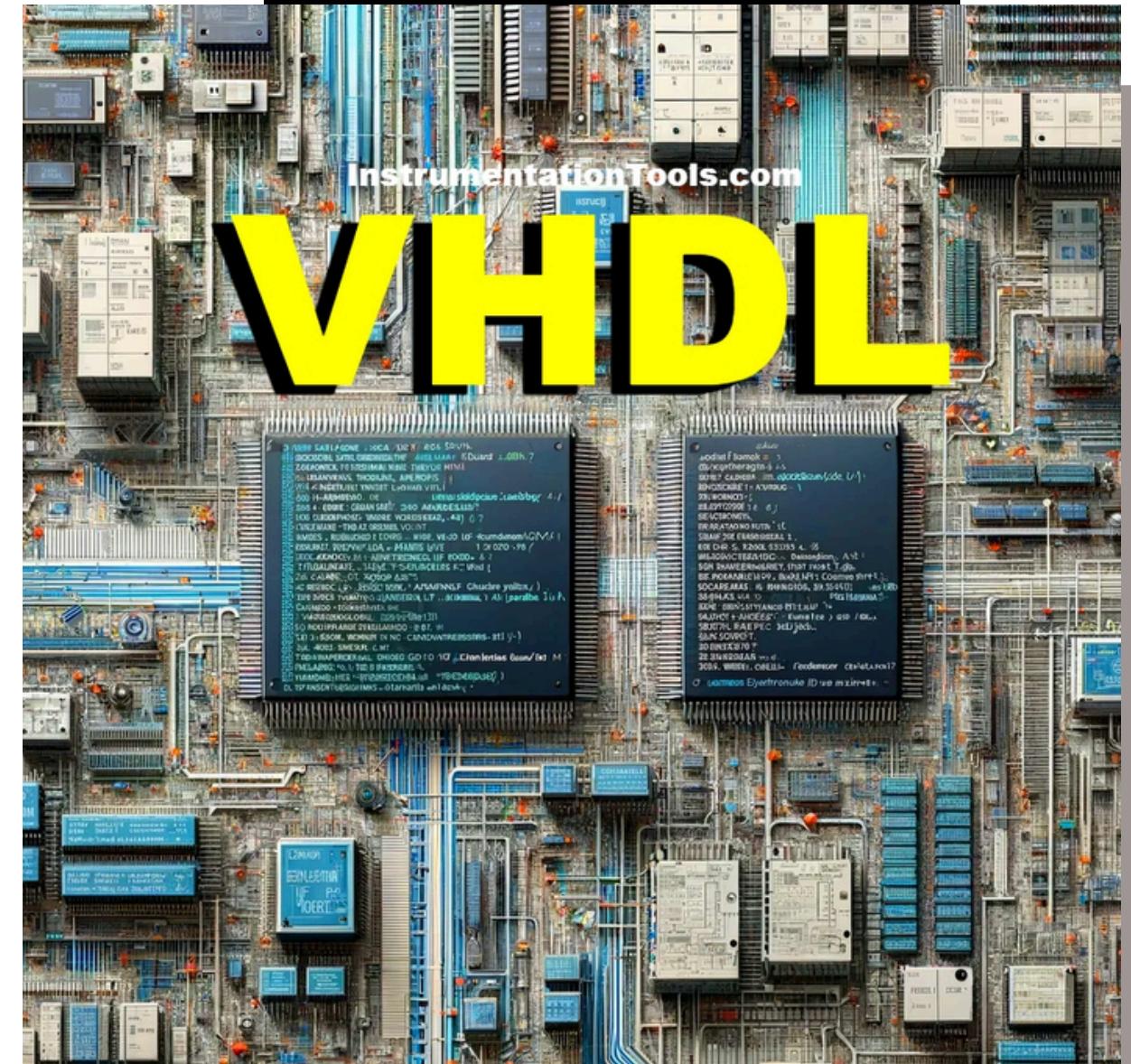


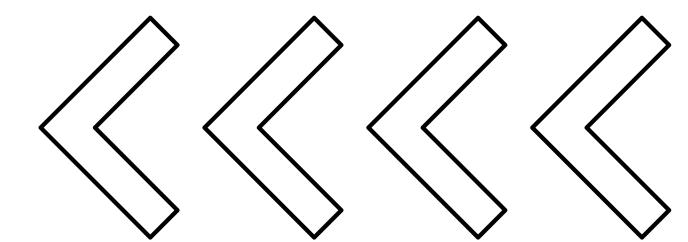
•APLICACIÓN DE VHDL CON FPGA

SENSOR DE TEMPERATURA

Presentada por Grupo C:

**ANTHONY YAGUANA
STEFANO CRUZ
MARÍA DEL CISNE VALAREZO
GERARDO NAULA**





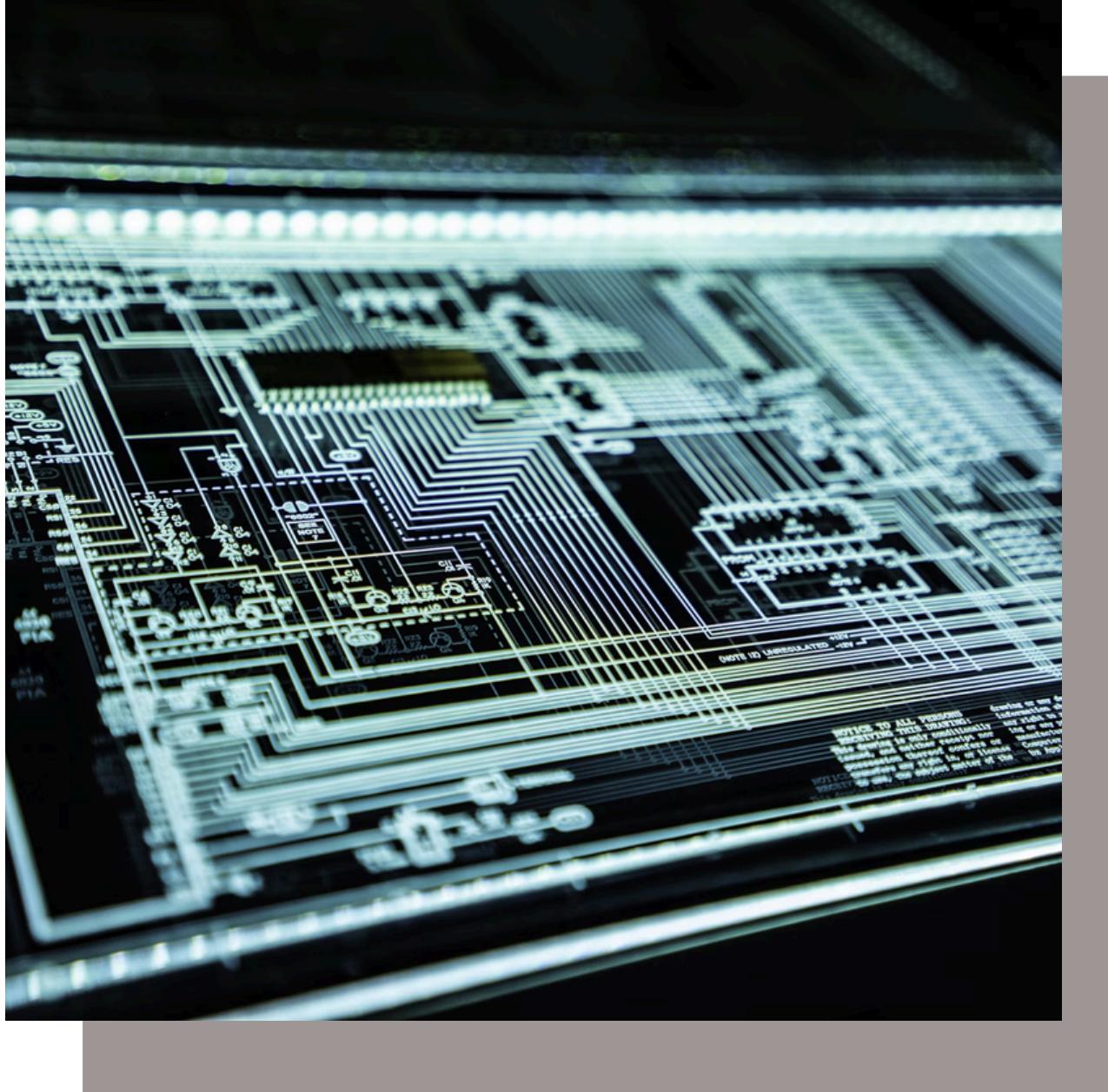
AGENDA

1

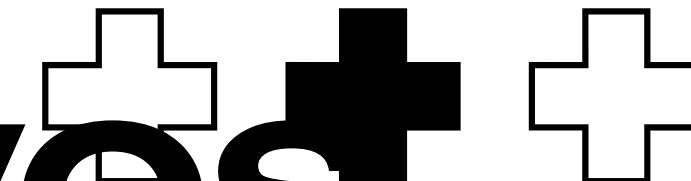
OBJETIVOS
INTODUCCION:

2

PROYECTO



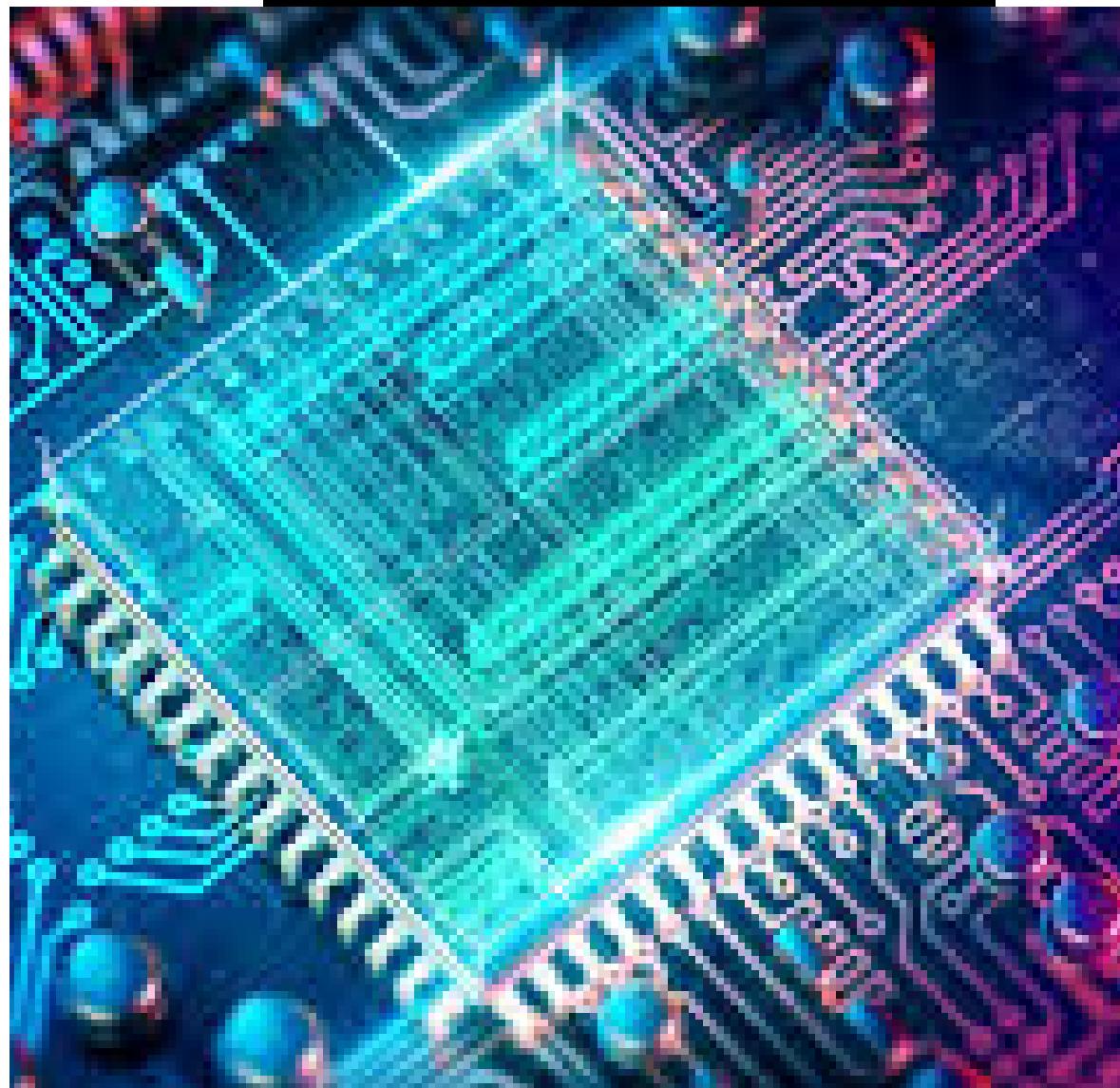
OBJETIVOS



Diseñar e implementar un sistema sensor de temperatura con la ayuda del procesador FPGA utilizando el lenguaje de hardware VHDL, capaz de procesar señales de temperatura provenientes de un sensor analógico LM35 mediante el uso de un convertidor analógico a digital (ADC0804).

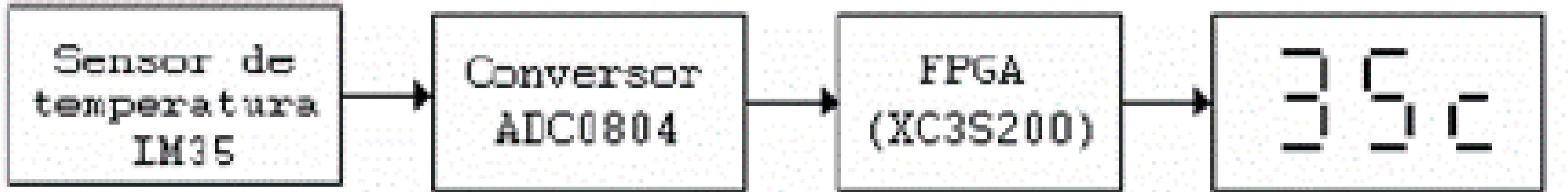
Optimizar la funcionalidad y precisión del termómetro garantizando una conversión eficiente de las señales analógicas a digitales y su correcta interpretación en la FPGA para mostrar lecturas de los valores adecuados en tiempo real.

INTRODUCCIÓN

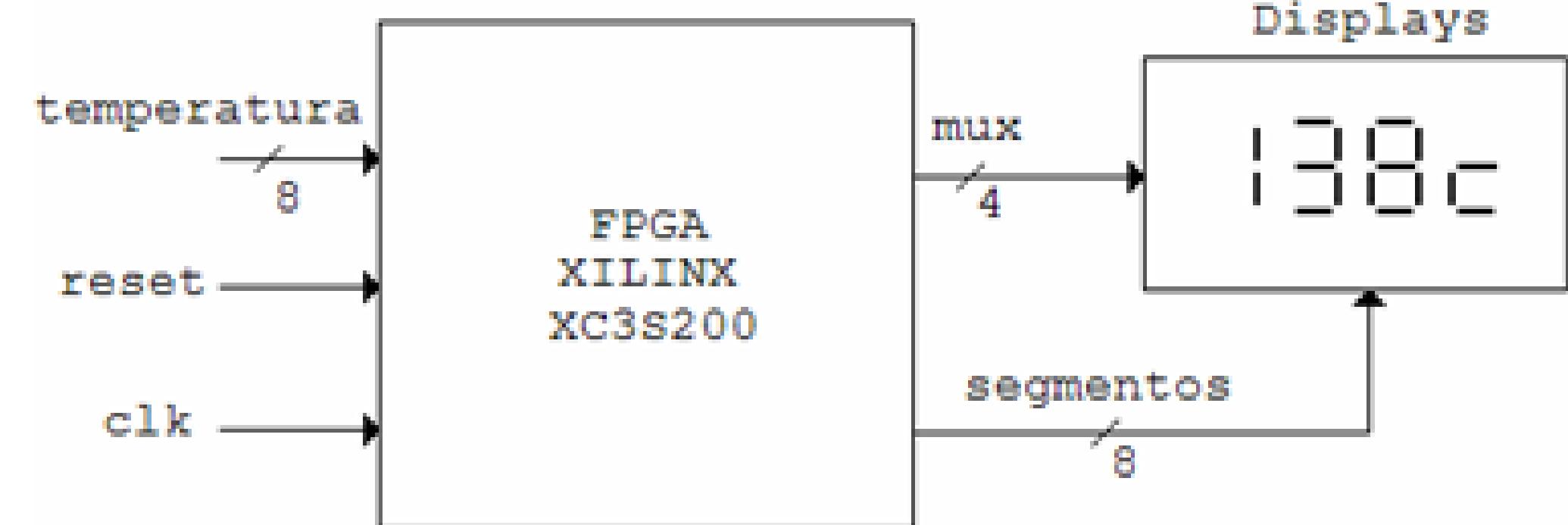
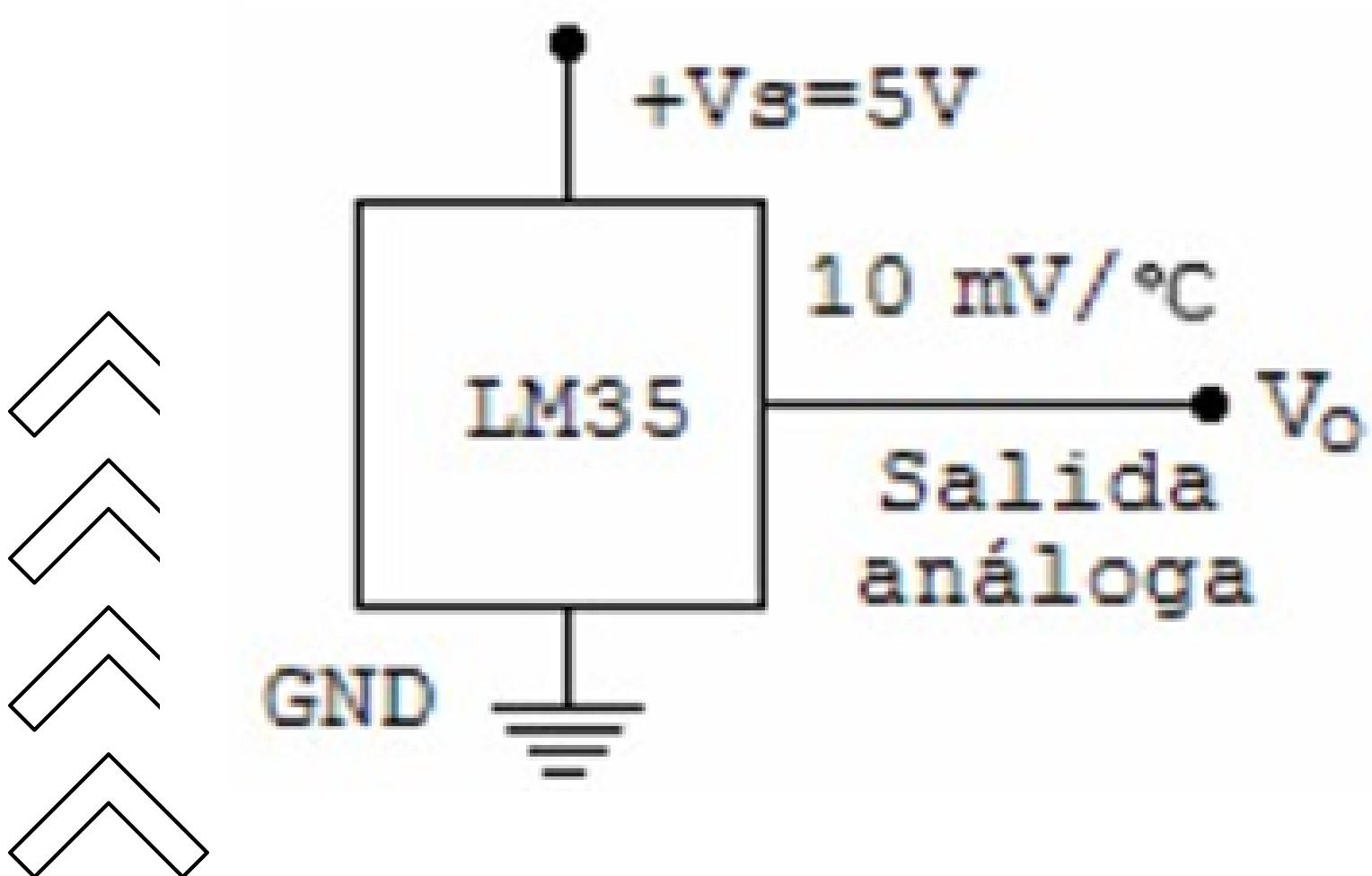


En este proyecto, se presenta el desarrollo de un termómetro digital implementado utilizando una FPGA. El trabajo se centra en el diseño y la programación de la parte digital del sistema utilizando el lenguaje VHDL, una herramienta fundamental para describir la estructura y el comportamiento de los circuitos digitales.

DIAGRAMA PROYECTO:



Visualización



CODIGO VHDL

SIMULATION - Behavioral Simulation - Functional - sim_1 - sensor

sensor.vhd x sensor_behav.wcfg*

C:/Users/USER/project_1/srcs/sources_1/new/sensor.vhd

Scope Sources Objects

```
19 library IEEE;
20 use IEEE.STD_LOGIC_1164.ALL;
21 use IEEE.NUMERIC_STD.ALL;
22
23 entity sensor is
24     Port (
25         clk : in STD_LOGIC; -- Reloj
26         reset : in STD_LOGIC; -- Reset
27         temperatura : in STD_LOGIC_VECTOR(7 downto 0); -- Entrada de temperatura (8 bits)
28         segmentos : out STD_LOGIC_VECTOR(7 downto 0); -- Salida para los segmentos
29         mux : out STD_LOGIC_VECTOR(3 downto 0) -- Salida para el multiplexor
30     );
31 end sensor;
32
33 architecture Behavioral of sensor is
34     signal cont : integer range 0 to 400_000 := 0; -- Contador
35     signal dig1_s, dig2_s, dig3_s : integer range 0 to 9 := 0; -- Dígitos individuales
36     signal dig_mux : integer range 0 to 10 := 0; -- Selector del dígito
37     signal mux_reg : STD_LOGIC_VECTOR(3 downto 0) := "1111"; -- Registro del multiplexor
38     signal seg_reg : STD_LOGIC_VECTOR(7 downto 0) := "11111111"; -- Registro de segmentos
39
40 begin
41
42     -- Proceso para calcular los dígitos individuales de la temperatura
43     temp : process(temperatura)
44         variable aux : integer range 0 to 255;
45         variable dig1, dig2, dig3 : integer range 0 to 9;
46     begin
47         aux := to_integer(unsigned(temperatura)); -- Conversión de temperatura a entero
48         dig1 := 0;
49         for i in 0 to 9 loop
50             if aux >= 100 then
51                 dig1 := 1;
52             elsif aux >= 20 then
53                 dig1 := 2;
54             elsif aux >= 30 then
55                 dig1 := 3;
56             elsif aux >= 40 then
57                 dig1 := 4;
58             elsif aux >= 50 then
59                 dig1 := 5;
60             elsif aux >= 60 then
61                 dig1 := 6;
62             elsif aux >= 70 then
63                 dig1 := 7;
64             elsif aux >= 80 then
65                 dig1 := 8;
66             else
67                 dig1 := 9;
68             end if;
69             aux := aux mod 100;
70             if aux >= 90 then
71                 dig2 := 1;
72             elsif aux >= 80 then
73                 dig2 := 2;
74             elsif aux >= 70 then
75                 dig2 := 3;
76             elsif aux >= 60 then
77                 dig2 := 4;
78             elsif aux >= 50 then
79                 dig2 := 5;
80             elsif aux >= 40 then
81                 dig2 := 6;
82             elsif aux >= 30 then
83                 dig2 := 7;
84             elsif aux >= 20 then
85                 dig2 := 8;
86             else
87                 dig2 := 9;
88             end if;
89             aux := aux mod 10;
90             if aux >= 9 then
91                 dig3 := 1;
92             elsif aux >= 8 then
93                 dig3 := 2;
94             elsif aux >= 7 then
95                 dig3 := 3;
96             elsif aux >= 6 then
97                 dig3 := 4;
98             elsif aux >= 5 then
99                 dig3 := 5;
100            else
101                dig3 := 6;
102            end if;
103        end loop;
104        dig1_s := dig1;
105        dig2_s := dig2;
106        dig3_s := dig3;
107        dig_mux := dig_mux + 1;
108        if dig_mux > 10 then
109            dig_mux := 0;
110            mux_reg := "1111";
111        else
112            mux_reg := "11111111";
113        end if;
114    end process;
115
116    -- Asignación de señales
117    temperatura <=> temperatura;
118    segmentos <=> segmentos;
119    mux <=> mux;
120
```

Activar Windows
Ve a Configuración para activar Windows.

CODIGO VHDL

SIMULATION - Behavioral Simulation - Functional - sim_1 - sensor

sensor.vhd x sensor_behav.wcfg*

C:/Users/USER/project_1/srcs/sources_1/new/sensor.vhd

Scope Sources Objects

```
temp : process(temperatura)
  variable aux : integer range 0 to 255;
  variable dig1, dig2, dig3 : integer range 0 to 9;
begin
  aux := to_integer(unsigned(temperatura)); -- Conversión de temperatura a entero
  dig1 := 0;
  for i in 0 to 9 loop
    if aux >= 100 then
      aux := aux - 100;
      dig1 := dig1 + 1;
    end if;
  end loop;

  dig2 := 0;
  for j in 0 to 9 loop
    if aux >= 10 then
      aux := aux - 10;
      dig2 := dig2 + 1;
    end if;
  end loop;

  dig3 := aux;

  dig1_s <= dig1;
  dig2_s <= dig2;
  dig3_s <= dig3;
end process temp;

-- Proceso secuencial para manejar el multiplexor
secuencial : process(clk, reset)
begin
  if reset = '1' then
    dig1_s <= 0;
    dig2_s <= 0;
    dig3_s <= 0;
  elsif rising_edge(clk) then
    if dig1 > 9 then
      dig1_s <= 0;
    else
      dig1_s <= dig1;
    end if;
    if dig2 > 9 then
      dig2_s <= 0;
    else
      dig2_s <= dig2;
    end if;
    if dig3 > 9 then
      dig3_s <= 0;
    else
      dig3_s <= dig3;
    end if;
  end if;
end process secuencial;
```

Activar Windows

CÓDIGO VHDL

sensor.vhd x sensor_behav.wcfg*

:/Users/USER/project_1/project_1.srcts/sources_1/new/sensor.vhd

Q | B | ← | → | X | ⌂ | X | // | ⏷ | ⓘ | ⚙

```
70
71      -- Proceso secuencial para manejar el multiplexor
72  secuencial : process(clk, reset)
73  begin
74    if reset = '1' then
75      cont <= 0;
76      mux_reg <= "1111";
77    elsif rising_edge(clk) then
78      if cont >= 400_000 then
79        cont <= 0;
80      else
81        cont <= cont + 1;
82      end if;

83
84      case cont is
85        when 100_000 => mux_reg <= "0111";
86        when 200_000 => mux_reg <= "1011";
87        when 300_000 => mux_reg <= "1101";
88        when 400_000 => mux_reg <= "1110";
89        when others => mux_reg <= mux_reg;
90      end case;
91    end if;
92  end process secuencial;

93
94  -- Proceso para decodificar los digitos a segmentos
95  decodificador : process(dig_mux)
96  begin
97    if dig_mux is
98      when 0 => seg_reg <= "10000001";
99      when 1 => seg_reg <= "11001111";
100     when 2 => seg_reg <= "10010010";
101     when 3 => seg_reg <= "10000110".
```

CÓDIGO VHDL

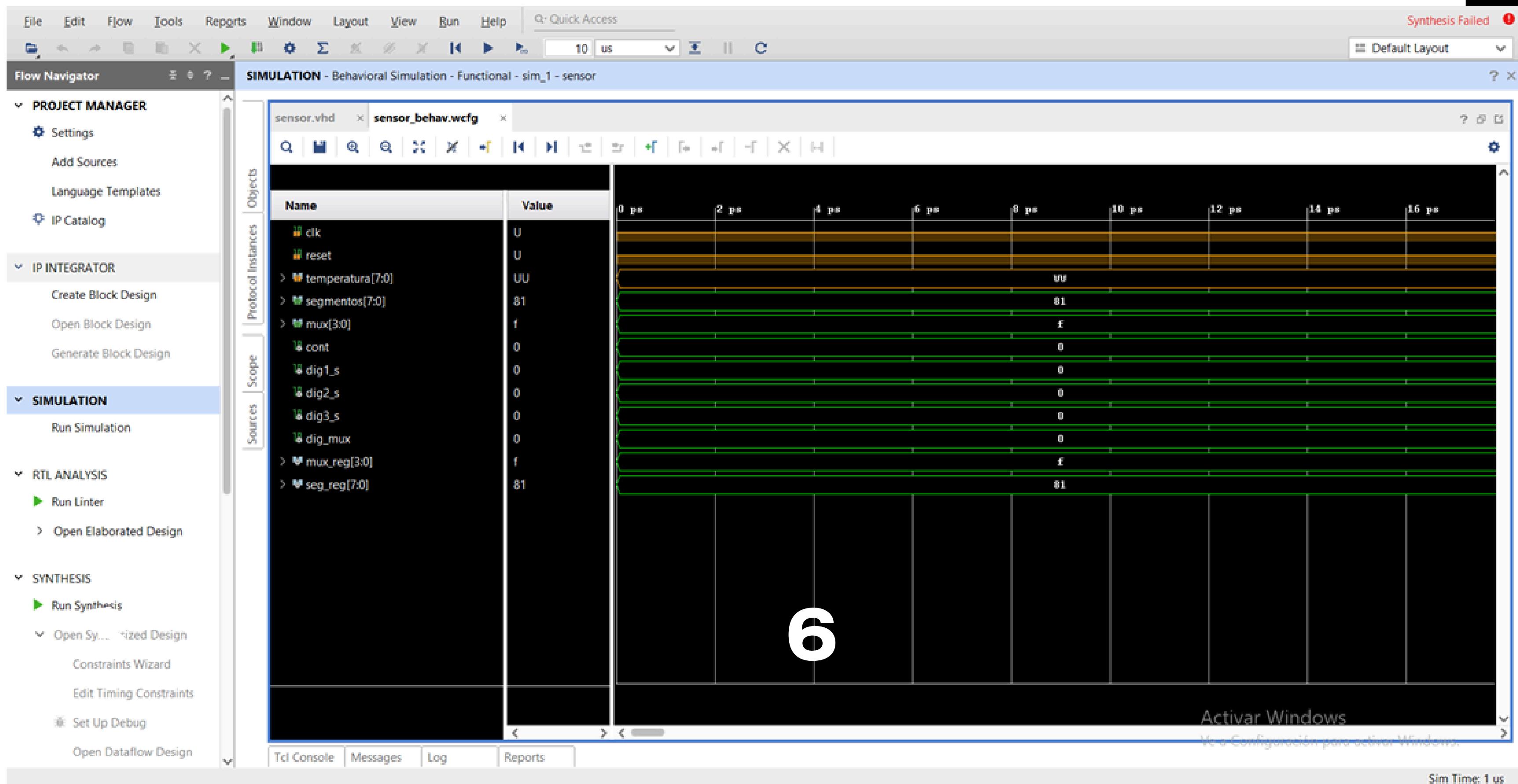
sensor.vhd x sensor_behav.wcfg*

C:/Users/USER/project_1/project_1.srcts/sources_1/new/sensor.vhd

Q | H | ← | → | X | D | F | X | // | E | ? | ↻

```
98  O      when 0 => seg_reg <= "10000001";
99  O      when 1 => seg_reg <= "11001111";
100 O      when 2 => seg_reg <= "10010010";
101 O      when 3 => seg_reg <= "10000110";
102 O      when 4 => seg_reg <= "11001100";
103 O      when 5 => seg_reg <= "10100100";
104 O      when 6 => seg_reg <= "10100000";
105 O      when 7 => seg_reg <= "10001111";
106 O      when 8 => seg_reg <= "10000000";
107 O      when 9 => seg_reg <= "10000100";
108 O      when 10 => seg_reg <= "11110010"; -- Representación de "-"
109 O      when others => seg_reg <= "11111111"; -- Vacío
110 O      end case;
111 O      end process decodificador;
112
113      -- Proceso para seleccionar el dígito que se muestra
114 O      selector : process(dig1_s, dig2_s, dig3_s, mux_reg)
115      begin
116 O          case mux_reg is
117 O              when "0111" => dig_mux <= dig1_s;
118 O              when "1011" => dig_mux <= dig2_s;
119 O              when "1101" => dig_mux <= dig3_s;
120 O              when "1110" => dig_mux <= 10; -- Separador decimal
121 O              when others => dig_mux <= 0;
122 O          end case;
123 O      end process selector;
124
125      -- Asignaciones finales a las salidas
126 O      mux <= mux_reg;
127 O      segmentos <= seg_reg;
128
129 O      end Behavioral.
```

SIMULACION VHDL



SIMULACION VHDL

