Práctica 1. Introducción a sistemas Digitales en Verilog.

1.1.1 Introducción.

Los circuitos lógicos para sistemas digitales pueden ser combinacionales o secuenciales. Un circuito combinacional consiste en compuertas lógicas cuyas salidas en cualquier momento están determinadas por la combinación actual de entradas. Un circuito combinacional realiza una operación que se puede especificar lógicamente con un conjunto de funciones booleanas.
Los circuitos secuenciales usan elementos de almacenamiento además de compuertas lógicas, y sus salidas son función de las entradas y del estado de los elementos de almacenamiento.
Esto último, a su vez, es función de entradas anteriores. Por ello, las salidas de un circuito secuencial dependen no sólo de los valores actuales de las entradas, sino también de entradas anteriores, y el comportamiento del circuito se debe especificar con una sucesión temporal de entradas y estados internos.

1.1.2 Acerca del lenguaje.

Verilog es un lenguaje de descripción hardware (Hardware Description Language, HDL) utilizado para describir sistemas digitales, tales como procesadores, memorias o un simple flip-flop. Esto significa que realmente un lenguaje de descripción hardware puede utilizarse para describir cualquier hardware (digital) a cualquier nivel.

Existen dos tipos de datos principales en Verilog:

- **Nets**: Representan conexiones estructurales entre componentes. No tienen capacidad de almacenamiento de información.
- **Registers**: Representan variables con capacidad de almacenar información.

Y son declarados de la siguiente forma:

- **Inputs:** El tipo de las señales de entrada NO SE DEFINEN, por defecto se toman como **wire**.
- **Outputs:** Las salidas pueden ser de tipo wire o reg, dependiendo si tiene capacidad de almacenamiento de información, En Verilog, un nodo tipo wire puede conectarse a una salida.
- **Nodos internos:** Igual que las salidas.

1.2. Previo Práctica 1.

1) Defina con sus palabras los niveles de abstracción en Verilog.

-Nivel de puerta:

-Nivel de transferencia de registro:

-Nivel de comportamiento:

2) Describa los siguientes tipos de procesos.

-Initial

-Always

3) Defina y escriba la sintaxis para las asignaciones continuas en Verilog.

1.3. Ejemplo Práctico.

- Procedimiento de análisis

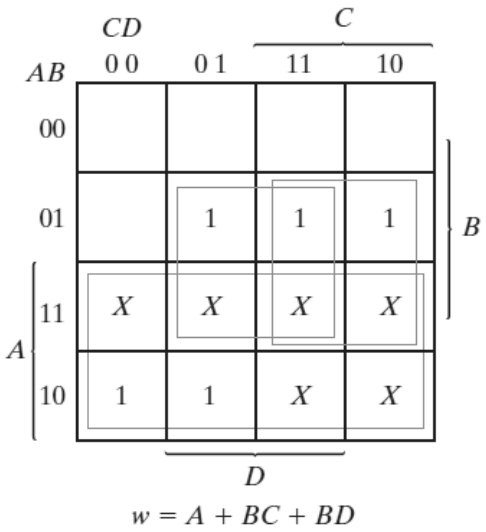
El análisis de un circuito combinacional requiere deducir la función que realiza el circuito. Este proceso parte de un diagrama lógico dado y culmina en un conjunto de funciones booleanas, una tabla de verdad o una posible explicación del funcionamiento del circuito. Si el diagrama lógico a analizar va acompañado de un nombre de función o de una explicación de lo que se supone que hace, el problema de análisis se reducirá a una verificación de la función planteada. El análisis se efectúa manualmente encontrando las funciones booleanas o la tabla de verdad, o bien, utilizando un programa de simulación en computadora.
El primer paso del análisis consiste en asegurarse de que el circuito dado sea combinacional y no secuencial. El diagrama de un circuito combinacional tiene compuertas lógicas sin trayectorias de retroalimentación ni elementos de memoria. Una trayectoria de retroalimentación es una conexión de la salida de una compuerta a la entrada de una segunda compuerta que forma parte de la entrada a la primera compuerta.

- Método de obtención por mapas de Karnaugh

Tabla de verdad para el ejemplo de conversión de código

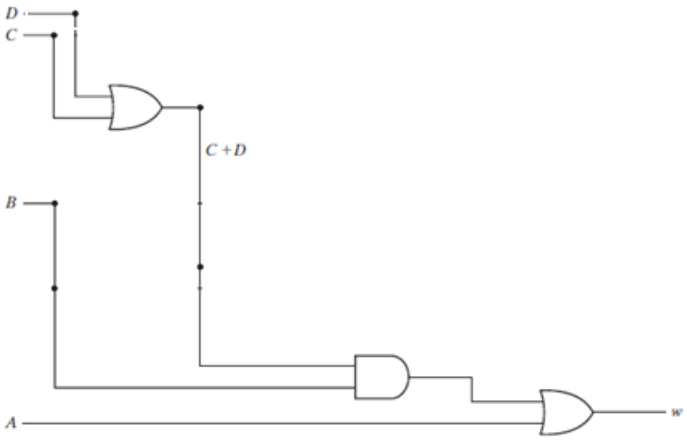
Entrada BCD				Salida código exceso-3			
A	*B*	*C*	*D*	*w*	*x*	*y*	*z*
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

Para **w:**



Pero $A + BC + BD = A + B(C + D)$

Entonces:



Implementación en verilog.

A continuación, se describirá "w" a nivel compuerta.

```
module myAND (
   input a,
   input b,
   output z
);
   assign z = a & b;

   endmodule
```

```
module myOR (
   input a,
   input b,
   output z
);
   assign z = a | b;

endmodule
```

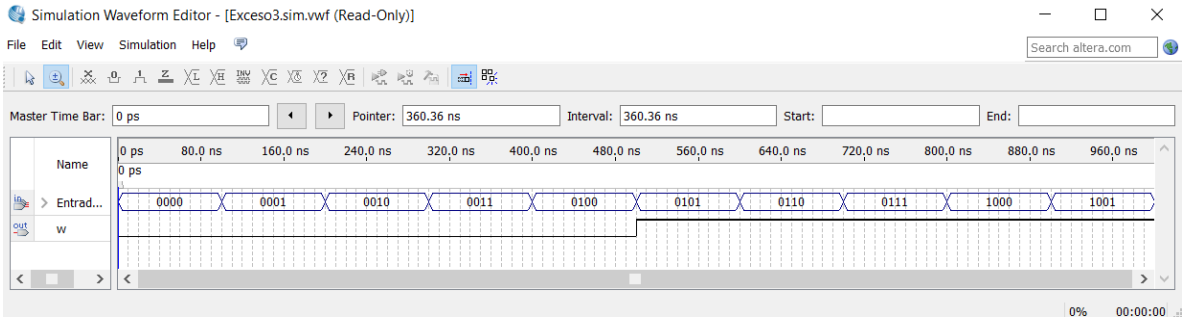
```
module Exceso3 (
   input systemA,
   input systemB,
   input systemC,
   input systemD,
   output w
);
   wire ORtoAND;
   wire ANDtoOR;

   myOR OR1 (
      .a(systemD),
      .b(systemC),
      .z(ORtoAND)
   );

   myAND AND1 (
      .a(ORtoAND),
      .b(systemB),
      .z(ANDtoOR)
   );

   myOR OR2 (
      .a(ANDtoOR),
      .b(systemA),
      .z(w)
   );
endmodule
```

- Simulación Funcional para w



1.4. Trabajo de laboratorio

Obtener las ecuaciones booleanas para x, y, z mediante el método de mapas de **Karnaugh,** realizar su instanciamiento en verilog y su simulación funcional, para posteriormente implementarlo en su FPGA y ser presentado a su instructor.

***Nota: La implementación del circuito anterior se deberá realizar en un solo proyecto, se recomienda trabajar en grupo para obtener las 3 ecuaciones restantes lo más rápido posible.**

1.5. Conclusiones