UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

LABORATORIO ORGANIZACIÓN Y ARQUITECTURA DE COMPUTADORAS

Práctica 2 - Diseño de máquinas de estado.

PROFESORA:

M.A. Ayesha Sagrario Roman García

Alumno	No. Cuenta	Grupo teoría
Becerril Olivar Axel Daniel	317113888	5
Cedillo Palacios Christian Javier	317097742	2
Rojas Mares Luis Iván	315299713	3

FECHA DE ENTREGA: 28/02/2025

ÍNDICE

Objetivo.....3

Introducción.....3

Desarrollo y resultados.....3

Resultados.....10

Conclusiones.....10

Referencias.....11

Objetivo

El objetivo de esta práctica fue familiarizarnos con los algoritmos de las máquinas de estado utilizando Quartus y el lenguaje de descripción de hardware VHDL. Se nos encomendó diseñar y simular una máquina de estados basada en una carta ASM, implementando un circuito secuencial con flip flops tipo D y un proyecto en VHDL para validar su correcto funcionamiento.

Introducción

En esta práctic veremos el diseño y simulación de máquinas de estado utilizando Quartus y el lenguaje VHDL, ya que nuestro objetivo principal fue comprender el funcionamiento de los algoritmos de control basados en máquinas de estado finito, implementando un circuito secuencial diseñado a partir de una carta ASM. Para ello, empleamos flip-flops tipo D como elementos de almacenamiento y desarrollamos una máquina en VHDL que permitió validar el comportamiento del sistema a través de simulaciones, lo cual nos ayudó a reforzar conocimientos sobre diseño digital y a familiarizarnos con herramientas clave en el desarrollo de hardware.

Desarrollo y resultados

Realizar los siguientes ejercicios.

Dada la carta ASM de la figura 1, elabore lo que se indica.

Obtenga el circuito secuencial de la carta ASM utilizando Flip Flops tipo D. Cree un proyecto en quartus llamado practica2 , implemente su diseño en el ambiente de desarrollo quartus y simularlo para validar su comportamiento.

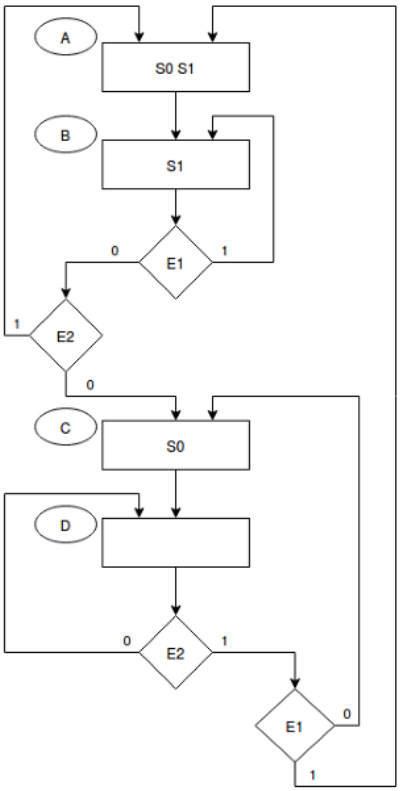


Figura 1: Carta ASM.

Ser realiza la tabla de de verdad de la carta ASM

Estado Presente		Entradas		Estado siguiente		Salidas	
Q1	**Q0**	**E2**	**E1**	**Q1'**	**Q0'**	**S1**	**S0**
0	0	0	0	0	1	1	1
0	0	0	1	0	1	1	1
0	0	1	0	0	1	1	1
0	0	1	1	0	1	1	1
0	1	0	0	1	0	1	0
0	1	0	1	0	1	1	0
0	1	1	0	0	0	1	0
0	1	1	1	0	1	1	0
1	0	0	0	1	1	0	1
1	0	0	1	1	1	0	1
1	0	1	0	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	0	1	1	0	0
1	1	0	1	1	1	0	0
1	1	1	0	1	0	0	0
1	1	1	1	0	0	0	0

Se aplican mapas de Karnaugh para obtener las ecuaciones

Mapas de Karnaugh

$S0 = \overline{Q0}$

E2E1 \ Q1Q0	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	1	0	0	1
10	1	0	0	1

$S1 = \overline{Q1}$

E2E1 \ Q1Q0	00	01	11	10
00	1	1	0	0
01	1	1	0	0
11	1	1	0	0
10	1	1	0	0

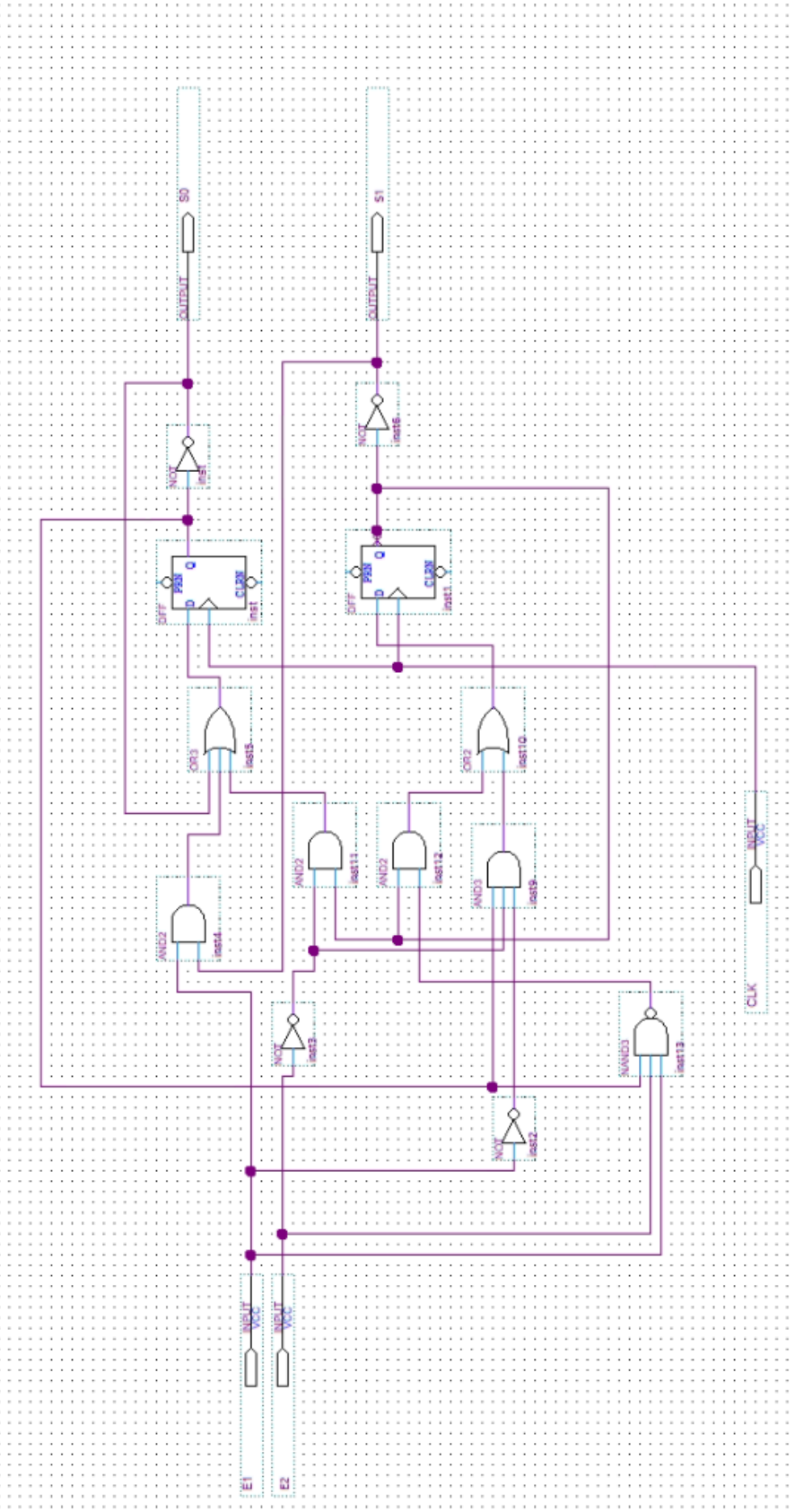
$Q0' = \overline{Q0} + E1\overline{Q1} + \overline{E2}Q1$

E2E1 \ Q1Q0	00	01	11	10
00	1	0	1	1
01	1	1	1	1
11	1	1	0	1
10	1	0	0	1

$Q1' = Q1(\overline{Q0E2E1}) + \overline{E2}\,\overline{E1}Q0$

E2E1 \ Q1Q0	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	1
10	0	0	1	1

Con las ecuaciones creamos el circuito secuencial en quartus de la siguiente manera


S0
OUTPUT
S1
OUTPUT
NOT
inst
NOT
inst6
DFF
PRE
D
CLRN
inst1
OR3
inst5
OR2
inst10
AND2
inst11
AND2
inst12
AND3
inst9
AND2
inst4
NOT
inst3
NOT
inst2
NAND3
inst13
INPUT
VCC
CLK
INPUT
VCC
E1
E2

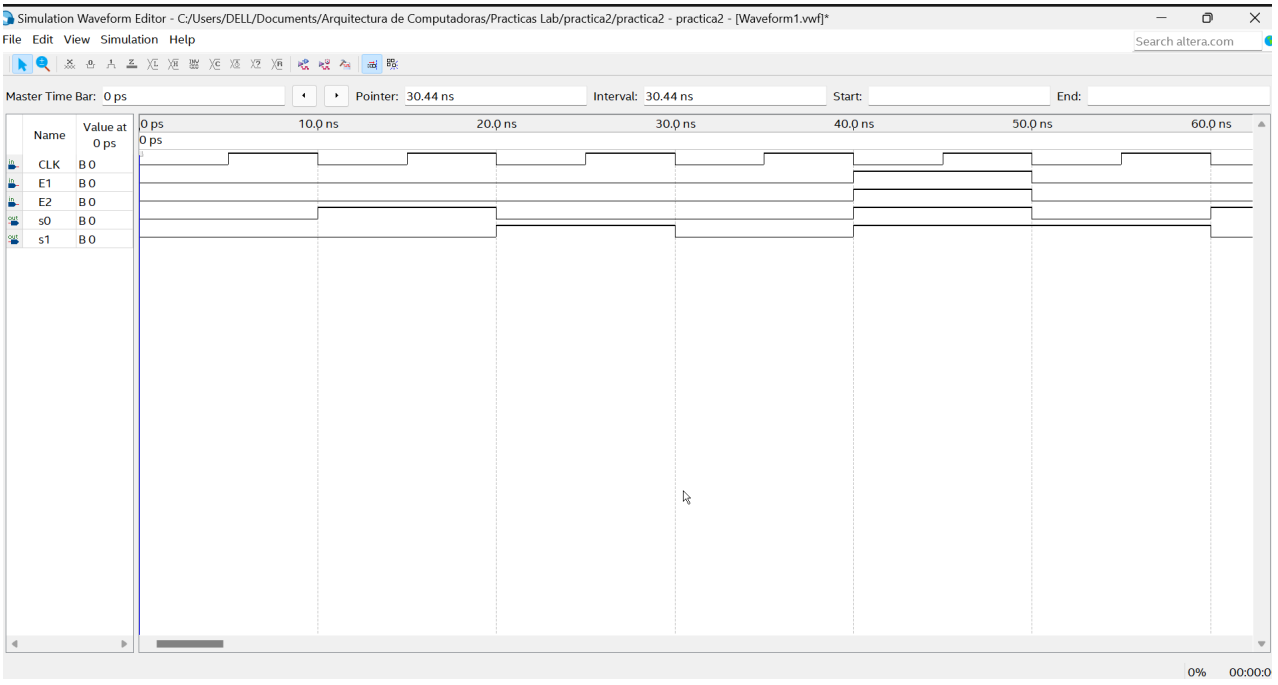
Para comprobar su funcionamiento analizamos el siguiente caso

ENTRADAS	CLK	S0	S1
E1=* E2=*	UN CICLO	1	1
E1=* E2*=	UN CICLO	0	1
E1= 0 E2=0	UN CICLO	1	0
E1=* E2=*	UN CICLO	0	0
E1=1 E2=1	UN CICLO	1	1

.

- Cuando se usa el * en las entradas se debe a que el estado no depende una decisión

Realizamos la simulación con wavefrom, colocando los datos de la tabla, las salidas deberian coincidir con estas, como se demostró en clase

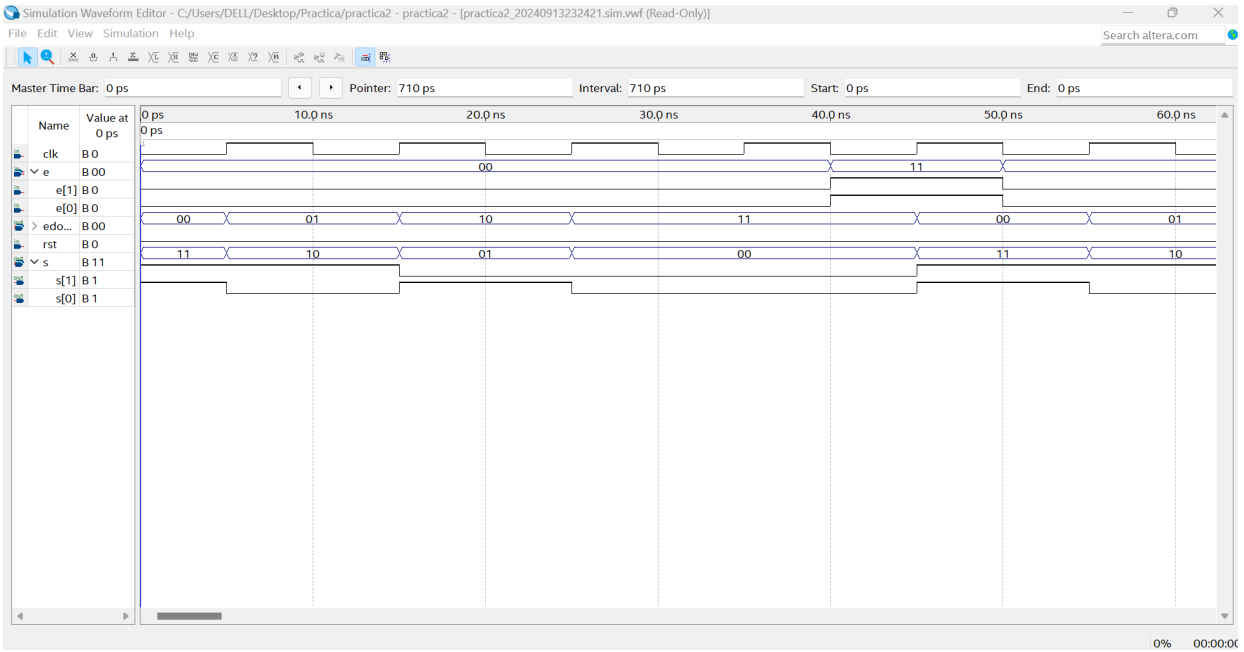


Cree un nuevo proyecto en Quartus llamado practica2 vhdl y diseñe la máquina de estados de la carta ASM utilizando el lenguaje de descripción de hardware VHDL. Simule su diseño para validar que funciona correctamente.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
use IEEE.numeric_std.all;

entity Practica2 is
   port(
      clk: in std_logic;
      rst: in std_logic;
      e: in std_logic_vector(1 downto 0);
        edoPresente :out std_logic_vector(1 downto 0);
      s: out std_logic_vector(1 downto 0)
   );
end Practica2;

architecture behavioral of Practica2 is
   type ESTADOS is (A, B, C, D);
   signal epresente, esiguiente: ESTADOS;

   begin
      process(clk, rst)
      begin
         if rst='1' then
               epresente <= A;
         elsif rising_edge(clk) then
            epresente <= esiguiente;
         end if;
      end process;

        process(epresente)
        begin
          case epresente is
             when A =>
                edoPresente <= "00";
             when B =>
                edoPresente <= "01";
             when C =>
                edoPresente <= "10";
             when D =>
                edoPresente <= "11";
             when others =>
                edoPresente <= "UU";
           end case;
        end process;
```

```
    process(epresente, e)
    begin
       case epresente is
          when A =>

             s<="11";
             esiguiente <= B;

          when B=>

             s<="10";

                 if e="00" then
              esiguiente <= C;
                 end if;
             if e="01" then
              esiguiente <= B;
                 end if;
                 if e="10" then
              esiguiente <= A;
                 end if;
                 if e="11" then
              esiguiente <= B;
                 end if;

          when C=>
             s<="01";
             esiguiente <= D;

          when D=>

                 s<="00";
                 if e="00" then
              esiguiente <= D;
                 end if;
             if e="01" then
              esiguiente <= D;
                 end if;
                 if e="10" then
              esiguiente <= C;
                 end if;
                 if e="11" then
              esiguiente <= A;
                 end if;

       end case;

    end process;
end behavioral
```

Realizamos la simulación con wavefrom, colocando los datos de la tabla, las salidas deberian coincidir con estas, como se demostró en clase



Resultados

Podemos observar que tanto el diseño del circuito como el diseño implementado en VHDL permitió obtener los resultados esperados, pudiendo observar los casos esperados en nuestra tabla de verdad.

Conclusiones

Becerril Olivar Axel Daniel: En un principio tuvimos problemas en la implementación del circuito secuencial ya que no nos mostraba el resultado esperado, sin embargo pudimos resolverlo y obtener los resultados esperados, además la práctica me permitió comprender mejor el diseño de máquinas de estado en hardware y reforzar lo aprendido anteriormente con el uso de VHDL.

Cedillo Palacios Christian Javier:

En conclusión, la implementación del circuito secuencial fue una complicación que, a través del análisis y la resolución de problemas, se convirtió en una oportunidad de aprendizaje. La corrección de los errores permitió no solo alcanzar el funcionamiento esperado, sino también

reforzar mi comprensión sobre el diseño de máquinas de estado en hardware y mejorar mis habilidades en VHDL.

Rojas Mares Luis Iván:

Al enfrentar la implementación del circuito secuencial, surgieron algunos inconvenientes que impedían obtener el resultado esperado. Afortunadamente, pudimos identificar y corregir el problema, logrando así el funcionamiento deseado. Esta experiencia me permitió afianzar mis conocimientos sobre el diseño de máquinas de estado en hardware y fortalecer mi manejo de VHDL.

Referencias

Organización y arquitectura de computadoras – BioRobotics. (s. f.). https://biorobotics.fi-p.unam.mx/organizacion-y-arquitectura-de-computadoras/

FPGA Support Resources | Altera. (s. f.). Intel. https://www.intel.com/content/www/us/en/support/programmable/support-resources/overview.html