DISEÑO DE REGISTROS DE CORRIMIENTOS EN CASCADA

EJERCICIO 2.3

INTEGRANTES:
HERNANDEZ RAMIREZ MIGUEL ANGEL
GÓMEZ URBANO MARIANA
MARTINEZ JIMENEZ ISRAEL
PÉREZ GONZÁLEZ SHARON LESLIE

EQUIPO: 3

PRACTICA 2

D.D.VLSI GRUPO:2

EJERCICIO

EJERCICIO 2.3 CAMBIAR EL MENSAJE PARA LOS VISUALIZADORES CON CORRIMIENTO.

MÒDULO CONTA

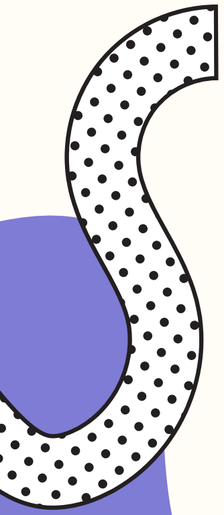
```vhdl
library ieee;
use ieee.std_logic_1164.all;

-- Entidad 'conta' define un contador secuencial que transita entre
   varios estados
-- de 4 bits en funcion de un reloj y una señal de reinicio.
entity conta is
  port (
    clk      : in std_logic;  -- Señal de reloj de entrada
    reset    : in std_logic;  -- Señal de reinicio de entrada
    SalMoore : out std_logic_vector(3 downto 0)  -- Salida del estado
       actual del contador
  );
end entity conta;

architecture arqconta of conta is
  subtype state is std_logic_vector(3 downto 0);  -- Definición del tipo
     de dato 'state' como un vector lógico de 4 bits
  signal present_state, next_state : std_logic_vector(3 downto 0);  --
     Señales para almacenar el estado presente y el siguiente estado
  -- Definición de los posibles estados del contador
  constant e0 : state := "0000";
  constant e1 : state := "0001";
  constant e2 : state := "0010";
  constant e3 : state := "0011";
  constant e4 : state := "0100";
  constant e5 : state := "0101";
  constant e6 : state := "0110";
  constant e7 : state := "0111";
  constant e8 : state := "1000";
  constant e9 : state := "1001";
  constant ea : state := "1010";

begin
  -- Proceso que describe la lógica secuencial para la transición de
     estados en función del reloj y la señal de reinicio.
  process (clk)
  begin
    if rising_edge(clk) then
      if reset = '0' then
        present_state <= e0;  -- Reinicia al estado inicial si reset está
           activo
      else
        present_state <= next_state;  -- Avanza al siguiente estado
      end if;
    end if;
  end process;

```

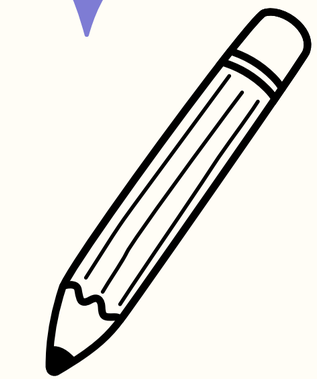
MÒDULO CONTA

```
    -- Proceso que define la lógica combinacional para determinar el
       siguiente estado basado en el estado presente.
    process (present_state)
    begin
      case present_state is
        when e0     => next_state <= e1;
        when e1     => next_state <= e2;
        when e2     => next_state <= e3;
        when e3     => next_state <= e4;
        when e4     => next_state <= e5;
        when e5     => next_state <= e6;
        when e6     => next_state <= e7;
        when e7     => next_state <= e8;
        when e8     => next_state <= e9;
        when e9     => next_state <= ea;
        when ea     => next_state <= e0;
        when others => next_state <= e0;   -- Estado por defecto
      end case;
      SalMoore <= present_state;  -- Actualiza la salida con el estado
         presente
    end process;
end architecture;
```

MÒDULO DISPLAY

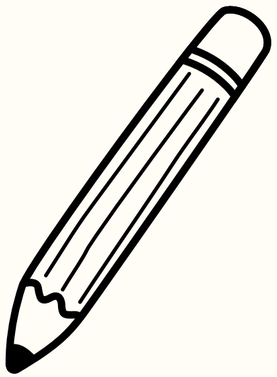
```vhdl
library ieee;
use ieee.std_logic_1164.all;

-- Entidad 'display' que transmite el valor de entrada a la salida en el
   flanco de subida del reloj.
entity display is
  port (
    clk : in std_logic;  -- Señal de reloj de entrada
    mi  : in std_logic_vector(6 downto 0);  -- Entrada de datos de 7 bits
    mo  : out std_logic_vector(6 downto 0)  -- Salida de datos de 7 bits
  );
end entity;

architecture arqdisp of display is
begin
  -- Proceso secuencial que actualiza la salida 'mo' con la entrada 'mi'
     en cada flanco de subida del reloj.
  process (clk)
  begin
    if rising_edge(clk) then
      mo <= mi;
    end if;
  end process;
end architecture;
```





MÓDULO DIV.FREC

```vhdl
library ieee;
use ieee.std_logic_1164.all;

-- Entidad 'divfreq' que implementa un divisor de frecuencia.
entity divfreq is
  port (
    clk  : in std_logic;  -- Señal de reloj de entrada
    clkl : buffer std_logic  -- Señal de reloj de salida con frecuencia
       dividida
  );
end entity;

architecture rtl of divfreq is
  signal contador : integer range 0 to 25000000;  -- Contador para
     dividir la frecuencia

begin
  -- Proceso secuencial que divide la frecuencia del reloj de entrada.
  process (clk)
  begin
    if rising_edge(clk) then
      if (contador = 25000000) then
        contador <= 0;
        clkl     <= not clkl;  -- Invierte la señal de salida cuando el
           contador alcanza su valor máximo
      else
        contador <= contador + 1;  -- Incrementa el contador
      end if;
    end if;
  end process;
end architecture;
```



MÒDULO SS7

```vhdl
library ieee;
use ieee.std_logic_1164.all;

-- Entidad 'ss7' que convierte un código BCD de 4 bits en un valor de 7 segmentos para un display.
entity ss7 is
  port (
    bcd : in std_logic_vector(3 downto 0);   -- Entrada BCD de 4 bits
    hex : out std_logic_vector(6 downto 0)   -- Salida para display de 7 segmentos
  );
end entity;

architecture arqss7 of ss7 is
begin
  -- Decodificador para display de 7 segmentos basado en la entrada BCD
  with bcd select
    hex <=
    "0001001" when "0000", -- H
    "1000000" when "0001", -- O
    "1000111" when "0010", -- L
    "1001111" when "0011", -- I
    "1111111" when "0100", -- Espacio
    "0001100" when "0101", -- P
    "0101111" when "0110", -- r
    "1000000" when "0111", -- O
    "0001110" when "1000", -- F
    "0000110" when "1009", -- E
    "1111111" when "1010", -- Espacio
    "1111111" when others; -- Espacio por defecto
end architecture;
```

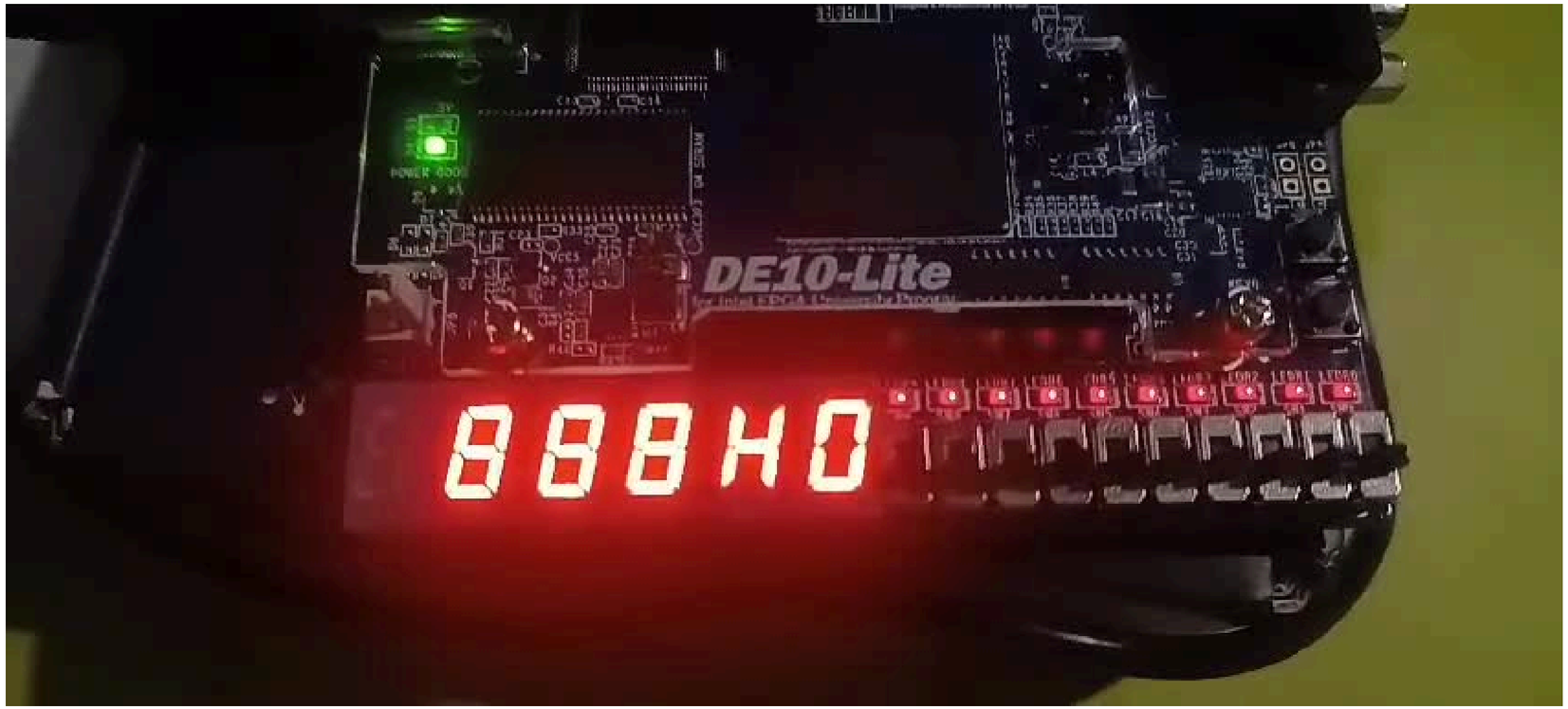
EJERCICIO A DESARROLLAR

```vhdl

library ieee;
use ieee.std_logic_1164.all;

-- La entidad 'Ejercicio03' combina varios modulos para desplegar el mensaje "HOLI PrOFE"
-- en un conjunto de displays de 7 segmentos, donde los caracteres se desplazan hacia la izquierda
-- en cada ciclo del reloj dividido.
entity Ejercicio03 is
  port (
    clk, reset                    : in std_logic;  -- Señales de reloj y reinicio de entrada
    hex0, hex1, hex2, hex3, hex4 : buffer std_logic_vector(6 downto 0)
      -- Salidas para los displays de 7 segmentos
  );
end entity;

architecture arqmain of Ejercicio03 is
  signal clkl : std_logic;  -- Señal de reloj dividida que controla la velocidad de desplazamiento del mensaje
  signal bcd  : std_logic_vector(3 downto 0) := "0000";  -- Señal BCD que representa el carácter actual a mostrar
begin
  -- Proceso 1: División de frecuencia
  -- El módulo 'divfreq' divide la frecuencia del reloj de entrada 'clk' para obtener un reloj más lento 'clkl',
  -- que controla la velocidad de desplazamiento del mensaje en los displays.
  u1 : entity work.divfreq(rtl) port map(clk, clkl);

  -- Proceso 2: Contador de estados
  -- El módulo 'conta' actúa como un contador secuencial que cambia entre los estados definidos,
  -- representando las posiciones de cada carácter del mensaje "HOLI PrOFE".
  -- El contador avanza en cada flanco de subida del reloj 'clkl'.
  -- La salida del contador (bcd) es usada para seleccionar qué carácter se mostrará en el display correspondiente.
  u2 : entity work.conta(arqconta) port map(clkl, reset, bcd);

  -- Proceso 3: Decodificación de caracteres
  -- El módulo 'ss7' decodifica la señal BCD a un patrón de 7 segmentos,
  -- que representa un carácter específico (H, O, L, I, espacio, P, r, O, F, E) en el display.
  -- Cada carácter del mensaje es decodificado y enviado a uno de los displays.
  u3 : entity work.ss7(arqss7) port map(bcd, hex4);

  -- Procesos 4-7: Despliegue en displays con desplazamiento
```

EJERCICIO A DESARROLLAR

```
38    -- Los módulos 'display' se utilizan para crear un efecto de
         desplazamiento.
39    -- Cada vez que el reloj 'clkl' 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
         PrOFE" en secuencia.
41    u4 : entity work.display(arqdisp) port map(clkl, hex4, hex3);
42    u5 : entity work.display(arqdisp) port map(clkl, hex3, hex2);
43    u6 : entity work.display(arqdisp) port map(clkl, hex2, hex1);
44    u7 : entity work.display(arqdisp) port map(clkl, hex1, hex0);
45
46  end architecture;
```

VIDEO DE FUNCIONAMIENTO


DE10-Lite
888H0

REFERENCIAS

PEDRONI, V. A. (2004). DIGITAL DESIGN AND MODELING WITH VHDL AND SYNTHESIS: AN INTEGRATED APPROACH. MIT PRESS

NAVABI, Z. (2007). VHDL: MODULAR DESIGN AND SYNTHESIS OF CORES AND SYSTEMS. MCGRAW-HILL EDUCATION.J

RNAVDOM. (2013, 21 SEPTIEMBRE). CONTROLAR REGISTROS DE DESPLAZAMIENTO DESDE FPGA (VHDL). FOROS DE ELECTRÓNICA. HTTPS://WWW.FOROSDEELECTRONICA.COM/THREADS/CONTROLAR-REGISTROS-DE-DESPLAZAMIENTO-DESDE-FPGA-VHDL.105025/

GRACIAS POR SU ATENCIÓN