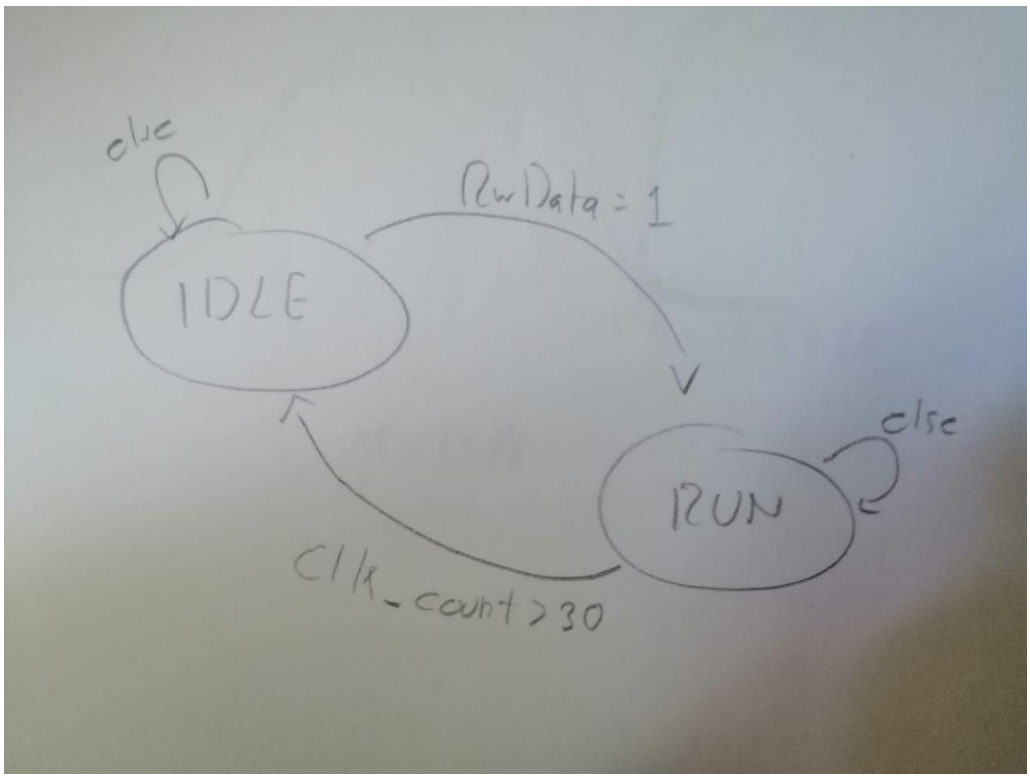
Sergio Gael Martínez Sarmiento A00825769

En este trabajo se elaboró el código en vhdl para un controlador LCD, mediante una máquina de estados con 2 estados: IDLE y RUN. El testbench se hizo con el propósito para que pudiese leer un archivo de texto, y así se le asignara un valor a cada entrada.

Me basé en la siguiente máquina de estados:



Código VHDL del Controlador LCD:

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


entity LCD is
port(RESET: in bit;
    CLK : in bit;

    RS: in bit;
    RWDATA: in bit;
    DATA_INSTRUCTIONS: in bit_vector(7 downto 0);

    Señal_RS: out bit;
    Señal_RW: out bit;
    Señal_EN: out bit;
    DATA: out bit_Vector(7 downto 0));
end entity LCD;

architecture Behavioral of LCD is
type state is(IDLE,RUN);
signal estado: state;

begin

process(CLK)
 variable clk_count: integer := 0;
 begin
    if CLK='1' and CLK'event then
        case estado is
            when IDLE=>
```

```
32
33                  if RESET = '1' then
34                    Señal_RS <= '0';
35                    Señal_RW <= '0';
36                    Señal_EN <= '0';
37                    DATA<="00000000";
38                    estado <= IDLE;
39                  elsif RWDATA = '1'  then
40                    Señal_RS <= RESET;
41                    Señal_RW <= RWDATA;
42                    DATA <= DATA_INSTRUCTIONS;
43                    estado <= RUN;
44                  else
45
46                  end if;
47
48              when RUN =>
49                  if(clk_count <= 30) then
50                  clk_count := clk_count + 1;
51                  Señal_EN <= '1';
52                  estado <= RUN;
53                  else
54                      Señal_EN <= '0';
55                      clk_count := 0;
56                      estado <= IDLE;
57                  end if;
58              when others => null;
59          end case;
60      end if;
61  end process;
62  end Behavioral;
```

En el código se puede observar que los cambios de estado se harán en cada pulso positivo de reloj. En el estado IDLE si RESET toma el valor de 1, todas las salidas se reinician al valor de 0; si el RESET tomar el valor de 1 y el RWDATA también, se habilita el leer y escribir, además de cambiar de estado a RUN. En este estado, se agregó la variable clk_count, para que se le de tiempo al LCD de realizar la acción dada, antes de volver a pasar al estado IDLE.

Este es el código del TestBench del controlador:

```
28  LIBRARY ieee;
29  USE ieee.std_logic_1164.ALL;
30  use ieee.numeric_bit.all;
31  use ieee.std_logic_textio.all;
32
33  Library std;
34  use std.textio.all;
35
36  -- Uncomment the following library declaration if using
37  -- arithmetic functions with Signed or Unsigned values
38  --USE ieee.numeric_std.ALL;
39
40  ENTITY TB IS
41  END TB;
42
43  ARCHITECTURE behavior OF TB IS
44
45      -- Component Declaration for the Unit Under Test (UUT)
46
47      COMPONENT LCD
48      PORT(
49           RESET : IN  bit;
50           CLK : IN  bit;
51           DATA_INSTRUCTIONS : IN  bit_vector(7 downto 0);
52           RS : IN  bit;
53           RWDATA : IN  bit;
54           Señal_RS : OUT  bit;
55           Señal_RW : OUT  bit;
56           Señal_EN : OUT  bit;
57           DATA : OUT  bit_vector(7 downto 0)
58          );
59      END COMPONENT;
60
```

```
    --Inputs
    signal RESET : bit := '0';
    signal CLK : bit := '0';
    signal DATA_INSTRUCTIONS : bit_vector(7 downto 0) := (others => '0');
    signal RS : bit := '0';
    signal RWDATA : bit := '0';

    --Outputs
    signal Señal_RS : bit;
    signal Señal_RW : bit;
    signal Señal_EN : bit;
    signal DATA : bit_vector(7 downto 0);


    -- Clock period definitions
    constant CLK_period : time := 10 ns;

BEGIN

    -- Instantiate the Unit Under Test (UUT)
    uut: LCD PORT MAP (
           RESET => RESET,
           CLK => CLK,
           DATA_INSTRUCTIONS => DATA_INSTRUCTIONS,
           RS => RS,
           RWDATA => RWDATA,
           Señal_RS => Señal_RS,
           Señal_RW => Señal_RW,
           Señal_EN => Señal_EN,
           DATA => DATA
         );
```

```
CLK_process :process
begin
   CLK <= '0';
   wait for CLK_period/10;
   CLK <= '1';
   wait for CLK_period/10;
end process;


-- Stimulus process
stim_proc: process

   file fin: TEXT open READ_MODE is "input.txt";

   variable current_read_line : line;
   variable current_read_field : bit_vector (10 downto 0);
begin

   while (not endfile(fin)) loop

      wait for 100 ns;
      readline(fin, current_read_line);
      read(current_read_line, current_read_field);

      RESET<= current_read_field(10);
      RS<= current_read_field(9);
      RWDATA<= current_read_field(8);
      DATA_INSTRUCTIONS<= current_read_field(7 downto 0);


   end loop;

   wait;
end process;
```

El siguiente el resultado de la simulación, como entrada se utilizó un archivo llamada “Input.txt” donde tenía escrito los siguientes valores:

01110111011

10111111111

00110000000

01111111100