**Práctica 4:** Construcción de máquinas de estado usando memoria con direccionamiento entrada-estado

Laboratorio de Organización y Arquitectura de Computadoras

**Grupo:** 2

**Alumnos:**

* Guzmán Sánchez José Emmanuel
* Mejía Ortiz Aarón Enrique
* Sáenz Barragán Ricardo

Objetivo

Familiarizar al alumno en el conocimiento de construcción de máquinas de estados usando direccionamiento de memorias con el método de direccionamiento entrada-estado.

**Dispositivo**: MAX10 DE-Lite 10M50DAF484C7G

1. Desarrollo

Para poder implementar el direccionamiento entrado-estado es necesario llenar nuestra memoria de acuerdo a nuestra carta ASM. Lo primero que hicimos fue definir el código de estados y el código de entradas de nuestro sistema, es decir, la representación binaria de cada uno dentro del mismo, el cual quedó de la siguiente manera (La variable vale 1 por defecto).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Estados** | **Código** |  |  |  |
| EST0 | 000 |  | **Entradas** | **Código** |
| EST1 | 001 |  | A | 00 |
| EST2 | 010 |  | B | 01 |
| EST3 | 011 |  | C | 10 |
| EST4 | 100 |  |  | 11 |

Figura 1. Representación binaria de estados y entradas

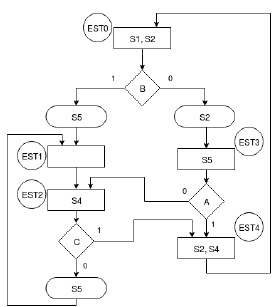
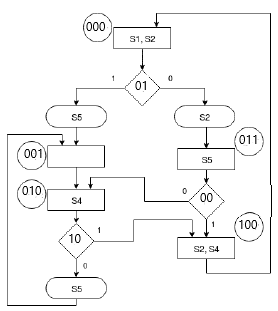
 

Figura 2. Carta ASM con la representación binaria de estados y entradas

Una vez que definimos la representación de nuestras entradas y estados, fue necesario llenar una tabla de transición, para poder llevar la interacción entre entradas y estados a un nivel más lógico. La tabla con esta información se muestra a continuación.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Presente** | | | **Prueba** | | **Liga F.** | | | **Liga V.** | | | **Salidas F.** | | | | | **Salidas V.** | | | | |
| P2 | P1 | P0 | K1 | K0 | L2 | L1 | L0 | L2 | L1 | L0 | S5 | S4 | S3 | S2 | S1 | S5 | S4 | S3 | S2 | S1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

Figura 3. Tabla de transición de estados

Una vez que llenamos la tabla, concatenado los bits de prueba, de liga y de salida, tanto en su rama falsa como verdadera, podemos obtener el contenido que tendrá la memoria en la dirección apuntada por los bits correspondientes al estado presente.

Con lo anterior, podemos definir el comportamiento que tendrá la memoria ROM en VHDL. Como se observa, primero obtenemos todo el contenido de la memoria de acuerdo al valor del estado presente. Con el contenido de la memoria, dependiendo del valor de prueba asignamos alguna de las entradas y posteriormente leemos el valor de dicha entrada para saber si a la salida mandamos la rama falsa o la rama verdadera.

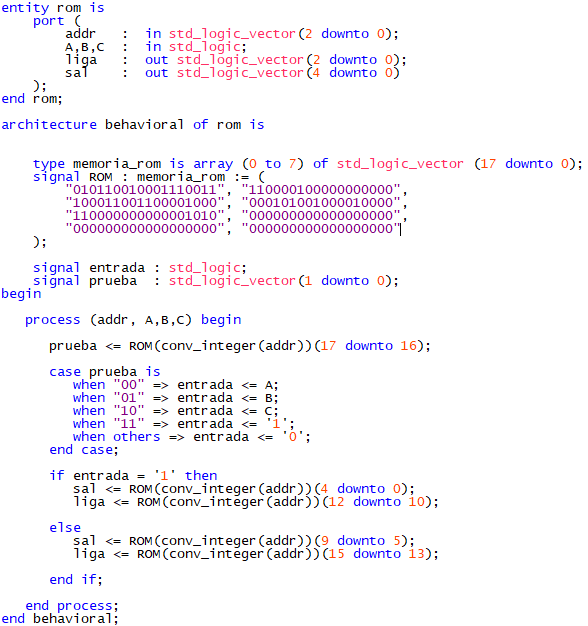


Figura 4. Código del comportamiento de la memoria

Después de recoger los valores de la memoria, definimos cómo se irá asignando el valor del estado siguiente, la liga, al estado presente, para de esta manera poder incluir el botón de reset.

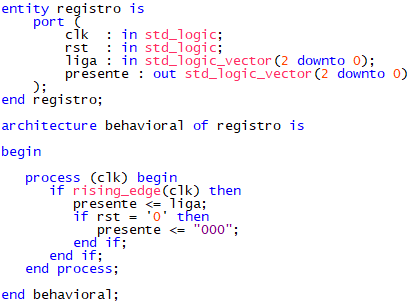


Figura 5. Código del registro

Con los bloques anteriores podemos formar todo nuestro circuito, agregando dos bloques extra. Un divisor de frecuencia por poder visualizar adecuadamente las salidas y un bloque que nos permita elegir entre ver el contenido de la memoria o ver el estado presente del sistema.

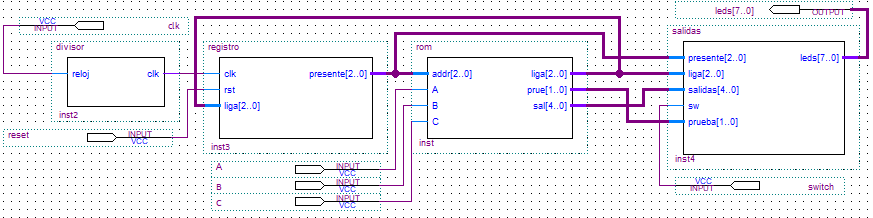
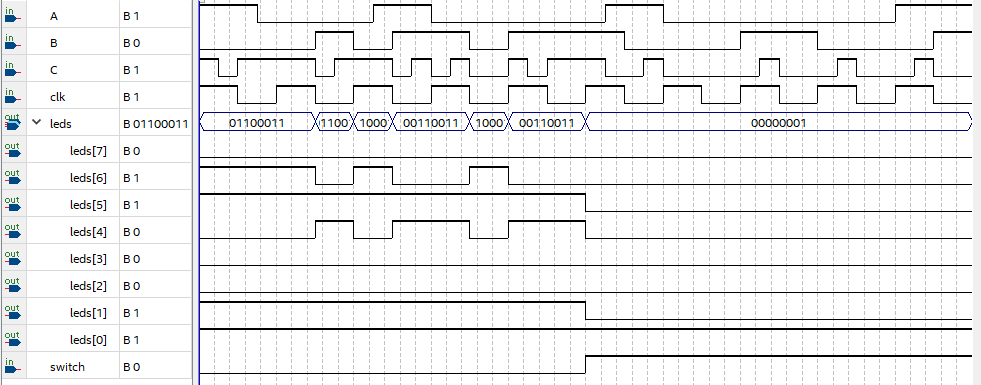


Figura 4. Diagrama de bloques del sistema

1. Simulaciones

En la simulación observamos que, como SW está abajo, en lo sleds se mostrarán del 7 al 5, la liga y del 4 al 0, las salidas. Debido a que B está apagado entonces en la liga se muestra “011” y como salidas solo se prenden los leds 1 y 0, correspondientes a las salidas S2 y S1.

Cuando SW se prende, se muestra el estado actual, en el ciclo anterior a que esto pase podemos observar que la liga vale “001”, por lo que cuando SW = ‘1’, el estado actual es “001”, por lo que en los leds del 2 al 0 se muestra este valor .



1. Ejecución del programa en FPGA









1. Conclusiones

Guzmán Sánchez José Emmanuel

El direccionamiento entrada-estado tiene una ventaja significativa sobre el de trayectoria debido a que se ahorra mucho espacio al multiplexar las entradas, esto hace en la tabla de transiciones no existan muchas combinaciones cuando se tien entradas condicionales, ya que sólo se toma el valor de salida, independientemente de la variable. Aunque si bien, es un buen método, aún no es posible manejar más de una entrada en un estado, pero es una buena aproximación a un buen modo de direccionamiento.

Mejia Ortiz Aarón Enrique

Este direccionamiento ayuda a reducir la cantidad de memoria que se utiliza en comparación al direccionamiento por trayectoria, aunque no es aplicable a todas las cartas ASM. Esto lo hace una gran opción para los sistemas que no dependen de dos entradas en un mismo estado. Aquí se muestra la utilidad de la multiplexación, pude ser una gran herramienta para simplificar circuitos y reducir la cantidad de recursos que se necesitan.

Sáenz Barragán Ricardo

Con ayuda de la práctica conseguimos reducir la cantidad de memoria usada a comparación del direccionamiento por trayectoria, también vimos que ahora no puede existir varias condiciones para moverse a otro estado, causando que este direccionamiento no sea valido para toda carta ASM. Podemos ver como existen el dilema de sacrificar memoria o funcionalidad para la creación de las maquinas de estado. Es un muy buen direccionamiento si las cartas ASM son simples y no se tiene tanta memoria disponible.