PRÁCTICA 3:

CONSTRUCCIÓN DE MÁQUINAS DE ESTADOS USANDO DIRECCIONAMIENTO POR TRAYECTORIA

INTEGRANTES:

BARRON PEREZ MARIAN ANDREA

PADILLA CASTILLO AARON SAMIR

GRUPO 3

Índice

Objetivo

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

Introducción

Se realizará la implementación de una carta ASM utilizando memoria por el método de trayectoria. Todo con el uso del software Quartus, el cual nos permite crear las memorias y registros a partir del lenguaje VHDL.

El direccionamiento por trayectoria es el más básico de los métodos de implementación en memoria aun siendo el que ocupa mas memoria, es el más sencillo de utilizar, por lo cual se analizan las entradas, estados presentes, las ligas y la salida. Este tipo de direccionamiento guarda el estado siguiente, también llamadas ligas y las salidas de cada estado de una carta ASM en un lugar de memoria. La memoria del estado siguiente contiene la liga del estado presente junto con las entradas.

Para cada estado es importante considerar todas las posibles combinaciones de las variables de entrada, aun si no se utilizan. También es importante asignarle a cada estado una representación binaria para poder construir la tabla de verdad.

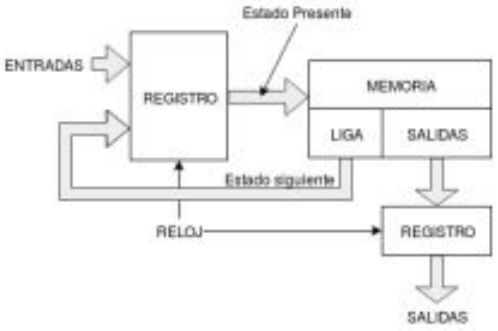

Figura 1: Diagrama de direccionamiento por trayectoria

Desarrollo

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.

El primer paso para comenzar a trabajar con el direccionamiento de trayectoria, es enumerar cada uno de los estados, por la implementación del mismo direccionamiento el orden que tenga no importa. A continuación se muestra nuestra propuesta:

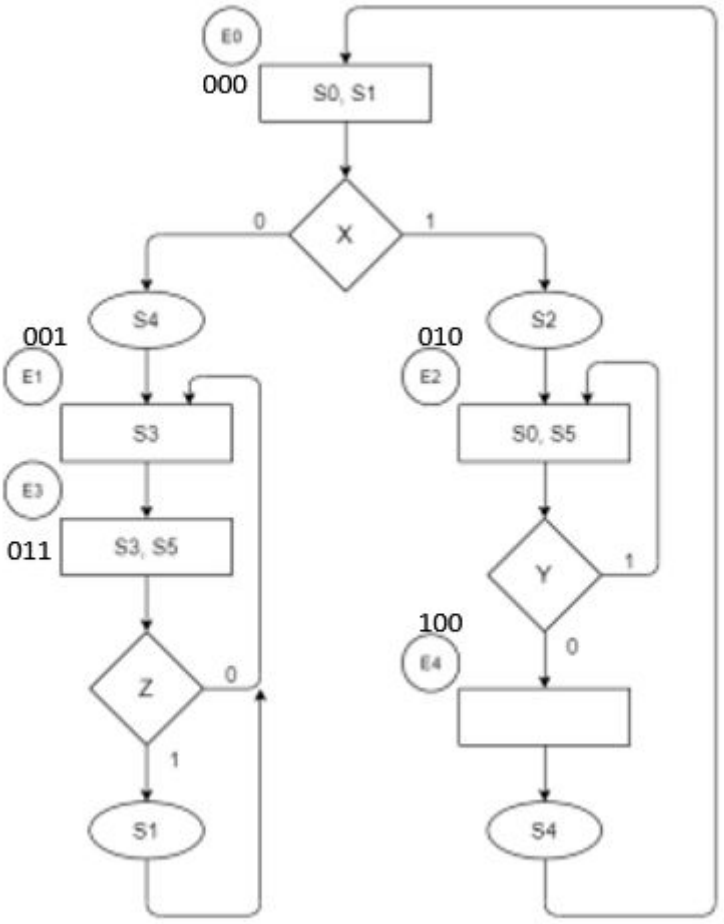

Figura 2: Carta ASM

La figura anterior ya se encuentra codificada, esto con el fin de poder realizar la tabla con el contenido de estados. Para poder resolver la carta ASM por trayectoria es necesario poder ayudarnos de la tabla en donde encontramos cada uno de los registros.

Memoria Teórica

	Direccion de memoria			Entradas			Contenido de la Memoria								
	Estado Presente			x	y	z	Liga			Salidas					
										s0	s1	s2	s3	s4	s5
E0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0
	0	0	0	0	0	1	0	0	1	1	1	0	0	1	0
	0	0	0	0	1	0	0	0	1	1	1	0	0	1	0
	0	0	0	0	1	1	0	0	1	1	1	0	0	1	0
	0	0	0	1	0	0	0	1	0	1	1	1	0	0	0
	0	0	0	1	0	1	0	1	0	1	1	1	0	0	0
	0	0	0	1	1	0	0	1	0	1	1	1	0	0	0
	0	0	0	1	1	1	0	1	0	1	1	1	0	0	0
E1	0	0	1	0	0	0	0	1	1	0	0	0	1	0	0
	0	0	1	0	0	1	0	1	1	0	0	0	1	0	0
	0	0	1	0	1	0	0	1	1	0	0	0	1	0	0
	0	0	1	0	1	1	0	1	1	0	0	0	1	0	0
	0	0	1	1	0	0	0	1	1	0	0	0	1	0	0
	0	0	1	1	0	1	0	1	1	0	0	0	1	0	0
	0	0	1	1	1	0	0	1	1	0	0	0	1	0	0
	0	0	1	1	1	1	0	1	1	0	0	0	1	0	0
E2	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
E3	0	1	1	0	0	0	0	0	1	0	0	0	1	0	1
	0	1	1	0	0	1	0	0	1	0	1	0	1	0	1
	0	1	1	0	1	0	0	0	1	0	0	0	1	0	1
	0	1	1	0	1	1	0	0	1	0	1	0	1	0	1
	0	1	1	1	0	0	0	0	1	0	0	0	1	0	1
	0	1	1	1	0	1	0	0	1	0	1	0	1	0	1
	0	1	1	1	1	0	0	0	1	0	0	0	1	0	1
	0	1	1	1	1	1	0	0	1	0	1	0	1	0	1

E4	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0
	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0
	1	0	0	0	1	1	0	0	0	0	0	0	0	1	0
	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0
	1	0	0	1	0	1	0	0	0	0	0	0	0	1	0
	1	0	0	1	1	0	0	0	0	0	0	0	0	1	0
	1	0	0	1	1	1	0	0	0	0	0	0	0	1	0
*	1	0	1	0	0	0	0	0	0	1	1	1	1	1	1
	1	0	1	0	0	1	0	0	0	1	1	1	1	1	1
	1	0	1	0	1	0	0	0	0	1	1	1	1	1	1
	1	0	1	0	1	1	0	0	0	1	1	1	1	1	1
	1	0	1	1	0	0	0	0	0	1	1	1	1	1	1
	1	0	1	1	0	1	0	0	0	1	1	1	1	1	1
	1	0	1	1	1	0	0	0	0	1	1	1	1	1	1
	1	0	1	1	1	1	0	0	0	1	1	1	1	1	1
*	1	1	0	0	0	0	0	0	0	1	1	1	1	1	1
	1	1	0	0	0	1	0	0	0	1	1	1	1	1	1
	1	1	0	0	1	0	0	0	0	1	1	1	1	1	1
	1	1	0	0	1	1	0	0	0	1	1	1	1	1	1
	1	1	0	1	0	0	0	0	0	1	1	1	1	1	1
	1	1	0	1	0	1	0	0	0	1	1	1	1	1	1
	1	1	0	1	1	0	0	0	0	1	1	1	1	1	1
	1	1	0	1	1	1	0	0	0	1	1	1	1	1	1
*	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1
	1	1	1	0	0	1	0	0	0	1	1	1	1	1	1
	1	1	1	0	1	0	0	0	0	1	1	1	1	1	1
	1	1	1	0	1	1	0	0	0	1	1	1	1	1	1
	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1
	1	1	1	1	0	1	0	0	0	1	1	1	1	1	1
	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1
	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1

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.

Implementación

Código ROM

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.ALL;

ENTITY practicaM IS
	PORT(
		addr : IN std_logic_vector(5 DOWNTO 0); --0:5 incluye el estado presente + entradas x,y,z
		--data : OUT std_logic_vector(8 DOWNTO 0); --6:14 incluye la liga y las salidas s0-s5
```

```
                salidas : OUT std_logic_vector(5 DOWNTO 0);
                liga : OUT std_logic_vector(2 DOWNTO 0)
        );
END practicaM;

ARCHITECTURE behavioral OF practicaM IS
        TYPE mem_rom IS ARRAY(0 TO 63) OF std_logic_vector(8 DOWNTO 0); --memoria/matriz/ROM 64x15
        SIGNAL data_out : mem_rom; --declara data_out del tipo memoria rom
BEGIN
        --llenamos matriz
        data_out(0) <= "001110010";
        data_out(1) <= "001110010";
        data_out(2) <= "001110010";
        data_out(3) <= "001110010";
        data_out(4) <= "010111000";
        data_out(5) <= "010111000";
        data_out(6) <= "010111000";
        data_out(7) <= "010111000";
        data_out(8) <= "011000100";
        data_out(9) <= "011000100";
        data_out(10) <= "011000100";
        data_out(11) <= "011000100";
        data_out(12) <= "011000100";
        data_out(13) <= "011000100";
        data_out(14) <= "011000100";
        data_out(15) <= "011000100";
        data_out(16) <= "100100001";
        data_out(17) <= "100100001";
        data_out(18) <= "010100001";
        data_out(19) <= "010100001";
        data_out(20) <= "100100001";
        data_out(21) <= "100100001";
        data_out(22) <= "010100001";
        data_out(23) <= "010100001";
        data_out(24) <= "001000101";
        data_out(25) <= "001010101";
        data_out(26) <= "001000101";
        data_out(27) <= "001010101";
        data_out(28) <= "001000101";
        data_out(29) <= "001010101";
        data_out(30) <= "001000101";
        data_out(31) <= "001010101";
        data_out(32) <= "000000010";
        data_out(33) <= "000000010";
        data_out(34) <= "000000010";
        data_out(35) <= "000000010";
        data_out(36) <= "000000010";
        data_out(37) <= "000000010";
        data_out(38) <= "000000010";
        data_out(39) <= "000000010";
        data_out(40) <= "000111111";
        data_out(41) <= "000111111";
        data_out(42) <= "000111111";
        data_out(43) <= "000111111";
        data_out(44) <= "000111111";
        data_out(45) <= "000111111";
        data_out(46) <= "000111111";
        data_out(47) <= "000111111";
        data_out(48) <= "000111111";
        data_out(49) <= "000111111";
```

```
        data_out(50) <= "000111111";
        data_out(51) <= "000111111";
        data_out(52) <= "000111111";
        data_out(53) <= "000111111";
        data_out(54) <= "000111111";
        data_out(55) <= "000111111";
        data_out(56) <= "000111111";
        data_out(57) <= "000111111";
        data_out(58) <= "000111111";
        data_out(59) <= "000111111";
        data_out(60) <= "000111111";
        data_out(61) <= "000111111";
        data_out(62) <= "000111111";
        data_out(63) <= "000111111";

        PROCESS(addr) BEGIN
                liga <= data_out(conv_integer(addr))(8 DOWNTO 6); --solo guardamos los ultimos 3 que son  liga
                salidas <= data_out(conv_integer(addr))(5 DOWNTO 0); --lo mismo pero solo guardamos los primeros 5
        END PROCESS;
END ARCHITECTURE behavioral;
```

Código Registro

```
library ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.ALL;

ENTITY registro IS
        PORT(
                clk : IN std_logic;
                reset : IN std_logic;
                entradas : IN std_logic_vector(2 DOWNTO 0);
                liga : IN std_logic_vector(2 DOWNTO 0);
                data_out: OUT std_logic_vector(5 DOWNTO 0) --estado presente
        );
END registro;

ARCHITECTURE behavioral OF registro IS
        SIGNAL internal_value : std_logic_vector(5 DOWNTO 0) := B"000000"; --primer direccion de la memoria en
ceros
BEGIN
        PROCESS (clk, reset, entradas, liga) BEGIN
                IF reset = '1' THEN
                        internal_value <= B"000000";
                ELSIF rising_edge(clk) THEN
                        internal_value <= entradas&liga; --entradas y liga a auxiliar
                END IF;
        END PROCESS;

        PROCESS(internal_value) BEGIN
                data_out <= internal_value; --auxiliar a estado presente
        END PROCESS;
END behavioral;
```

Circuito secuencial

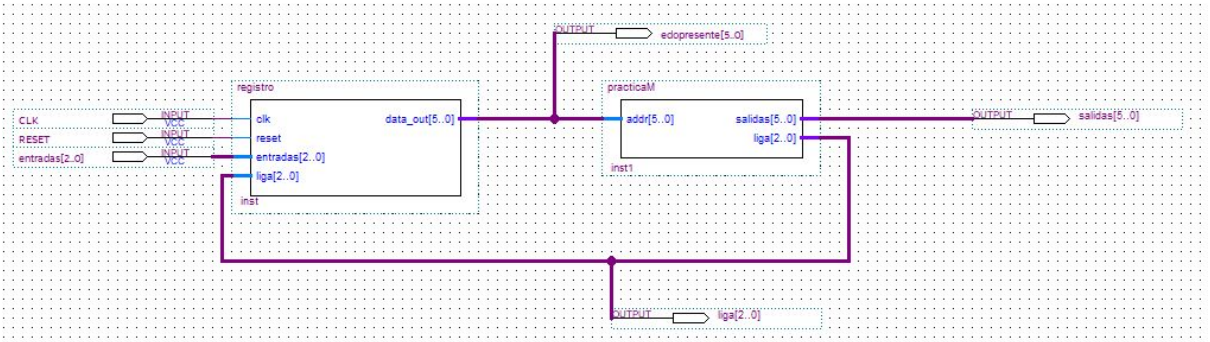

Figura 3: Circuito Secuencial

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

Simulación

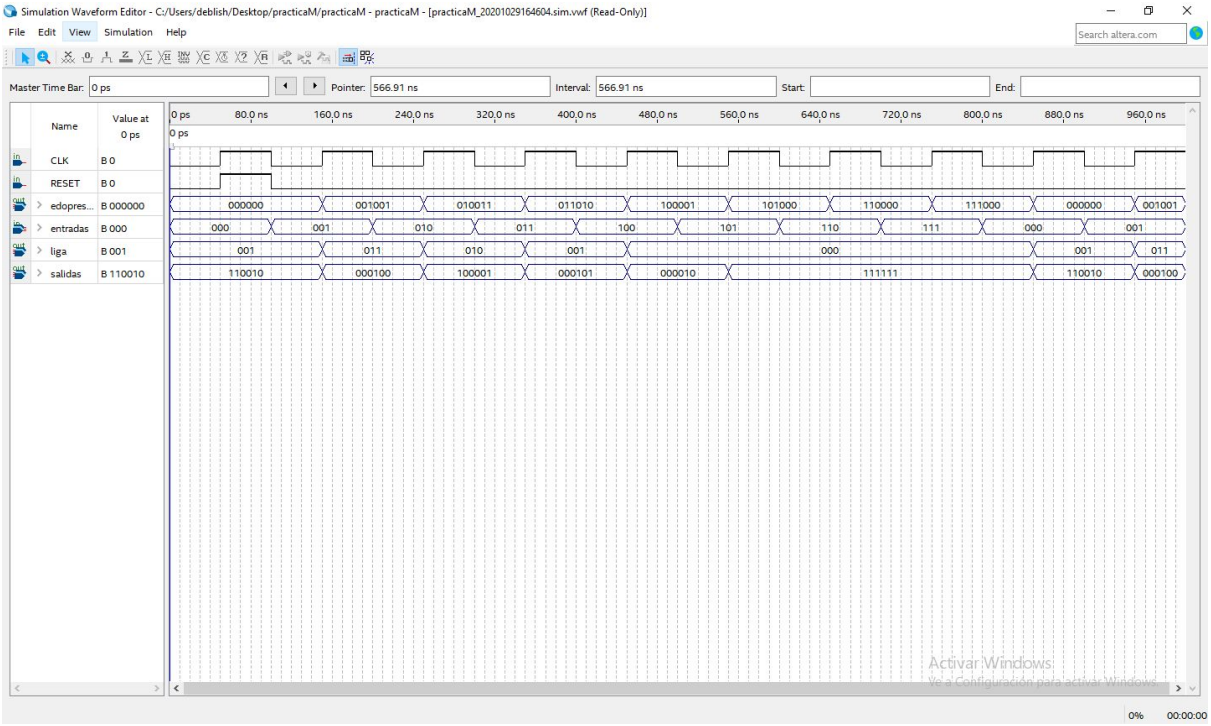

Figura 4: Captura de una simulación

Conclusiones

La realización de esta práctica nos permitió implementar una máquina de estados utilizando un método de direccionamiento en lugar de hacerlo con un circuito secuencial. Observamos que dicho método tiene ventajas importantes con respecto al anterior, aunque de igual manera tiene su complejidad y limitaciones. Con el direccionamiento por trayectoria se puede resolver casi cualquier carta ASM, aunque no es el más óptimo.

Barron Perez Marian Andrea

Esta práctica nos permitió conocer y comprender el comportamiento interno de una memoria al momento de direccionar. Si bien, el método que ocupamos es uno de los más sencillos, es el más ineficiente en cuanto a memoria, ya que por cada combinación que se tiene aumenta el tamaño. Conforme aumenta el número de entradas también aumenta el tamaño de la memoria, pero es una buena aproximación para comprender bien el concepto.

Padilla Castillo Aaron Samir

El direccionamiento por trayectoria es el direccionamiento más sencillo para implementar cartas ASM con memoria, pero tiene como desventaja un uso ineficiente de la misma ya que es obligatorio almacenar todos los casos posibles. La ventaja que tiene en términos de facilidad de aplicación es muy notable, sin embargo, aplicar este tipo de direccionamiento físicamente puede traer consigo unos costos elevados.

Participación en la práctica

Aportación teórica

Barron Perez Marian Andrea

Para la aportación teórica realice la introducción, desarrollo y conclusiones junto con mi compañero Samir, ya que realizamos una videollamada, cada uno dio su punto de vista y su opinión, de tal manera que en conjunto se realizó cada punto anterior, ya que de esta manera fue más sencillo de realizar la práctica.

Padilla Castillo Aaron Samir

Las únicas partes que se hicieron de manera individual fueron las conclusiones, esta nueva sección de ‘aportación’ y la portada, para el resto hicimos una videollamada, donde se discutió cómo avanzar en el desarrollo y actividades de esta práctica (código y simulaciones).

Aportación experimental

Barron Perez Marian Andrea

Para la aportación experimental de igual manera en cómo se realizó la aportación teórica, se realizó en equipo de tal manera que hicimos una videollamada, para que se tuviera clara la idea de como realizar la tabla del contenido de memoria y para la programación en vhdl, ya que de esta manera no habria ningun problema al entender el código.

Padilla Castillo Aaron Samir

Como se dijo anteriormente, el aporte fue bilateral y secuencial vía videollamada. Hemos intentado no hacerlo independientemente para que ambos podamos observar los cambios y avances en tiempo real, ha tomado más tiempo pero nos hemos sentido cómodos con la manera de trabajo.

Referencias

Savage, J. (2015). Diseño de microprocesadores. En *Diseño de microprocesadores* (2015.ª ed., pp. 50-52). Facultad de ingeniería. https://classroom.google.com/c/MTYwNjg5NzQ1Mjgz/m/MTY5OTM1NDA5OTk4/details?hl=es