entre
   varios estados
5 -- de 4 bits en funcion de un reloj y una señal de reinicio.
6 entity conta is
7     port (
8         clk          : in std_logic;  -- Señal de reloj de entrada
9         reset        : in std_logic;  -- Señal de reinicio de entrada
10        SalMoore      : out std_logic_vector(3 downto 0)  -- Salida del estado
           actual del contador
11    );
12 end entity conta;
13
14 architecture arqconta of conta is
15     subtype state is std_logic_vector(3 downto 0);  -- Definición del tipo
           de dato 'state' como un vector lógico de 4 bits
16     signal present_state, next_state : std_logic_vector(3 downto 0);  --
           Señales para almacenar el estado presente y el siguiente estado
17
```



```
18  -- Definición de los posibles estados del contador
19  constant e0 : state := "0000";
20  constant e1 : state := "0001";
21  constant e2 : state := "0010";
22  constant e3 : state := "0011";
23  constant e4 : state := "0100";
24  constant e5 : state := "0101";
25  constant e6 : state := "0110";
26  constant e7 : state := "0111";
27  constant e8 : state := "1000";
28  constant e9 : state := "1001";
29  constant ea : state := "1010";
30
31  begin
32  -- Proceso que describe la lógica secuencial para la transición de
33    estados en función del reloj y la señal de reinicio.
34  process (clk)
35  begin
36    if rising_edge(clk) then
37      if reset = '0' then
38        present_state <= e0;  -- Reinicia al estado inicial si reset está
39                               activo
40      else
41        present_state <= next_state;  -- Avanza al siguiente estado
42      end if;
43    end if;
44  end process;
```



```
44  -- Proceso que define la lógica combinacional para determinar el
    siguiente estado basado en el estado presente.
45  process (present_state)
46  begin
47      case present_state is
48          when e0      => next_state <= e1;
49          when e1      => next_state <= e2;
50          when e2      => next_state <= e3;
51          when e3      => next_state <= e4;
52          when e4      => next_state <= e5;
53          when e5      => next_state <= e6;
54          when e6      => next_state <= e7;
55          when e7      => next_state <= e8;
56          when e8      => next_state <= e9;
57          when e9      => next_state <= ea;
58          when ea      => next_state <= e0;
59          when others => next_state <= e0;  -- Estado por defecto
60      end case;
61      SalMoore <= present_state;  -- Actualiza la salida con el estado
    presente
62  end process;
63 end architecture;
```



## 1.2. Módulo display

El módulo display toma una señal de entrada y la transfiere a la salida en cada flanco de subida del reloj. Este módulo es esencial para la visualización de datos en un display de 7 segmentos, permitiendo mostrar el valor que se le asigna desde otros módulos.

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3
4 -- Entidad 'display' que transmite el valor de entrada a la salida en el
   flanco de subida del reloj.
5 entity display is
6     port (
7         clk : in std_logic;    -- Señal de reloj de entrada
8         mi  : in std_logic_vector(6 downto 0); -- Entrada de datos de 7 bits
9         mo  : out std_logic_vector(6 downto 0) -- Salida de datos de 7 bits
10    );
11 end entity;
12
13 architecture arqdisp of display is
14 begin
15     -- Proceso secuencial que actualiza la salida 'mo' con la entrada 'mi'
       en cada flanco de subida del reloj.
16     process (clk)
17     begin
18         if rising_edge(clk) then
19             mo <= mi;
20         end if;
21     end process;
22 end architecture;
```



### 1.3. Módulo divfreq

Este módulo divide la frecuencia de la señal de reloj de entrada para generar una señal de reloj más lenta. Esto es útil para controlar la velocidad a la que se ejecutan ciertos procesos en el sistema, como el desplazamiento de mensajes en un display.

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3
4 -- Entidad 'divfreq' que implementa un divisor de frecuencia.
5 entity divfreq is
6     port (
7         clk    : in std_logic;    -- Señal de reloj de entrada
8         clk1    : buffer std_logic -- Señal de reloj de salida con frecuencia
9                                     dividida
10    );
11 end entity;
12
13 architecture rtl of divfreq is
14     signal contador : integer range 0 to 25000000; -- Contador para
15                                                     dividir la frecuencia
16
17 begin
18     -- Proceso secuencial que divide la frecuencia del reloj de entrada.
19     process (clk)
20     begin
21         if rising_edge(clk) then
22             if (contador = 25000000) then
23                 contador <= 0;
24             end if;
25         end if;
26     end process;
27 end architecture;
```



```
22         clk1      <= not clk1;  -- Invierte la señal de salida cuando el
                                   contador alcanza su valor máximo
23     else
24         contador <= contador + 1;  -- Incrementa el contador
25     end if;
26 end if;
27 end process;
28 end architecture;
```





## 1.4. Módulo ss7

El módulo ss7 decodifica un código BCD (Binary-Coded Decimal) de 4 bits en un patrón correspondiente a un display de 7 segmentos. Este decodificador permite representar caracteres específicos (como letras y espacios) en un display, basado en la entrada BCD.

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  -- Entidad 'ss7' que convierte un código BCD de 4 bits en un valor de 7
   segmentos para un display.
5  entity ss7 is
6      port (
7          bcd : in std_logic_vector(3 downto 0);  -- Entrada BCD de 4 bits
8          hex : out std_logic_vector(6 downto 0)  -- Salida para display de 7
               segmentos
9      );
10 end entity;
11
12 architecture arqss7 of ss7 is
13 begin
14     -- Decodificador para display de 7 segmentos basado en la entrada BCD
15     with bcd select
16         hex <=
17             "0001001" when "0000", -- H
18             "1000000" when "0001", -- O
19             "1000111" when "0010", -- L
20             "1001111" when "0011", -- I
21             "1111111" when "0100", -- Espacio
22             "0001100" when "0101", -- P

```



```
23      "0101111" when "0110", -- r
24      "1000000" when "0111", -- O
25      "0001110" when "1000", -- F
26      "0000110" when "1009", -- E
27      "1111111" when "1010", -- Espacio
28      "1111111" when others; -- Espacio por defecto
29 end architecture;
```



## 1.5. Ejercicio 2.3

Este programa integra todos los módulos anteriores para mostrar el mensaje "HOLI PrOFE".<sup>en</sup> un conjunto de displays de 7 segmentos. El mensaje se desplaza de derecha a izquierda, controlado por el reloj dividido. El contador selecciona qué carácter mostrar en cada momento, y el decodificador ss7 traduce los valores BCD en patrones de 7 segmentos. Los módulos de visualización desplazan el mensaje a lo largo de los displays, creando un efecto de corrimiento.

```

1
2 library ieee;
3 use ieee.std_logic_1164.all;
4
5 -- La entidad 'Ejercicio03' combina varios modulos para desplegar el
   mensaje "HOLI PrOFE"
6 -- en un conjunto de displays de 7 segmentos, donde los caracteres se
   desplazan hacia la izquierda
7 -- en cada ciclo del reloj dividido.
8 entity Ejercicio03 is
9     port (
10         clk, reset                : in std_logic;  -- Señales de reloj y
               reinicio de entrada
11         hex0, hex1, hex2, hex3, hex4 : buffer std_logic_vector(6 downto 0)
               -- Salidas para los displays de 7 segmentos
12     );
13 end entity;
14
15 architecture arqmain of Ejercicio03 is
16     signal clk1 : std_logic;  -- Señal de reloj dividida que controla la
               velocidad de desplazamiento del mensaje

```



```
17  signal bcd    : std_logic_vector(3 downto 0) := "0000";  -- Señal BCD que
    representa el carácter actual a mostrar
18  begin
19  -- Proceso 1: División de frecuencia
20  -- El módulo 'divfreq' divide la frecuencia del reloj de entrada 'clk'
    para obtener un reloj más lento 'clk1',
21  -- que controla la velocidad de desplazamiento del mensaje en los
    displays.
22  u1 : entity work.divfreq(rtl) port map(clk, clk1);
23
24  -- Proceso 2: Contador de estados
25  -- El módulo 'conta' actúa como un contador secuencial que cambia entre
    los estados definidos,
26  -- representando las posiciones de cada carácter del mensaje "HOLI
    PrOFE".
27  -- El contador avanza en cada flanco de subida del reloj 'clk1'.
28  -- La salida del contador (bcd) es usada para seleccionar qué carácter
    se mostrará en el display correspondiente.
29  u2 : entity work.conta(arqconta) port map(clk1, reset, bcd);
30
31  -- Proceso 3: Decodificación de caracteres
32  -- El módulo 'ss7' decodifica la señal BCD a un patrón de 7 segmentos,
33  -- que representa un carácter específico (H, O, L, I, espacio, P, r, O,
    F, E) en el display.
34  -- Cada carácter del mensaje es decodificado y enviado a uno de los
    displays.
35  u3 : entity work.ss7(arqss7) port map(bcd, hex4);
36
37  -- Procesos 4-7: Despliegue en displays con desplazamiento
```



```
38  -- Los módulos 'display' se utilizan para crear un efecto de
    desplazamiento.
39  -- Cada vez que el reloj 'clk1' tiene un flanco de subida, el contenido
    de un display se mueve al siguiente display.
40  -- Esto genera el efecto de corrimiento, mostrando el mensaje "HOLI
    Pr0FE" en secuencia.
41  u4 : entity work.display(arqdisp) port map(clk1, hex4, hex3);
42  u5 : entity work.display(arqdisp) port map(clk1, hex3, hex2);
43  u6 : entity work.display(arqdisp) port map(clk1, hex2, hex1);
44  u7 : entity work.display(arqdisp) port map(clk1, hex1, hex0);
45
46  end architecture;
```

## Referencias

- [1] Navabi, Z. (2007). *VHDL: Modular Design and Synthesis of Cores and Systems*. McGraw-Hill Education.
- [2] Pedroni, V. A. (2004). *Digital Design and Modeling with VHDL and Synthesis: An Integrated Approach*. MIT Press.