EIE
Escuela de
Ingeniería Eléctrica

UNIVERSIDAD DE COSTA RICA

Universidad de Costa Rica

Escuela de Ingeniería Eléctrica

Proyecto Final
(Parte Individual)

Circuitos Digitales I

Estudiante:
Royer Méndez Ramírez - A43333

Profesor:
Rafael Esteban Badilla Alvarado

Circuitos Digitales I
IE-0323

San José,
1 de marzo de 2022

1. Investigue qué es un sumador completo de 1 bit. Adjunte su tabla de verdad bajo los siguientes nombres de variables:

i. Primer entrada: inA

ii. Segunda entrada: inB

iii. Acarreo de entrada: carryIn

iv. Resultado de la suma: sum

v. Acarreo de salida: carryOut

Un sumador de un bit es un modelo que representa el resultado de sumar cierta cantidad de bits. Mediante el modelo se expresan las posibles combinaciones de los elementos binarios 1 y 0 y sus resultados a la hora de ser sumados; el mismo modelo explica coherentemente los casos en los que no ocurren los llamados 'acarreos', cuando ocurren acarreos salientes y cuando ocurren acarreos entrantes.

Se tiene entonces que los elementos a que van a ser sumados se pueden sumar en paralelo o verticalmente. Todos los elementos a sumar en esa misma columna o fila (dependiendo de la distribución elegida) tendrán el mismo valor posicional (bit mas significativo o bit menos significativo, entre otros).

En la Figura 1 se muestra un pequeño ejemplo de la suma de un bit, dentro del óvalo se están sumando A + B = 1 + 1 = 10, pero a esa columna está entrando un bit de llevo que resultó de la suma de la columna anterior por lo que la suma entonces sera A + B + carryIn = 1 + 1 + 1 = 11, cifra que se separa en sum = 1 y carryOut = 1; en otras palabras, hay un bit de llevo para la siguiente columna.

Figura 1: Ejemplo de la suma de 1 Bit

carryIn		1	1		1	1	
A	1	0	1	1	0	1	1
B	0	0	1	1	0	0	1
Suma	**1**	**1**	**1**	**0**	**1**	**0**	**0**
carryOut			1	1		1	1

Elaboración personal

Un sumador completo de 1 Bit se rige mediante la tabla de verdad representada en la Tabla 1

Cuadro 1: Tabla de verdad del contador de 1 Bit

Entradas			**Salidas**	
carryIn	**inA**	**inB**	**sum**	**carryOut**
0	0	0	0	0
1	0	0	1	0
0	0	1	1	0
1	0	1	0	1
0	1	0	1	0
1	1	0	0	1
0	1	1	0	1
1	1	1	1	1

Basado en Tabla 1 y con los nombres de sus entradas y salidas arriba indicadas (Tabla 1) es posible elaborar las compuertas necesarias para su funcionamiento mediante expresiones booleanas:

De la misma tabla se hace la sumatoria de minterminos para las salidas *sum* (Ecuación 1) y *carryOut* (Ecuación 2)

$$\boldsymbol{sum} = carryIn \cdot \overline{inA} \cdot \overline{inB} + \overline{carryIn} \cdot \overline{inA} \cdot inB + \overline{carryIn} \cdot inA \cdot \overline{inB} + carryIn \cdot inA \cdot inB \quad (1)$$

$$\boldsymbol{carryOut} = carryIn \cdot \overline{inA} \cdot inB + carryIn \cdot inA \cdot \overline{inB} + \overline{carryIn} \cdot inA \cdot inB + carryIn \cdot inA \cdot inB \quad (2)$$

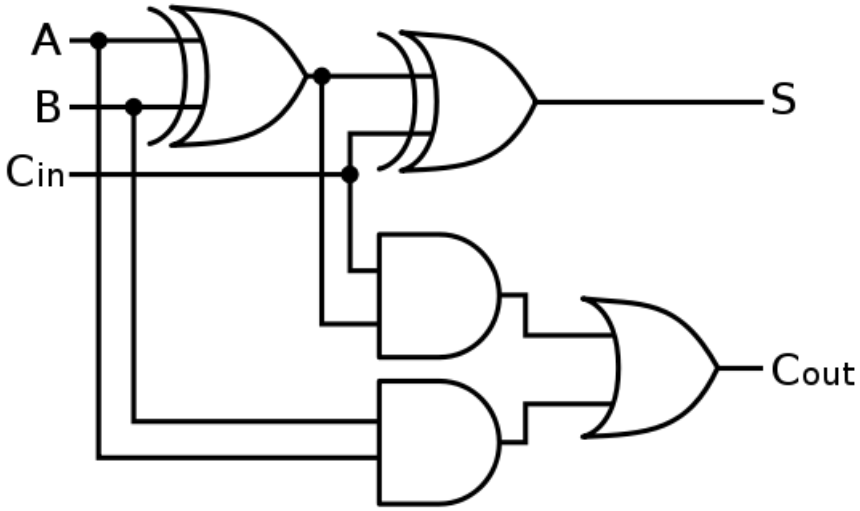
Las expresiones anteriores se pueden simplificar aplicándoles las proposiciones elementales y las leyes fundamentales de álgebra de conmutación para llegar a sus ecuaciones simplificadas equivalentes (Ecuación 3 y Ecuación 4)

$$\boldsymbol{sum} = inA \oplus inB \oplus carryIn \quad (3)$$

$$\boldsymbol{carryOut} = inA \cdot inB + carryIn(inA \oplus inB) \quad (4)$$

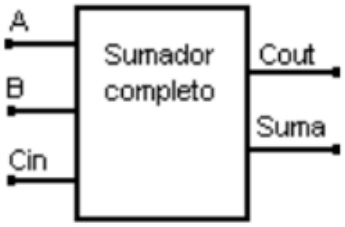
Las ecuaciones anteriores se pueden representar por medio de conectivas (Figura 2) y en su forma de diagrama de bloques (Figura 3)

Figura 2: Esquemático del sumador completo de 1 bit



Tomado de internet

Figura 3: Sumador completo de 1 Bit en la forma de diagrama de bloques



Tomado de internet

Con toda la información brindada es posible crear un modulo que interprete un sumador de 1 bit en el programa de diseño de circuitos electrónicos *Verilog*.

El código para el módulo *fullAdder.v* creado se muestra en la Figura 4

Figura 4: Módulo del fullAdder.v

```
1  module fullAdder(
2      input inA,
3      input inB,
4      input carryIn,
5      output sum,
6      output carryOut);
7
8  assign carryOut =
9      (~inA & inB & carryIn) |
10     (inA & ~inB & carryIn) |
11     (inA & inB & ~carryIn) |
12     (inA & inB & carryIn);
13
14 assign sum =
15     (~inA & ~inB & carryIn) |
16     (~inA & inB & ~carryIn) |
17     (inA & ~inB & ~carryIn) |
18     (inA & inB & carryIn);
19
20 endmodule
```

Elaboración: Esteban Badilla / Transcripción hecha por mi persona

En la Figura 4 se crea el modulo, este modulo se le llamó *fullAdder* (linea 1), se crean las entradas y las salidas a ser utilizadas en el modulo (lineas 2 - 6). Seguidamente se hace la lógica combinacional para las salidas (lineas 8 - 18), y finalmente se cierra el módulo.

1.1. Referente al banco de pruebas (*fullAdder test bench*).

A continuación se implementa otro módulo que será el banco de pruebas para con esto determinar que el módulo *fullAdder.v* funciona de la manera deseada.

En este banco de pruebas se crean las variables a utilizar para las pruebas (lineas 3 a 6 del código). Seguidamente se instancian las variables recién creadas a las variables del modulo *fullAdder.v* una a una (lineas 8 a 14). Las indicaciones *$dumpfile("test.vcd")* y *$dumpvars(1, fullAdder tb)* dan la orden de descargar todas las variables generadas en el código y de crear un archivo en donde se guardarán los resultados de la simulación.

En la linea 19 se da la orden de que cada variable tenga el valor de un bit. En la linea 20 se da la orden de que corra la simulación durante 10 unidades de tiempo (aún no se ha dado la orden de que se simulen las entradas).

En las líneas de código que van de la 21 a la 28 se dan todos los diferentes posibles valores a cada una de las variables durante diez unidades de tiempo (con excepción de la ultima señal que durará 50 unidades de tiempo). La simulación después podrá ser vista en el programa de visualización de señales *GTKwave* y así se podrá comprobar si la suma de las entradas se corresponde bien con las salidas *sum* y *carryOut*. Finalmente en la linea 30 se da la orden de finalización del test y seguidamente la finalización del módulo (Figura 5).

Figura 5: Elaboración del código del fullAdder$_t b.v$

```
1   module fullAdder_tb;
2
3   //Inputs
4   reg inA_tb, inB_tb, carryIn_tb;
5   //outputs
6   wire sum_tb, carryOut_tb;
7
8   fullAdder DUT (
9       .inA(inA_tb),
10      .inB(inB_tb),
11      .carryIn(carryIn_tb),
12      .sum(sum_tb),
13      .carryOut(carryOut_tb)
14  );
15
16  initial begin
17      $dumpfile("test.vcd");
18      $dumpvars(1, fullAdder_tb);
19      inA_tb = 1'b0; inB_tb = 1'b0; carryIn_tb = 1'b0;
20      #10
21      {inA_tb, inB_tb, carryIn_tb} = 3'b000; #10;
22      {inA_tb, inB_tb, carryIn_tb} = 3'b001; #10;
23      {inA_tb, inB_tb, carryIn_tb} = 3'b010; #10;
24      {inA_tb, inB_tb, carryIn_tb} = 3'b011; #10;
25      {inA_tb, inB_tb, carryIn_tb} = 3'b100; #10;
26      {inA_tb, inB_tb, carryIn_tb} = 3'b101; #10;
27      {inA_tb, inB_tb, carryIn_tb} = 3'b110; #10;
28      {inA_tb, inB_tb, carryIn_tb} = 3'b111; #50;
29
30  end
31
32  endmodule
```

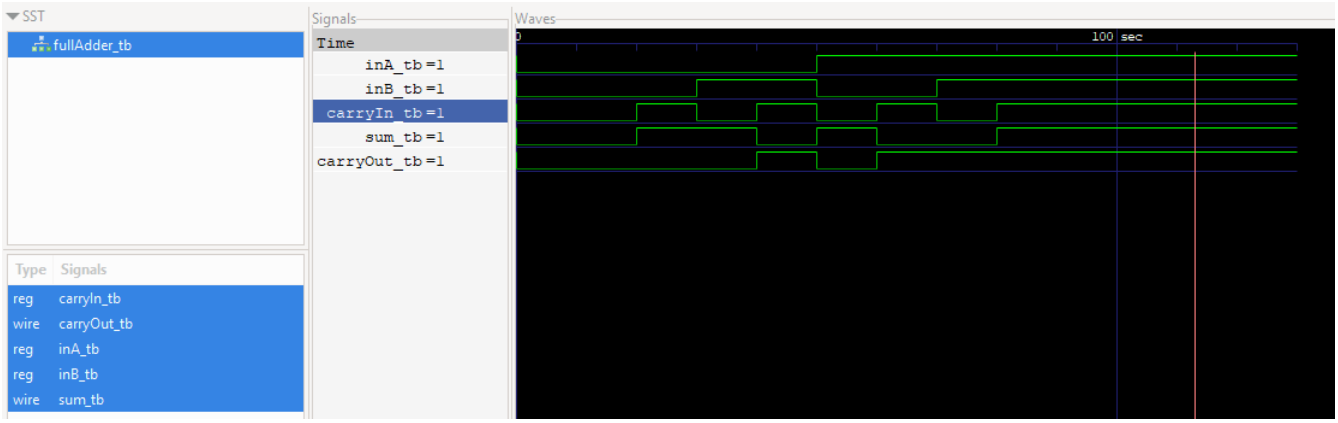
Elaboración: Esteban Badilla / Transcripción hecha por mi persona

Los pasos a seguir para compilar y ejecutar el sumador completo de 1 Bit en la terminal son los siguientes:

- El archivo con el nombre *fullAdder.v*, debe compilar sin problemas bajo el comando: *iverilog fullAdder.v*
- Para compilar el banco de pruebas se debe ingresar el siguiente comando: $iverilog - ofullAdder.ofullAdder.vfullAdder\_tb.v$
- Del punto anterior se obtiene el archivo *fullAdder.o*, este es el ejecutable que se ha creado. Para ejecutarlo se debe ingresar el siguiente comando: *vvp fullAdder.o*
- Una vez ejecutado el comando anterior se obtiene el archivo **test.vcd** el cual es el que contendrá las señales y posteriormente será usado en el programa *GTKwave*.
- Finalmente el archivo *test.vcd* se puede abrir mediante el comando: *GTKwave test.vcd*.
- Al ejecutar el último comando gtkwave test4Bits.vcd se abrirá una ventana en GTKWave en donde se podrá verificar el funcionamiento del sumador completo de 1 Bit.

Ya abierto *GTKwave* se procederá a hacer las verificaciones de las salidas con respecto a las entradas seleccionadas en el modulo de pruebas.
La Figura 6 es una impresión de pantalla de los datos obtenidos en GTKwave.

Figura 6: Captura de pantalla de los datos de simulación obtenidos en GTKwave.



Elaboracion propia.

Para hacer una pequeña mención del correcto funcionamiento del programa, a continuación se mencionarán dos secuencias:

- Se puede ver como en la ventana entre 20 y 30 unidades de tiempo se observa que la variable inAtb = 0, inBtb = 0, carryIntb = 1 y las salidas son: sumtb = 1 y carryOuttb = 0

  Las salidas concuerdan con la Tabla 1

- En la ventana entre 40 y 50 unidades de tiempo se observa que la variable inAtb = 0, la variable inBtb = 1, carryIntb = 1 y las salidas son: sumtb = 0 y carryOuttb = 1

  Valores de salidas que también concuerdan con la Tabla 1

Los demás valores de entradas vs sus respectivas salidas fueron corroboradas también y con un resultado positivo del test.