Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías

División de Electrónica y Computación
Departamento de Ciencias Computacionales
Licenciatura en Ingeniería en Computación
Arquitectura de computadoras
Clave: CC210 Sección: D02
19:00 – 20:55 Martes Jueves
Reporte Actividad 4.
Berrospe Barajas Héctor Eduardo
10-Marzo-2016
López Arce Delgado Jorge Ernesto

Introducción

Diseñaremos un decodificador binario de 2 a 4 con señal “enable”, se utilizara la sentencia if y la sentencia case en la práctica a cada una de las practicas se le anexara su respectivo testbench para su evaluación.

Objetivos

El alumno conocerá la estructura del decodificador, utilizando estructuras condicionales de lógica combinaciones, if.., case….

Se desea un decodificador 2-4, es decir con dos bits y señal de control ‘enable’ obtener un valor de salida de 4 bits con el cual se habilita alguna operación comportamental de mínimo 4 bits, como se observa en la Fig. 1.

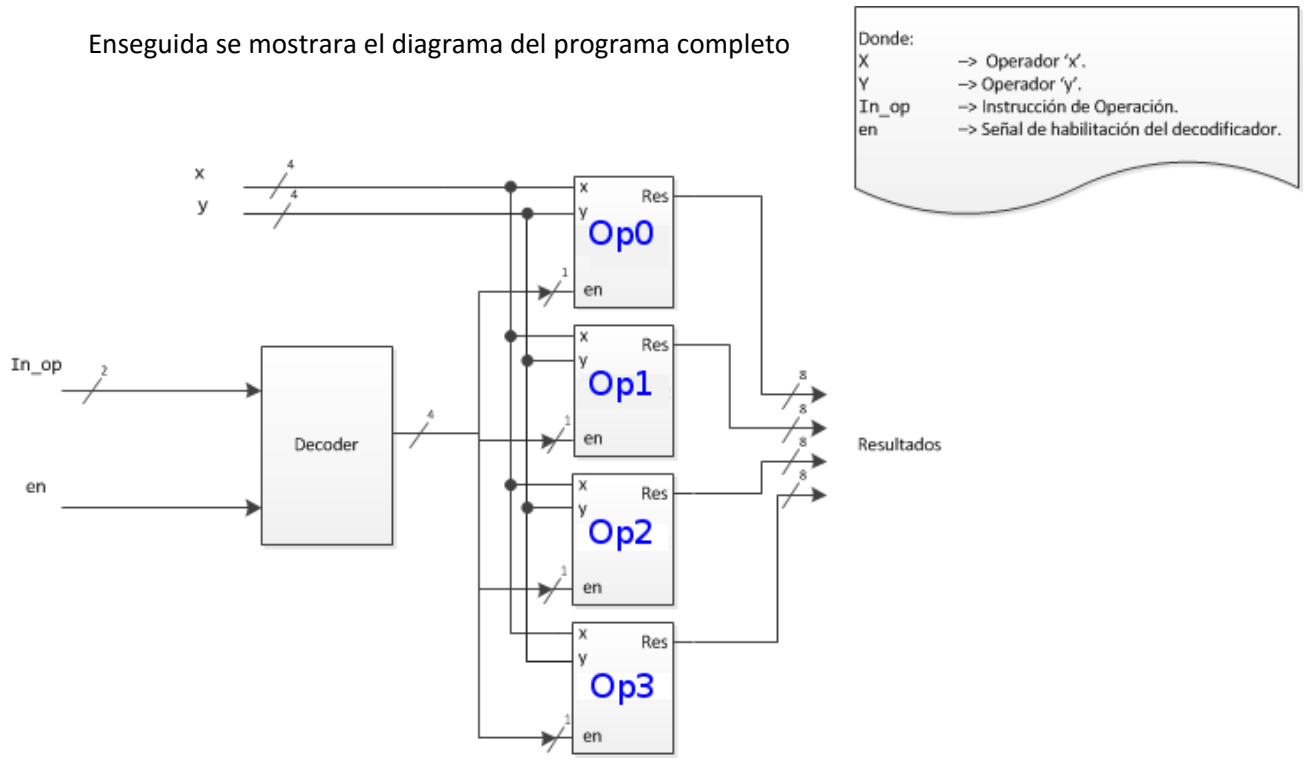
Desarrollo

Se generara un circuito el cual conste de 4 módulos de operaciones y 1 decodificador, la finalidad de esto es que dependiendo de la entrada del decodificador sea es la operación que se va realizar, la tabla de verdad del decodificador es la siguiente:

IN_OP	C
0 0	0001
0 1	0010
1 0	0100
1 1	1000

El bit en la posición del bit menos significativo se ira a la primera operación, el siguiente será la operación siguiente y así sucesivamente, cuando el bit sea 1, será la operación que se realizara.

Enseguida se mostrara el diagrama del programa completo



En esta practica se hicieron dos decodificadores, uno con la sentencia case y otro con la sentencia if.

If (condition)

statement;

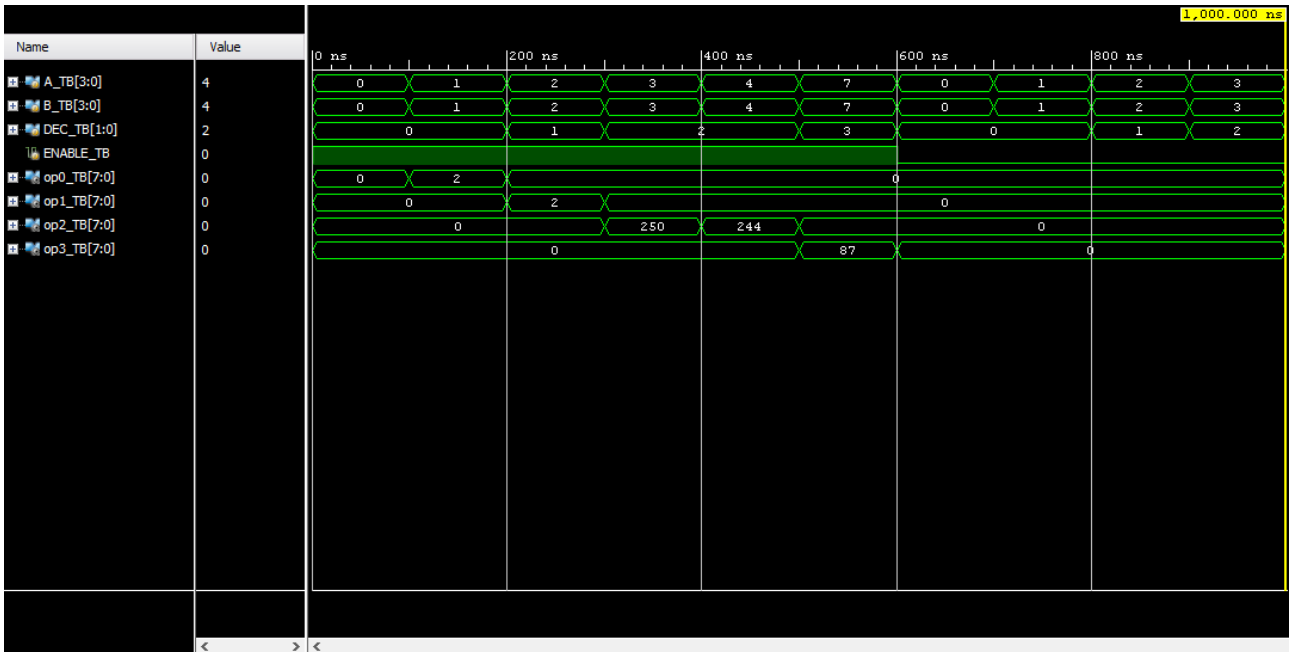
Else

statement;

---

case (expression)
expression : statement
expression {, expression} : statement
default : statement

En seguida se muestra la implementación con testbench de el decodificador



Aqui el codigo del testbench

```
ENABLE_TB = 1'b1;      //ENABLE = 1

DEC_TB = 2'b00;        //Ejecuta la operación 1

   A_TB = 4'b0000;
      B_TB = 4'b0000;
      #100;

   A_TB = 4'b0001;
      B_TB = 4'b0001;
      #100;

DEC_TB = 2'b01;        //Ejecuta la operación 2

   A_TB = 4'b0010;
      B_TB = 4'b0010;
      #100;

DEC_TB = 2'b10;        //Ejecuta la operación 3

   A_TB = 4'b0011;
      B_TB = 4'b0011;
      #100;

   A_TB = 4'b0100;
      B_TB = 4'b0100;
      #100;

DEC_TB = 2'b11;        //Ejecuta la operación 4

   A_TB = 4'b0111;
      B_TB = 4'b0111;
      #100;

ENABLE_TB = 1'b0;         //ENABLE = 0, no permite realizar ninguna operación

DEC_TB = 2'b00;

    A_TB = 4'b0000;
      B_TB = 4'b0000;
      #100;

    A_TB = 4'b0001;
      B_TB = 4'b0001;
      #100;
```

```
DEC_TB = 2'b01;

   A_TB = 4'b0010;
     B_TB = 4'b0010;
     #100;

DEC_TB = 2'b10;

   A_TB = 4'b0011;
     B_TB = 4'b0011;
     #100;

   A_TB = 4'b0100;
     B_TB = 4'b0100;
     #100;

DEC_TB = 2'b11;

   A_TB = 4'b0111;
     B_TB = 4'b0111;
     #100;

  $stop;
  end
```

Conclusiones

Se aprendió a utilizar la sentencia if y else en verilog, es muy similar a la sintaxis de cualquier otro lenguaje de programación, una de las dificultades que se me presento fue en el case, pues tenía un problema en el cual quería indicar que si en caso de que ENABLE fuera igual a 1 no podía declarar el caso en el que mi segunda expresión, o sea la entrada del decodificador sea igual a 00 me arrojara un resultado, la resolví poniendo un case que me mande a otro case.