DISEÑO
DIGÍTAL VLSI

PRÁCTICA 6

EJERCICIO 7.4

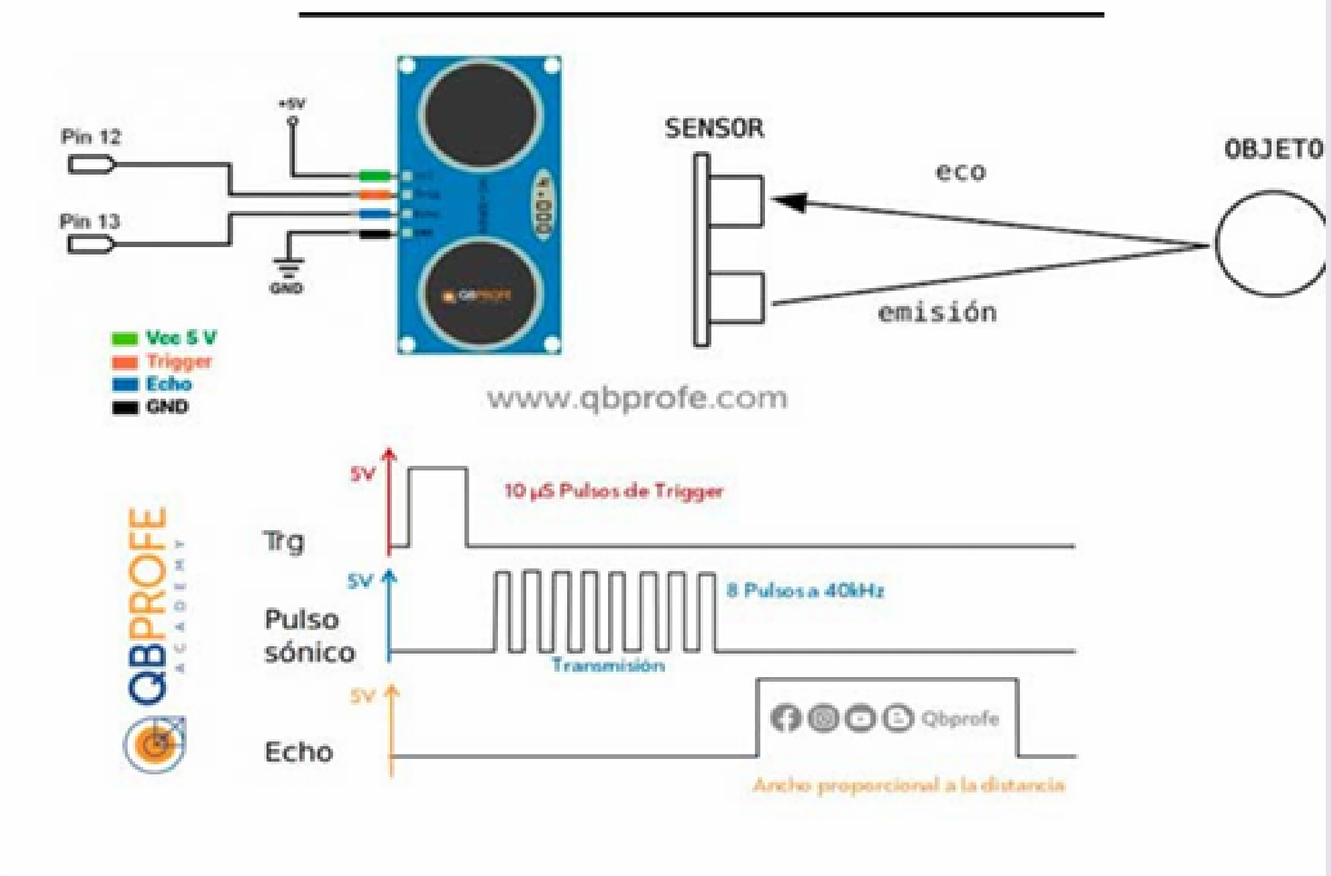
Alcalá Briseño Martha Alondra
Cadena Luna Iván Adrián
Limón Sosa Zair Odin

Grupo: 03
Equipo: 10

OBJETIVO

EL ALUMNO APRENDERÁ A DISEÑAR EL CONTROL DE UN SENSOR ULTRASÓNICO (HC-SR04)

INSTRUCCIONES



Mostrar la distancia a la que se encuentre el objeto enel display de 7 segmentos

ARCHIVOS UTILIZADOS

DISPLAY.VHD

Divide la frecuencia de un reloj, alternando la salida cada cierto número de ciclos.

BCD2SS7.VHD

Convierte un número decimal a su representación en binario, distribuyendo cada dígito en un formato de 4 bits

DIVF.VHD

Divide la frecuencia de un reloj, alternando la salida cada cierto número de ciclos

MAIN.VHD

Controla un sensor ultrasónico y muestra la distancia en tres displays de 7 segmentos, con salida binarria en LEDs.

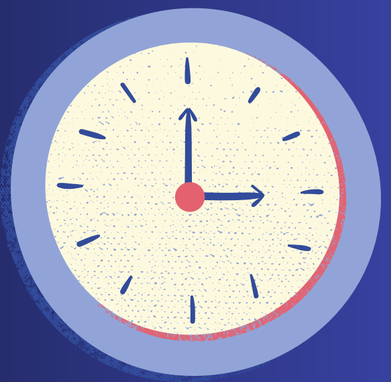
ULTRASONIC.VHD

Implementa la lógica de control del sensor ultrasónico para medir la distancia en centímetros

ENTIDAD DIVF

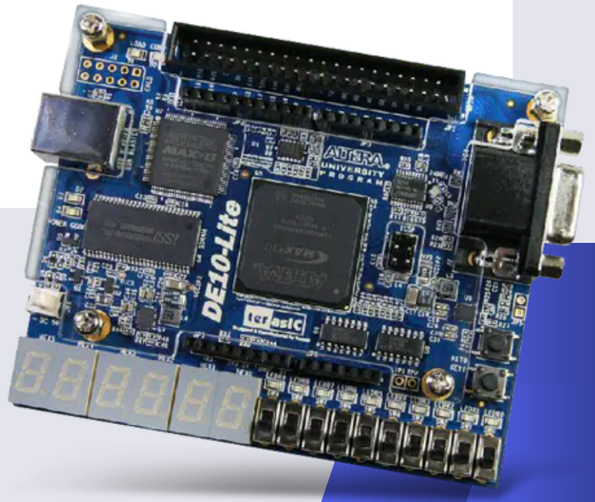
```
--MODULO DIVF
-- Este bloque reduce la frecuencia de la señal de reloj de entrada,
-- generando una señal de salida con una frecuencia más baja, útil para otros componentes del sistema.
-- La reducción de frecuencia se logra mediante un contador que cuenta hasta un valor predefinido.
library ieee;
use ieee.std_logic_1164.all;

-- Definición de la entidad divf.
-- num: Este parámetro define el número máximo de ciclos que el contador contará,
-- lo que determinará la división de frecuencia.
entity divf is
  generic (num : integer := 25000000);-- Divisor de frecuencia, en este caso 25 millones.
  port (
    clk  : in std_logic;--Señal de reloj de entrada, de alta frecuencia.
    clkl : buffer std_logic := '0' --Señal de reloj de salida, con frecuencia reducida.
  );
end entity;
```



ARQUITECTURA ARQDIVF

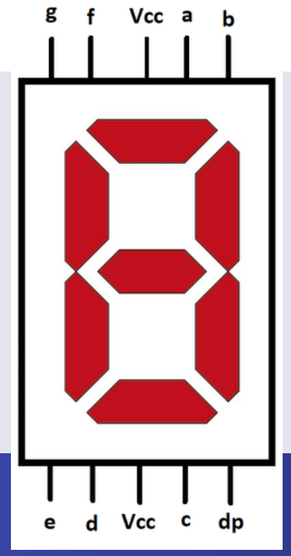
```vhdl
-- Arquitectura del divisor de frecuencia.
-- Un contador incrementa en cada flanco de subida del reloj, y cuando llega a su valor máximo (num),
-- invierte la señal de salida y reinicia el contador.
architecture arqdivf of divf is
  signal conteo : integer range 0 to num; -- Señal para contar el número de ciclos del reloj.
begin
-- Proceso que se activa en cada flanco de subida del reloj (clk).
  process (clk)
  begin
    if (rising_edge(clk)) then --Verifica si hay un flanco de subida en la señal de reloj
      if (conteo = num) then --Si el contador ha llegado al valor máximo
        conteo <= 0; --Reinicia el contador
        clkl   <= not clkl; --Invierte la señal de salida, generando una frecuencia más baja.
      else
        conteo <= conteo + 1; -- Incrementa el contador en cada ciclo de reloj.
      end if;
    end if;
  end process;
end architecture;
```



ENTIDAD DISPLAY

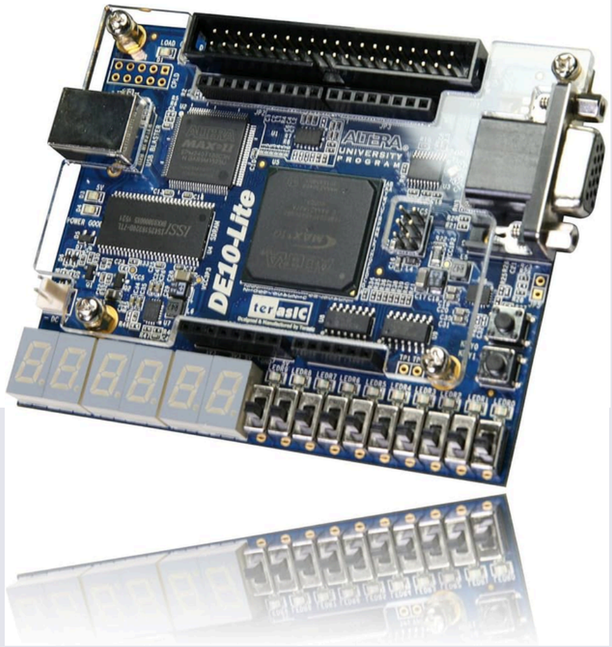
```vhdl
--MODULO DISPLAY
-- Este módulo convierte un número en formato BCD a la representación
--binaria necesaria para controlar un display de 7 segmentos. La salida
--es un vector de 7 bits que representa el estado de los segmentos (a-g) en el display.
library ieee;
use ieee.std_logic_1164.all;

-- Definición de la entidad display.
-- bcd: Entrada que representa el dígito en formato BCD (de 0 a 9).
-- hex: Salida que contiene la representación binaria de los segmentos del display de 7 segmentos.
entity display is
  port (
    bcd : in std_logic_vector(3 downto 0);--Entrada BCD, que representa un dígito en binario (4 bits).
    hex : out std_logic_vector(6 downto 0)--Salida para el display de 7 segmentos (7 bits, para los segmentos a-g).
  );
end entity;
```



ARQUITECTURA ARQDISPLAY

```
--Arquitectura "arqdisplay"
-- Se utiliza una sentencia "with select" para mapear el valor BCD de entrada a la configuración
-- correspondiente de los segmentos del display.
architecture arqdisplay of display is
begin
-- Proceso de conversión BCD a 7 segmentos.
  -- Dependiendo del valor de la entrada BCD, se asignan los valores binarios correspondientes
  -- a la salida "hex", que representa el estado de cada uno de los 7 segmentos.
  -- El display está representado por los bits hex(6 downto 0), que corresponden a los segmentos a, b, c, d, e, f, g.
  with bcd select
    hex <=
    "1000000" when "0000", -- 0
    "1111001" when "0001", -- 1
    "0100100" when "0010", -- 2
    "0110000" when "0011", -- 3
    "0011001" when "0100", -- 4
    "0010010" when "0101", -- 5
    "0000010" when "0110", -- 6
    "1111000" when "0111", -- 7
    "0000000" when "1000", -- 8
    "0011000" when "1001", -- 9
    "1111111" when others;
end architecture;
```



ENTIDAD ULTRASONIC

```vhdl
--MODULO ULTRASONIC
-- Este módulo controla el sensor de ultrasonido HC-SR04 mediante una Máquina de Estados Finita (FSM).
-- El sensor de ultrasonido mide la distancia basándose en el tiempo que tarda el eco en volver tras el disparo del trigger.
-- La FSM se encarga de enviar el trigger, esperar el eco y calcular la distancia.
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

-- Definición de la entidad ultrasonic.Entidad para controlar el sensor HC-SR04 utilizando la FPGA DE10 Lite
-- clk: Señal de reloj del sistema, generalmente a 50 MHz.
-- reset: Señal de reinicio para restablecer el estado del sistema.
-- echo: Señal de entrada que proviene del sensor ultrasonido, indicando el tiempo del eco.
-- trigger: Señal de salida que activa el sensor (envía el pulso).
-- distance: Salida que contiene la distancia medida en centímetros[2].
entity ultrasonic is
  port (
    clk      : in std_logic; -- Reloj principal del sistema a 50[MHz]
    reset    : in std_logic; -- Señal de reset que reinicia el sistema.
    echo     : in std_logic; -- Señal de entrada del eco del sensor de ultrasonido.
    trigger  : out std_logic; -- Señal de salida que activa el sensor.
    distance : out integer := 0 -- Distancia medida en centímetros, calculada a partir del tiempo del eco (CM)
  );
end entity ultrasonic;
```

ARQUITECTURA ARCHULTRASONIC

```
-- Arquitectura que implementa la FSM  para controlar el sensor de ultrasonido.
-- Se definen dos estados: Wait_state (espera) y Echo_state (medición del eco).
architecture archultrasonic of ultrasonic is
-- Definición de los posibles estados de la FSM.
  type state_type is (Wait_state, Echo_state);
  signal PS                    : state_type := Wait_state;-- Estado presente (PS: Present State).
  signal NS                    : state_type := Wait_state;-- Próximo estado (NS: Next State).
  signal cuenta                : integer    := 0;-- Contador que se utiliza para medir el tiempo y la distancia.
  signal centimeters           : integer    := 0;-- Variable que almacena la distancia medida en centímetros.
  signal distance_out          : integer    := 0;-- Señal de salida que contiene la distancia calculada.
  signal past_echo, sync_echo : std_logic  := '0';-- Señales auxiliares para detectar cambios en el eco.
```

ARQUITECTURA ARCHULTRASONIC

```
-- Proceso que controla la transición de estados de la FSM.
-- En cada ciclo de reloj, se evalúa si la FSM debe cambiar de estado.
state_machine : process (clk, reset)
begin
  if reset = '0' then -- Si se recibe una señal de reset, se vuelve al estado de espera.
    PS <= Wait_state;
  elsif rising_edge(clk) then -- En cada flanco de subida del reloj, se evalúa el próximo estado.
    PS <= NS; -- El estado actual se actualiza al próximo estado.
  end if;
end process;
```

```
-- Proceso que maneja las señales de eco, sincronizando el eco con el reloj del sistema.
echo_inputs : process (clk)
begin
  if rising_edge(clk) then
    past_echo <= sync_echo;-- Guarda el valor anterior de la señal de eco.
    sync_echo <= echo; -- Sincroniza la señal de eco con el reloj.
  end if;
end process;
```

ARQUITECTURA ARCHULTRASONIC

```
-- Proceso principal de la FSM que controla el sensor de ultrasonido
  ultrasonic_sensor : process (clk, cuenta, past_echo, sync_echo, centimeters)
  begin
    if rising_edge(clk) then
      case PS is
      -- Estado de espera (Wait_state):
        -- En este estado, el sistema espera que el contador alcance un valor determinado
        -- antes de enviar la señal de trigger.
        when Wait_state =>
          if cuenta >= 500 then -- Si el contador ha alcanzado el valor de espera (500 ciclos).
            trigger <= '0'; -- Desactiva la señal de trigger (el pulso de activación del sensor se ha enviado).
            NS      <= Echo_state; -- Cambia al estado de medición del eco.
            cuenta  <= 0; -- Reinicia el contador.
          else
            trigger <= '1'; -- Activa la señal de trigger, enviando el pulso al sensor.
            NS      <= Wait_state;-- Permanece en el estado de espera.
            cuenta  <= cuenta + 1; -- Incrementa el contador en cada ciclo de reloj.
          end if;
```

ARQUITECTURA ARCHULTRASONIC

```
        -- Estado de medición del eco (Echo_state):
       -- En este estado, el sistema mide el tiempo que tarda en llegar la señal de eco.
       when Echo_state =>
         if past_echo = '0' and sync_echo = '1' then --Detecta el flanco de subida del eco.
           cuenta      <= 0; -- Reinicia el contador al detectar el inicio del eco.
           centimeters <= 0; -- Reinicia el cálculo de la distancia.
         elsif past_echo = '1' and sync_echo = '0' then -- Detecta el flanco de bajada del eco.
           distance_out <= centimeters; -- Almacena la distancia medida cuando termina el eco.
           cuenta       <= 0;  -- Reinicia el contador.
         elsif cuenta >= 2900 then -- Cada 2900 ciclos equivalen a 1 cm (basado en la frecuencia del reloj de 50 MHz).
           if centimeters >= 3448 then -- Limita la distancia máxima a 3448 cm.
             NS      <= Wait_state; -- Si se supera la distancia máxima, vuelve al estado de espera.
             cuenta <= 0; -- Reinicia el contador.
           else
             centimeters <= centimeters + 1;-- Incrementa la distancia medida en 1 cm.
             cuenta      <= 0; -- Reinicia el contador.
           end if;
         else
           NS      <= Echo_state; -- Permanece en el estado de medición del eco.
           cuenta <= cuenta + 1; -- Desactiva la señal de trigger mientras se mide el eco.
         end if;
         trigger <= '0';
     end case;
   end if;
 end process;
 distance <= distance_out; -- Asigna la distancia medida a la señal de salida "distance".
end architecture;
```

ENTIDAD BCD2SS7

```
--MODULO BCD2SS7
-- Este módulo convierte un número entero en su representación binaria
--en formato BCD (Binary-Coded Decimal).La salida es una cadena de 7
--segmentos que se puede mostrar en un display de 7 segmentos.
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.math_real.all;
-- Toma un número, por ejemplo 154, y retorna los dígitos en binario
-- ["0001", "0101", "0100"] pero en un arreglo único: "000101010100"


--Definición de la entidad bcd2ss7.
entity bcd2ss7 is
  port (
    number : in integer;--Entrada que representa el número que se convertirá a BCD.
    digits : out std_logic_vector(27 downto 0)--Salida que contiene los dígitos BCD, con 4 bits por cada dígito.
  );
end entity bcd2ss7;
```

SEGUROS DE GASTOS SEGURO DE VIDA SEGURO

ARQUITECTURA BCD2SS7

```vhdl
--Arquitetura "bcd2ss7arch"
-- Este proceso descompone el número de entrada en dígitos individuales BCD,
-- que son convertidos y almacenados en la salida
architecture bcd2ss7arch of bcd2ss7 is
  signal digits_sig : std_logic_vector(27 downto 0); -- Señal interna que almacena los dígitos convertidos.
begin
   -- Proceso que toma el número y los descompone en sus dígitos BCD
  process (number)
  variable temp, remaint : integer := 0; --Variables para manejar el número temporal y el dígito restante.
  begin
    digits_sig <= (others => '0');  -- Inicializa la salida en ceros.
    temp := number; -- Almacena el número en una variable temporal.
    for i in 0 to 6 loop -- Recorre hasta 7 dígitos (máximo 28 bits en BCD).

      -- Obtención del dígito
      remaint := temp mod 10; -- Obtiene el último dígito del número.
      temp    := temp / 10; -- Reduce el número eliminando el último dígito.

      -- Conversión a binario
      -- Asigna el dígito convertido a la señal de salida.
      digits_sig(i * 4 + 3 downto i * 4) <= std_logic_vector(to_unsigned(remaint, 4));
      exit when temp = 0;
    end loop;
  end process;
  digits <= digits_sig;
end architecture;
```

SEGUROS DE GASTOS MÉDICOS

Duis aute irure dolor in

SEGURO DE VIDA

Duis aute irure dolor in

SEGURO

Duis aute

EMPRESA
ENSIGNA

FORTALEZAS

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

OPORTUNIDADES

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

DEBILIDADES

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

AMENAZAS

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

```vhdl
--MODULO MAIN
--Este módulo principal conecta el sensor de ultrasonido, el convertidor de números a BCD (bcd2ss7),
-- y los displays de 7 segmentos. Gestiona las señales de entrada y salida, activando el sensor,
-- midiendo la distancia y mostrando los resultados en los displays.
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

-- Definición de la entidad principal main.
-- clk: Señal de reloj que controla todo el sistema.
-- reset: Señal que reinicia el sistema.
-- echo: Señal de entrada del eco desde el sensor de ultrasonido.
-- trigger: Señal de salida para activar el sensor de ultrasonido.
-- display1, display2, display3: Salidas para los displays de 7 segmentos que muestran las decenas, centenas y unidades.
-- led_out: Señal de salida en formato binario que muestra la distancia en LEDs (8 bits).
entity main is
  port (
    clk   : in std_logic; --Señal de reloj
    reset : in std_logic; --Señal de reinicio del sistema
    -- Parte sensor
    echo    : in std_logic; --! Entrada del echo del sensor
    trigger : out std_logic;-- Señal de activación del sensor (trigger).
    -- Displays
    display1 : out std_logic_vector(6 downto 0); --! Decenas
    display2 : out std_logic_vector(6 downto 0); --! Centenas
    display3 : out std_logic_vector(6 downto 0); --! Unidades
    led_out  : out std_logic_vector(7 downto 0) -- Señal de salida binaria que representa la distancia medida en LEDs.
  );
end entity main;
```

```vhdl
-- Arquitectura principal del sistema. Conecta las diferentes partes: sensor de ultrasonido, convertidor BCD y displays.
architecture arqmain of main is
  signal distance   : integer := 0; -- Señal interna para almacenar la distancia medida por el sensor.
  signal bin_digits : std_logic_vector(27 downto 0); -- Señal interna que almacena los dígitos BCD convertidos.

  -- En VHDL, se utiliza la declaración de "component" cuando queremos reutilizar un diseño previamente
  -- definido en otro módulo. Esta declaración nos permite instanciar dicho diseño (componente)
  -- dentro de otro, como en este caso. Declaramos los componentes para modularizar nuestro sistema,
  -- manteniendo cada funcionalidad separada y reutilizable.

  -- Declaración del componente del sensor de ultrasonido.
  -- Este componente mide la distancia y emite un eco.
  -- Utilizamos la declaración del "component" porque este sensor es un bloque de funcionalidad independiente,
  -- y nos permite reutilizar este módulo en diferentes partes del diseño sin tener que volver a escribir el código.
  component ultrasonic is
    port (
      clk      : in std_logic; -- Señal de reloj.
      reset    : in std_logic; -- Señal de reinicio.
      echo     : in std_logic; -- Señal de eco del sensor.
      trigger  : out std_logic; -- Señal de activación del sensor.
      distance : out integer -- Salida de la distancia medida en centímetros.
    );
  end component ultrasonic;

  -- Declaración del componente bcd2ss7, que convierte el número en formato binario
  -- a su representación en BCD para ser mostrada en los displays de 7 segmentos.
  -- Utilizamos este componente para separar la lógica de conversión de binario a BCD, lo que hace
  -- que el código sea más limpio y modular. Además, podemos reutilizar este componente en otros diseños
  -- que también requieran convertir números binarios a BCD.
  component bcd2ss7 is
    port (
      number : in integer; -- Número a convertir a BCD.
      digits : out std_logic_vector(27 downto 0) -- Salida en BCD (4 bits por dígito, hasta 7 dígitos).
    );
  end component bcd2ss7;
```

```vhdl
 -- Declaración del componente display, que convierte un dígito BCD
-- en una representación de 7 segmentos.
-- Aquí utilizamos el componente para manejar la lógica que convierte un dígito BCD
-- en la configuración de segmentos para un display de 7 segmentos. Esto nos permite reutilizar
-- este módulo en cada display y mantener el diseño más organizado.
component display is
  port (
    bcd : in std_logic_vector(3 downto 0); -- Entrada BCD (4 bits).
    hex : out std_logic_vector(6 downto 0) -- Salida para controlar un display de 7 segmentos.
  );
end component display;

begin
-- Asigna la distancia medida (en formato binario) a la salida de LEDs, representando la distancia en formato binario.
led_out <= std_logic_vector(to_unsigned(distance, 8));

-- Instancia del sensor de ultrasonido:
-- Aquí estamos creando una instancia del componente "ultrasonic".
-- Al instanciarlo, conectamos las señales internas de la entidad principal
-- a las entradas y salidas del componente.
-- Utilizamos "component" aquí porque queremos reutilizar un bloque de diseño específico (el sensor de ultrasonido).
-- Esto modulariza el diseño y permite una separación de responsabilidades, facilitando el mantenimiento.
sensor    : ultrasonic port map(clk, reset, echo, trigger, distance);

 -- Instancia del convertidor de número binario a BCD:
-- Al igual que con el sensor de ultrasonido, aquí estamos instanciando el componente "bcd2ss7".
-- Esto convierte la distancia medida en formato binario a BCD, que será utilizado
-- para mostrar la distancia en los displays de 7 segmentos.
-- Utilizamos el componente aquí para reutilizar el módulo de conversión, lo que facilita la organización
-- del código y permite que la lógica de conversión esté separada del resto del sistema.
converter : bcd2ss7 port map(distance, bin_digits);
```

```
  -- Display para las unidades (primeros 4 bits del vector BCD).
  -- Aquí instanciamos el componente "display" para controlar un display de 7 segmentos.
  -- Utilizamos componentes porque el módulo de control del display puede ser reutilizad
  -- en diferentes partes del sistema sin duplicar la lógica.
  digit1    : display port map(bin_digits(3 downto 0), display1);

   -- Display para las decenas (siguientes 4 bits del vector BCD).
  digit2    : display port map(bin_digits(7 downto 4), display2);

  -- Display para las centenas (siguientes 4 bits del vector BCD).
  digit3    : display port map(bin_digits(11 downto 8), display3);
end architecture;
```

TABLA COMPARATIVA

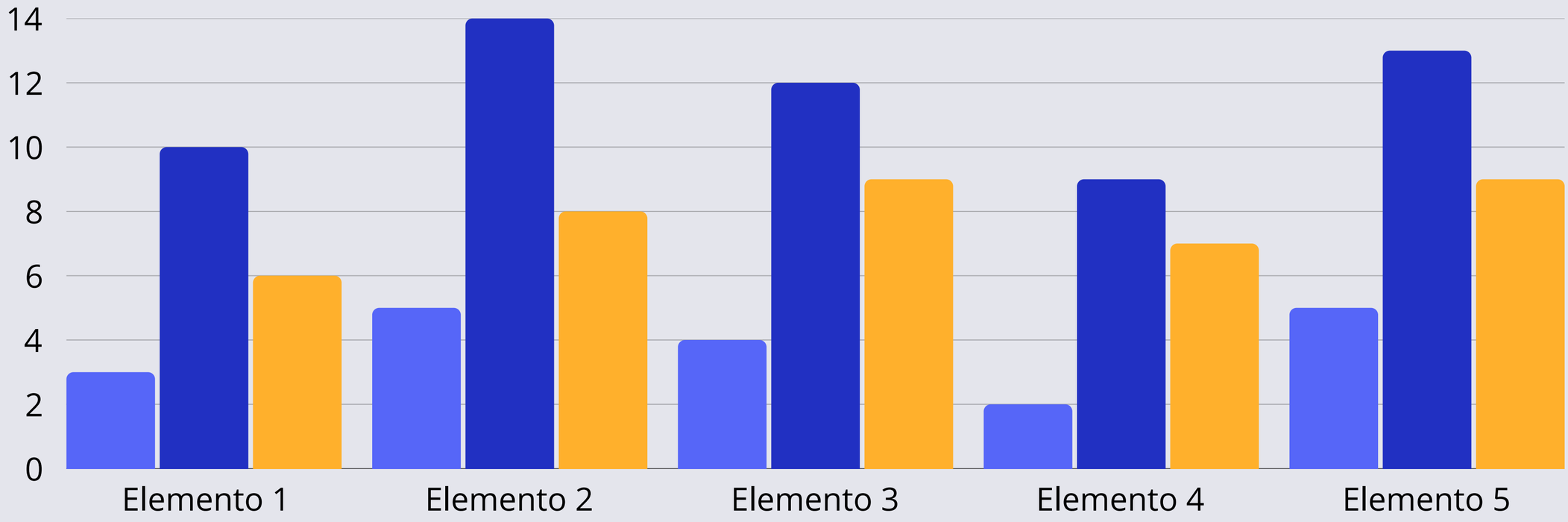
EMPRESA ENSIGNA

NUESTRAS EMPRESA	LA COMPETENCIA
• Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.	• Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
• Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.	• Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
• Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.	• Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

ESTADÍSTICAS

EMPRESA ENSIGNA

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam.



PLAN
DE ACCIÓN

EMPRESA
ENSIGNA

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam.

INVESTIGACIÓN

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

DESARROLLO

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

IMPLEMENTACIÓN

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

RESULTADO

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.