

LABORATORIO DE SISTEMAS DIGITALEZ AVANZADOS.

HÉCTOR JAVIER PEQUEÑO CHAIREZ.

A01246364

PROF. EMILIO CABALLERO.

Reto SEGUNDO PARCIAL.

GITHUBLINKS:

INPUT TXT:

https://github.com/HectorPequeno06/HectorPequeno06/blob/master/Reto%20Segundo%20Parcial/input_test.txt

Modulo VHDL:

<https://github.com/HectorPequeno06/HectorPequeno06/blob/master/Reto%20Segundo%20Parcial/Lcd.vhdl>

Test Bench:

<https://github.com/HectorPequeno06/HectorPequeno06/blob/master/Reto%20Segundo%20Parcial/LCDCONt.vhdl>

OutPut TXT:

https://github.com/HectorPequeno06/HectorPequeno06/blob/master/Reto%20Segundo%20Parcial/output_test.txt

Controlador pantalla LCD.

El proyecto consta en generar una maquina de estados que funcione como controlador de la pantalla LCD. En este caso se utilizaron 5 estados en los cuales, los primeros 4 se ejecutaban por default simulando la inicialización de la pantalla LCD, el ultimo estado consistía en recibir la información para así generar un código script, el cual podíamos utilizar en un simulador para verificar que nuestro modulo funcionará correctamente.

El port del módulo contiene lo siguiente:

```
entity Controller is
  Port ( CLK : in  BIT;
         RESET : in  BIT;
         RS : in  BIT;
         RW : in  BIT;
         DI : in  BIT_VECTOR (7 downto 0);
         RSO : out  BIT;
         RWO : out  BIT;
         EN : out  BIT;
         DATA : out  BIT_VECTOR (7 downto 0));
end Controller;
```

Los estados se conformaban de las siguientes instrucciones:

```
process(State, RS ,RW,DI)
--Declaracion del archivo
file fout: TEXT open write_Mode is "output_test.txt";
Variable ctext: line; --Variable para ejecutar texto
begin
    EN<=CLK;
    case State is

        when 0=>

            RWO<='0';
            RSO<='0';
            DATA<= x"38";
            write(ctext,string'("instr(0x38);"));
            writeline(fout, ctext);

        when 1=>

            RWO<='0';
            RSO<='0';
            DATA<= x"0F";
            write(ctext,string'("instr(0x0F);"));
            writeline(fout, ctext);

        when 2=>

            RWO<='0';
            RSO<='0';
            DATA<= x"01";
            write(ctext,string'("instr(0x01);"));
            writeline(fout, ctext);

        when 3=>

            RWO<='0';
            RSO<='0';
            DATA<= x"06";
            write(ctext,string'("instr(0x06);"));
            writeline(fout, ctext);

        when 4=>

            RWO<=RW;
            RSO<=RS;
```

```

        DATA<=DI;
        if RS='0' then
            write(ctext,string("instr"));

        else
            write(ctext,string("data"));

        end if;
        write(ctext,to_integer(Unsigned(DI)));
        write(ctext,string'(");");
        writeline(fout, ctext);
    end Case;

end process;

```

Además de un Process, que funciona conforme al Clock, para el cambio de los estados de manera automatizada, ejecutando el inicio de la pantalla.

Comando de Lectura y salida de Archivos de Texto:

```

-- Stimulus process
stim_proc: process
    file fin : TEXT open Read_MODE is "input_test.txt";
    variable Ctext: line;
    variable RWR, RSR, RESR: BIT;
    variable DataR: bit_vector(7 downto 0);

    begin

        wait for 45 ns;

        while not endfile(fin) loop
            readline(fin, ctext);
            read(ctext, RSR);
            read(ctext, RWR);
            read(ctext, DataR);
            read(ctext, ResR);

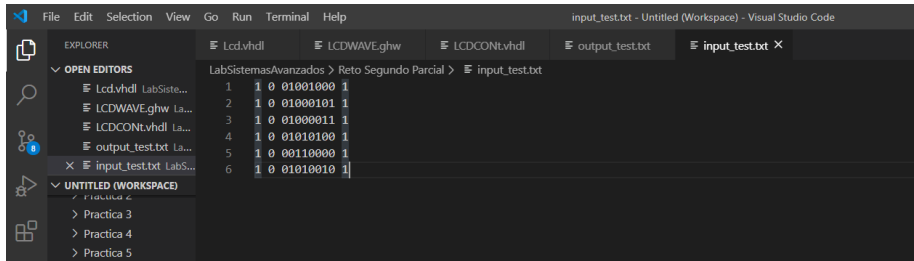
            Rs <= RSR;
            Rw <= Rwr;
            DI <= DataR;
            Reset <= Resr;
            wait for 0 ns;
        end loop;
    end process;

```

```
end loop;
```

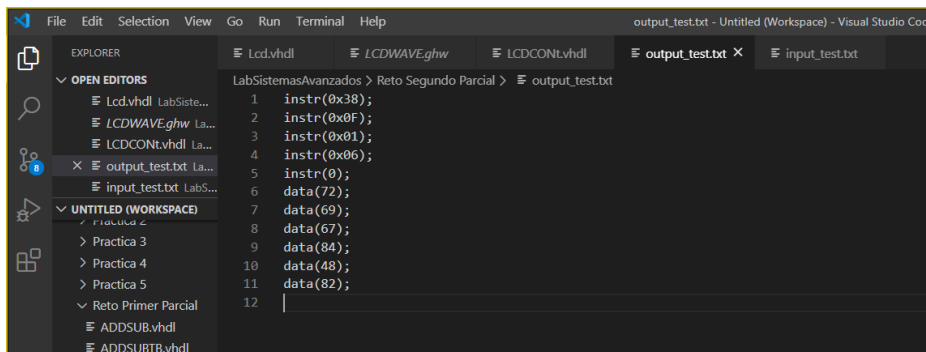
```
end process;
```

El Texto de Entrada Utilizado fue el siguiente:



```
1 1 0 01001000 1
2 1 0 01000101 1
3 1 0 01000011 1
4 1 0 01010100 1
5 1 0 00110000 1
6 1 0 01010010 1
```

El cual generó las siguientes salidas:



```
1 instr(0x38);
2 instr(0x0F);
3 instr(0x01);
4 instr(0x06);
5 instr(0);
6 data(72);
7 data(69);
8 data(67);
9 data(84);
10 data(48);
11 data(82);
12
```

Salidas Probadas en el simulador:

Una vez obtenidas las salidas, se ejecutó los comandos en el simulador.

