Facultad de Ingeniería

Lab. Organización y Arquitectura de Computadoras

Practica No. 3
Construcción de Máquinas de estados Usando Memorias Direccionamiento por Trayectoria

Alumnos

- Monsalvo Bolaños Melissa Monserrat
- Romero Andrade Cristian

Grupo: 01

Profesor
Ing. Adrian Ulises Mercado Martinez

Semestre
2022–1

Fecha de Entrega
07 de octubre de 2021

Practica 3

Monsalvo Bolaños Melissa Monserrat y Romero Andrade Cristian

Índice

1. Introducción

Las máquinas de estados finitos nos sirven para realizar procesos bien definidos en un tiempo discreto. Reciben una entrada, hacen un proceso y nos entregan una salida. La descripción de estas máquinas de estados finitos, en lenguaje de alto nivel, se realiza de forma sencilla, pero en algunas ocasiones esta descripción consume muchos recursos del dispositivo, y además el tiempo de respuesta que dan los retardos de propagación es difícil de entender y controlar. De ahí la necesidad de aprender a describirlos de diferentes formas en busca del diseño más eficiente, por lo que es fundamental utilizar una arquitectura basada en el uso de un bloque ROM de la FPGA.

2. Objetivo

Familiarizar al alumno en el conocimiento de construcción de máquinas de estados usando direccionamiento de memorias con el método de direccionamiento por trayectoria.

3. Desarrollo

El direccionamiento por trayectoria guarda el estado siguiente y las salidas de cada estado de la carta ASM en una localidad de memoria. La porción de la memoria que indica el estado siguiente es llamada **LIGA**, mientras que la segunda porción de la memoria es usada para las **SALIDAS**.

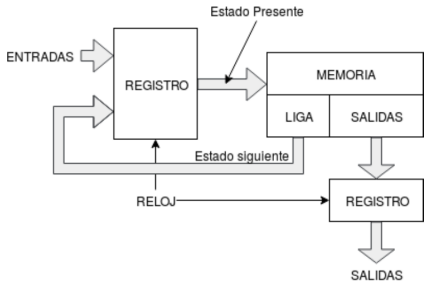

Figura 1: Direccionamiento por trayectoria

1. Dada la carta ASM de la figura 2, encuentre el contenido de memoria utilizando el direccionamiento por trayectoria. Recuerde que antes de construir la tabla se debe asignar a cada estado de la carta ASM una representación binaria. Así mismo, recuerde que para cada estado es necesario considerar todas las posibles combinaciones de las variables de entrada, aun cuando algunas de ellas no se utilicen en determinado estado

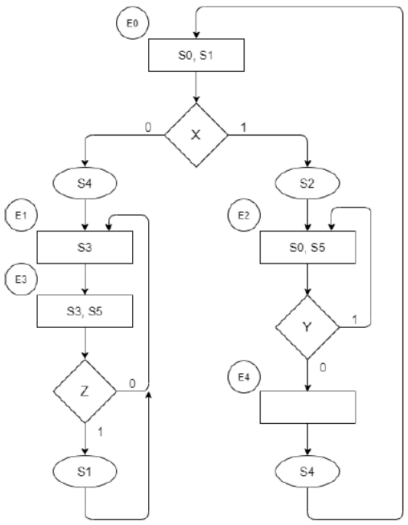

Figura 2: Carta ASM

Naturalmente asignaremos los siguientes valores binarios a la entradas:

- E0 = 000
- E1 = 001
- E2 = 010
- E3 = 011
- E4 = 100

Tabla 1: Contenido de la memoria

Dirección de memoria						Contenido de memoria								
E. Presente			Entradas			Liga			Salidas					
			X	y	z				s5	s4	s3	s2	s1	s0
0	0	0	0	0	0	0	0	1	0	1	0	0	1	1
0	0	0	0	0	1	0	0	1	0	1	0	0	1	1
0	0	0	0	1	0	0	0	1	0	1	0	0	1	1
0	0	0	0	1	1	0	0	1	0	1	0	0	1	1
0	0	0	1	0	0	0	1	0	0	0	0	1	1	1
0	0	0	1	0	1	0	1	0	0	0	0	1	1	1
0	0	0	1	1	0	0	1	0	0	0	0	1	1	1
0	0	0	1	1	1	0	1	0	0	0	0	1	1	1
0	0	1	0	0	0	0	1	1	0	0	1	0	0	0
0	0	1	0	0	1	0	1	1	0	0	1	0	0	0
0	0	1	0	1	0	0	1	1	0	0	1	0	0	0
0	0	1	0	1	1	0	1	1	0	0	1	0	0	0
0	0	1	1	0	0	0	1	1	0	0	1	0	0	0
0	0	1	1	0	1	0	1	1	0	0	1	0	0	0
0	0	1	1	1	0	0	1	1	0	0	1	0	0	0
0	0	1	1	1	1	0	1	1	0	0	1	0	0	0
0	1	0	0	0	0	1	0	0	1	0	0	0	0	1
0	1	0	0	0	1	1	0	0	1	0	0	0	0	1
0	1	0	0	1	0	0	1	0	1	0	0	0	0	1
0	1	0	0	1	1	0	1	0	1	0	0	0	0	1
0	1	0	1	0	0	1	0	0	1	0	0	0	0	1
0	1	0	1	0	1	1	0	0	1	0	0	0	0	1
0	1	0	1	1	0	0	1	0	1	0	0	0	0	1
0	1	0	1	1	1	0	1	0	1	0	0	0	0	1
0	1	1	0	0	0	0	0	1	1	0	1	0	0	0
0	1	1	0	0	1	0	0	1	1	0	1	0	1	0
0	1	1	0	1	0	0	0	1	1	0	1	0	0	0
0	1	1	0	1	1	0	0	1	1	0	1	0	1	0
0	1	1	1	0	0	0	0	1	1	0	1	0	0	0
0	1	1	1	0	1	0	0	1	1	0	1	0	1	0
0	1	1	1	1	0	0	0	1	1	0	1	0	0	0
0	1	1	1	1	1	0	0	1	1	0	1	0	1	0
1	0	0	0	0	0	0	0	0	0	1	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1	0	0	0	0
1	0	0	0	1	0	0	0	0	0	1	0	0	0	0
1	0	0	0	1	1	0	0	0	0	1	0	0	0	0
1	0	0	1	0	0	0	0	0	0	1	0	0	0	0
1	0	0	1	0	1	0	0	0	0	1	0	0	0	0
1	0	0	1	1	0	0	0	0	0	1	0	0	0	0
1	0	0	1	1	1	0	0	0	0	1	0	0	0	0

2. Una vez que haya obtenido el contenido de memoria, implemente el direccionamiento por trayectoria utilizando el software de desarrollo Quartus y escriba el contenido de memoria obtenido.

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity memoria_rom is
  generic(
    data_width : natural := 9;
    addr_length : natural := 6
    );
  port(
    clk: in std_logic;
    address: in std_logic_vector(addr_length-1 downto 0);
    data_out: out std_logic_vector(data_width-1 downto 0);
    monitor_entradas: out std_logic_vector(addr_length-1 downto 0));
end memoria_rom;
architecture behavioral of memoria_rom is
  constant mem_size: natural := 40;
  type mem_type is array (mem_size-1 downto 0) of std_logic_vector(data_width-1 downto 0);
  constant mem : mem_type :=
    (
      0=>b"001010011",1=>b"001010011",2=>b"001010011",3=>b"001010011",4=>b"010000111",
      5=>b"010000111",6=>b"010000111",7=>b"010000111",8=>b"011001000",9=>b"011001000",
      10=>b"011001000",11=>b"011001000",12=>b"011001000",13=>b"011001000",14=>b"011001000",
      15=>b"011001000",16=>b"100100001",17=>b"100100001",18=>b"010100001",19=>b"010100001",
      20=>b"100100001",21=>b"100100001",22=>b"010100001",23=>b"010100001",24=>b"001101000",
      25=>b"001101010",26=>b"001101000",27=>b"001101010",28=>b"001101000",29=>b"001101010",
      30=>b"001101000",31=>b"001101010",32=>b"000010000",33=>b"000010000",34=>b"000010000",
      35=>b"000010000",36=>b"000010000",37=>b"000010000",38=>b"000010000",39=>b"000010000",
      others=>b"111111111"
      );
begin
  rom : process(clk)
  begin
    if rising_edge(clk) then
      data_out <= mem(to_integer(unsigned(address)));
      monitor_entradas <= address;
    end if;
  end process rom;
end architecture behavioral;
```

Código 1: `memoria_rom.vhd`

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
entity registros_entrada is
  port (
    q : out std_logic_vector (5 downto 0);
    ent : in std_logic_vector (2 downto 0);
    liga : in std_logic_vector (2 downto 0);
    rst, clk: in std_logic
    );
end entity registros_entrada;
architecture arch_registros_entrada of registros_entrada is
begin
```

```
  identifier : process (clk)
  begin
    if (rising_edge(clk)) then
      if rst = '1' then
        q <= "000000";
      else
        q <= liga & ent;
      end if;
    end if;
  end process;
end architecture arch_registros_entrada;
```

Código 2: `registros_entrada.vhd`

Escribimos código que se usará para pasar los registros de un bloque a otro, para ello creamos el separador y la salida.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity separador is port (
  clk, rst : in std_logic;
  data_in : in std_logic_vector (8 downto 0);
  estado_siguiente : out std_logic_vector (2 downto 0);
  salidas : out std_logic_vector (5 downto 0)
  );
end separador;
architecture behavioral of separador is
  signal internal_value : std_logic_vector (8 downto 0) := b"000000000";
begin
  process (clk, rst, data_in)
  begin
    if rst = '1' then
      internal_value <= b"000000000";
    elsif rising_edge(clk) then
      internal_value <= data_in;
    end if;
  end process;
  process (internal_value)
  begin
    estado_siguiente <= internal_value(8 downto 6);
    salidas <= internal_value(5 downto 0);
  end process;
end behavioral;
```

Código 3: `separador.vhd`

Después de que verificamos todos los archivos, creamos los símbolos, de la barra de herramientas seleccionamos File→Create/Update→Create symbols for current file para cada archivo vhd.

Luego creamos un diagrama de bloques donde los conectamos y especificamos su salida (como se muestra en la figura 3).

3. Simule su diseño para probar su funcionamiento. Recuerden que en sus simulaciones debe aparecer el contenido de la memoria además del estado presente.

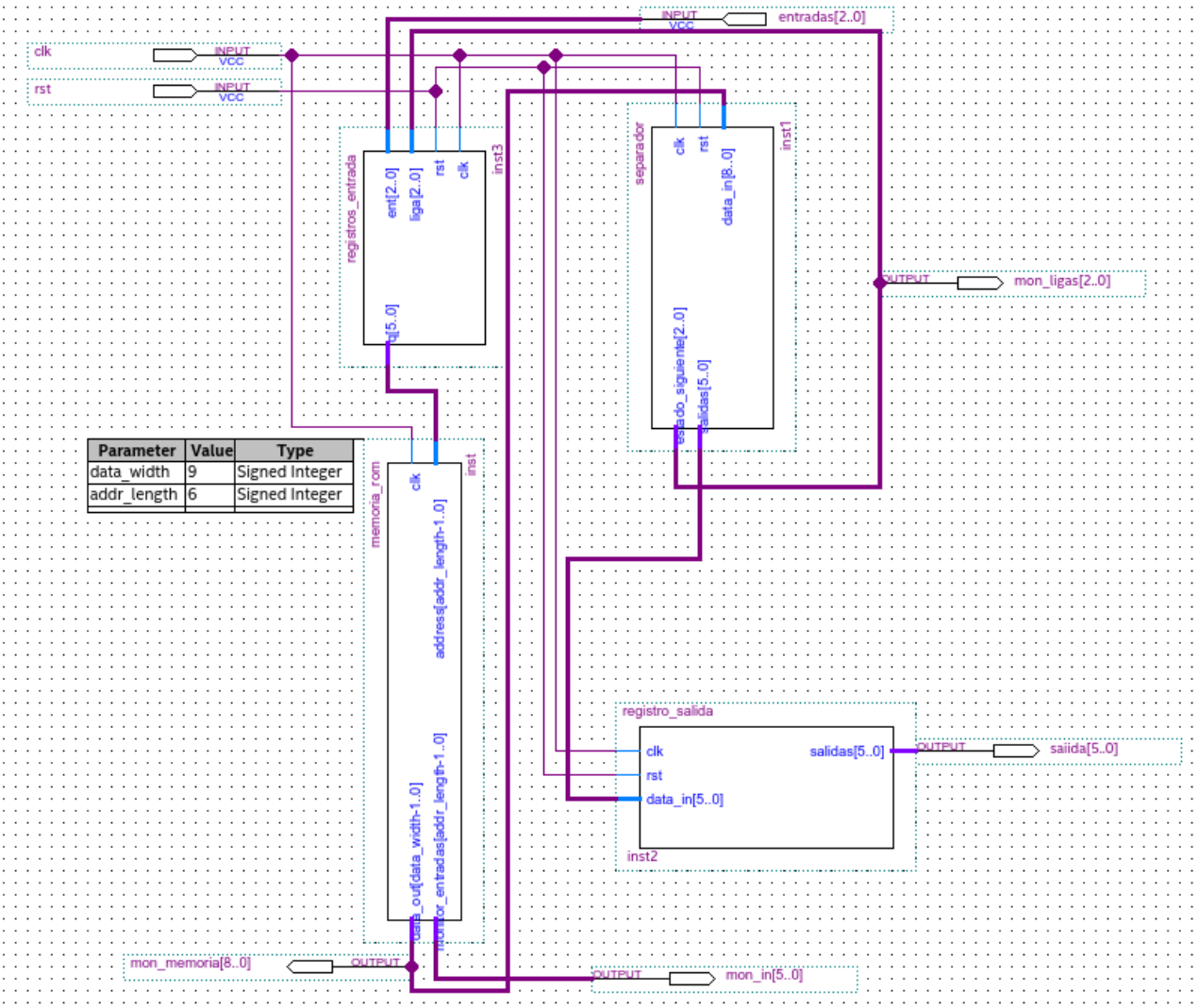

Figura 3: Diagrama de bloques de los archivos vhd

Para poder compilar el diagrama de bloques hay que seleccionar el archivo `*.bdf` que generamos como Top entity en las propiedades del proyecto. Adicionalmente se añadió unos monitores de memoria y de ligas en las salidas.

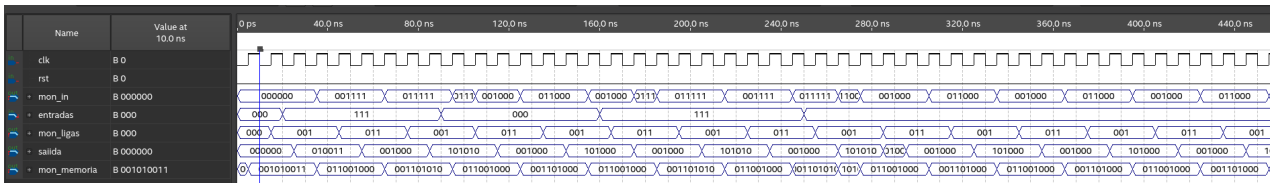

Figura 4: Simulación

4. Conclusiones

Monsalvo Bolaños Melissa Monserrat

Con el desarrollo de esta práctica pudimos describir una máquina ASM con el lenguaje VHDL siguiendo el método de direccionamiento por trayectoria y utilizando una memoria ROM y tres registros, uno de entrada, uno de salida y un separador de la salida que dividía la liga de las salidas. Realizamos bloques de cada una de las partes y a través de un diagrama esquemático realizamos las conexiones. Además pudimos comprobar su funcionamiento a través de la simulación.

Romero Andrade Cristian

El software de quartus nos facilita la implementación de nuestras tablas de verdad que generamos a partir de la carta ASM, si bien se puede hacer todo con código VHDL, el uso de los diagramas de bloques nos da "una pista" de los recursos que podremos utilizar a la hora de crear nuestro dispositivo.

Conclusiones en equipo

Con esta práctica pudimos implementar en el lenguaje VHDL una máquina ASM y para ello utilizamos el direccionamiento por trayectoria y con la ayuda de Quartus pudimos realizar un diagrama esquemático de sus componentes. Para obtener las salidas correspondientes, primero obtuvimos la tabla de verdad que consideraba las entradas X, Y y Z y la liga.

Referencias

Chavez, N. E. (s.f.). Construcción de máquinas de estados usando memorias. http://profesores.fi-b.unam.mx/normaelva

Savage, J. & Vázquez, G. (s.f.). *Diseño de microprocesadores*. Facultad de Ingenieria.