Proceso para crear un restador.

1. **Creaciòn de los circuitos en DIGITAL (Este es un título Alejo)**

Inicialmente partimos de crear un medio restador usando el software de Digital, para ello realizamos una tabla de verdad con entradas A y B y una salida S, que corresponde a la resta y C\_out, de cómo se realiza la resta, partimos de que:

1-0 = 1

0-1 = 1 y se activa el carry out, indicando que está negativo.

1-1 = 0

0-0 = 0.

Así entonces la tabla de verdad queda:

(Alejo, aquì toca rehacer la tabla de la esquina en excel, donde dice D hay una S)

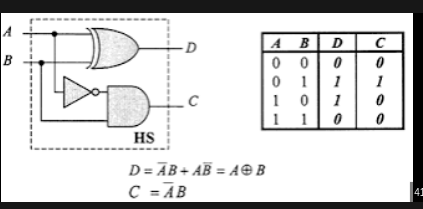


Fig 1. Tabla de verdad medio restador.

Analizando la tabla es facìl ver, que la columna rest, tiene forma de una compuerta XOR cuyo valor serà 1, solo cuando sean diferentes. y para la columna de C\_out tendremos entonces que se parece a una AND, Con la entrada de ‘A’ negada. Construimos el circuito para este medio restador en digital.

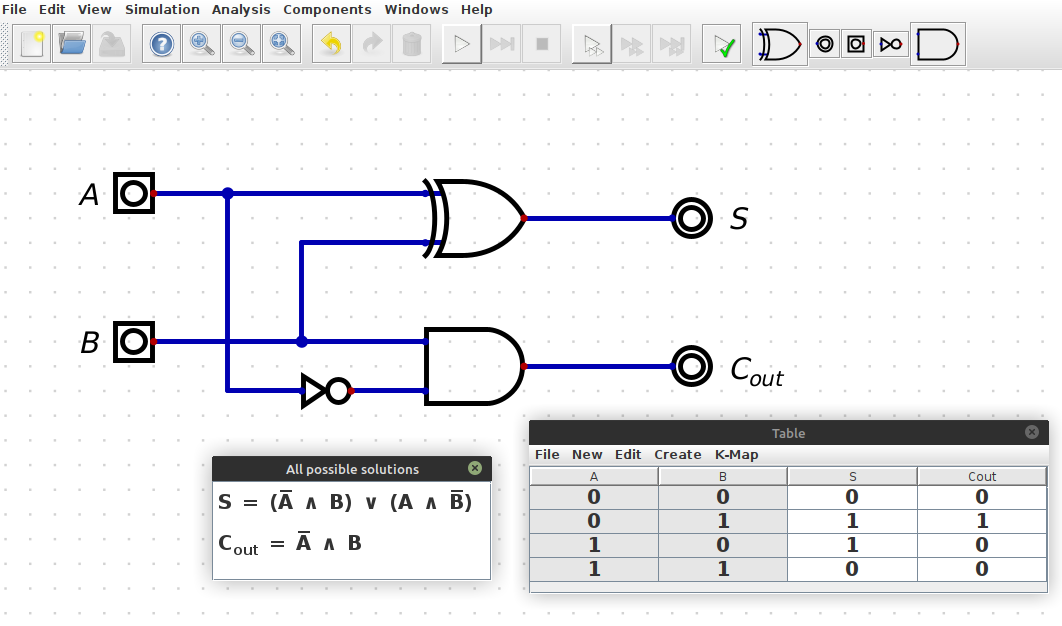


Fig 2. Circuito RTL para el medio restador.

Luego se diseñó en digital un restador completo, en este caso a partir de la tabla de verdad obtenemos su diseño, que contiene una compuerta XOR de 3 entradas, junto con dos AND y OR, ver Fig 3.

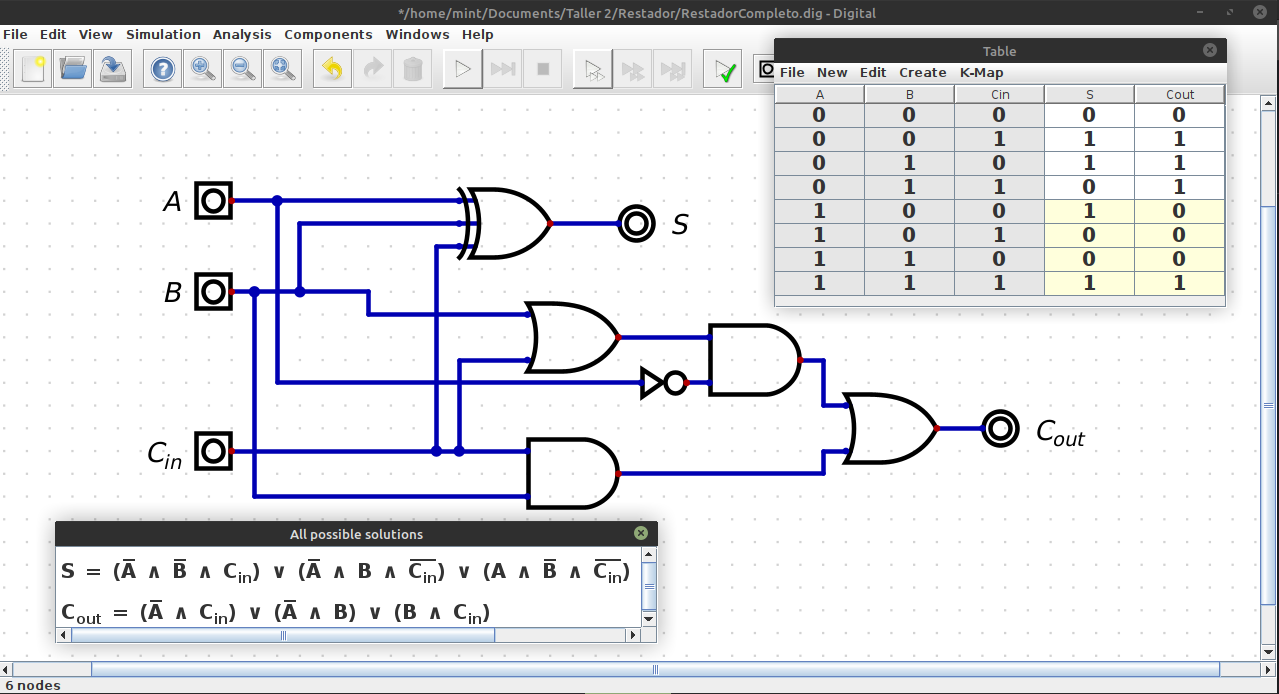


Fig 3. Circuito RTL Restador Completo.

Partiendo de lo anterior, entonces se realiza el restador de 4 bits, que tendrá 8 entradas, es decir dos números de 4 bits, que se restaràn entre sì, se sabe que el c\_0ut de un circuito irà conectado al c\_in del otro, pues se irà acumulado, ya que la resta se harà bit a bit.

Ver figura 4.

A su vez tendremos 5 salidas, 4 correspondientes a los bits restados y una 5, correspondiente al C\_out del sistema total, este indicarà si el número sale o no en complemento a2, en dado caso que sea 1, el número es negativo y los 4 bits deberán ser convertidos de complemento a dos a binario positivo.

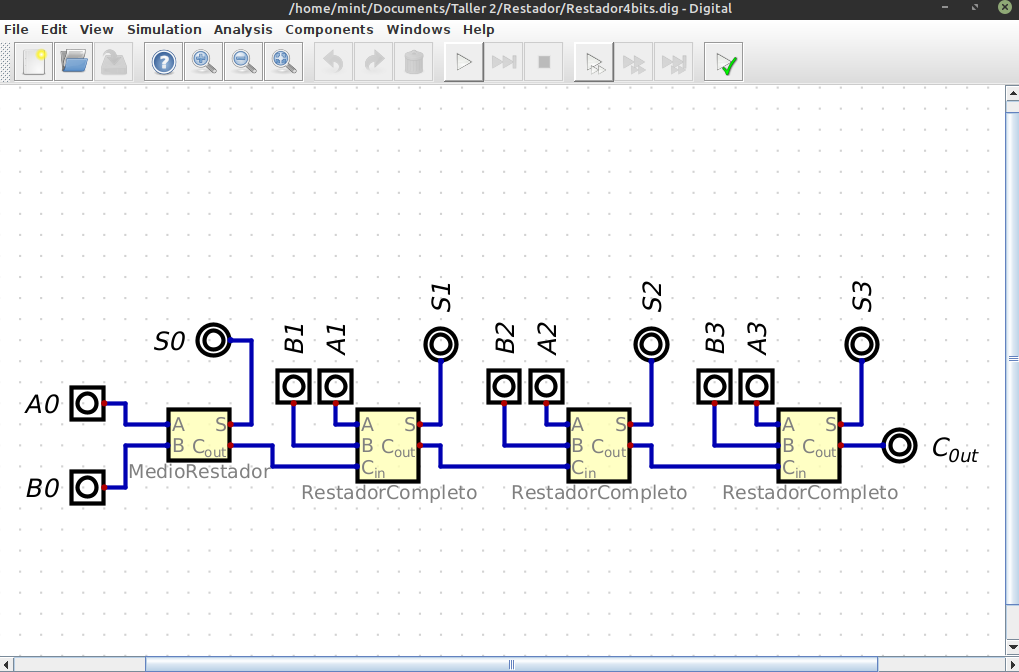


Fig 4. Circuito RTL del Restador de 4 Bits.

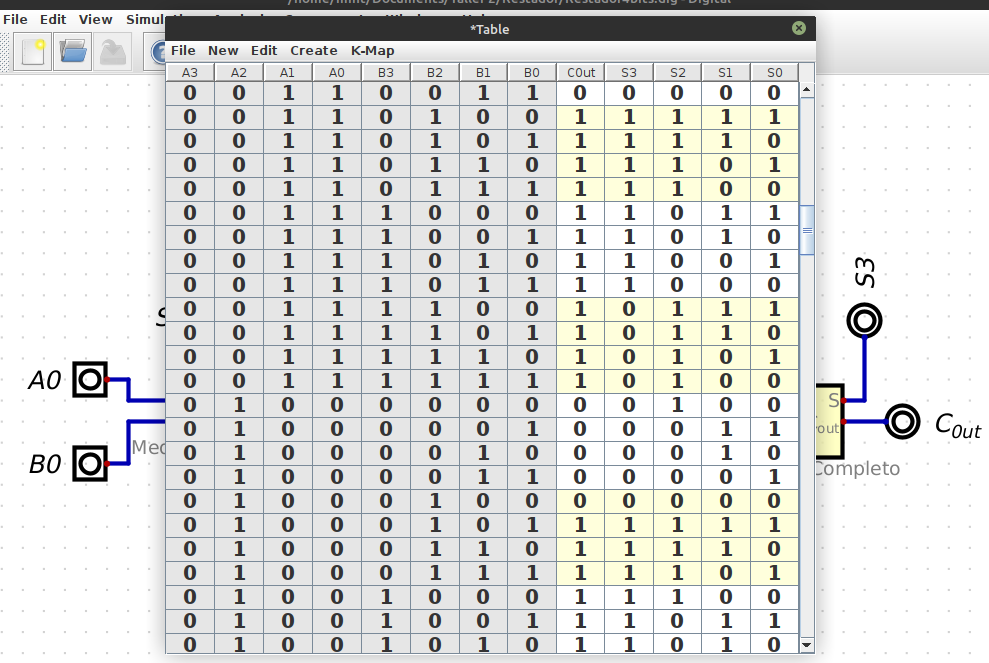


Fig 5. Tabla de Verdad abreviada del restador de 4 bits.

Si se desea obtener la tabla completa, ir al archivo .dig del restador de 4 bits y abrir la tabla de verdad.

En este caso la resta se hace del vector A - el vector B, en dado caso de que B sea mayor a A, el C\_out serà 1, indicando que el número es negativo y está en complemento a dos y debe ser convertido. Como podemos ver en la tercera fila de la tabla en la figura 5, la resta es 0011 que en decimal es 3, con el 0101 que en decimal es 5, lo que nos debe dar -2, en la tabla observamos el valor 11110, el primer dígito correspondiente al C\_out es negativo lo que indica que el valor està en complemento a dos, al pasarlo queda 0010, que corresponde al 2, y el C\_out negativo, actúa como un menos.

Si por el contrario, el C\_out tiene valor de 0, la respuestà està correcta y no debe pasarse a complemento a dos.

Como en el caso 0100 = A y B=0001 , en decimal la resta es 4-1 que da 3 positivo, es decir 00011.

1. **Implementaciòn de los circuitos en Verilog y Geany**

2.1 Para comprobar que el circuito funciona correctamente, lo exportamos a Verilog desde Digital y lo abrimos en geany. ver figura 6.

2.2 Ahì creamos un archivo \_tb.v que serà nuestra testbech, en este caso se ajustó, teniendo en cuenta las 8 entradas, creando un array de [7:0] inputs; y su variaciòn hasta 256, con un for. A cada valor del array se le asoció un valor de las entradas del restador de cuatro bits. para mejor detalle ver figura 7. Así mismo, se creó un array de [4:0], con 5 espacios para las salida outs y se asignó a cada salida un valor.

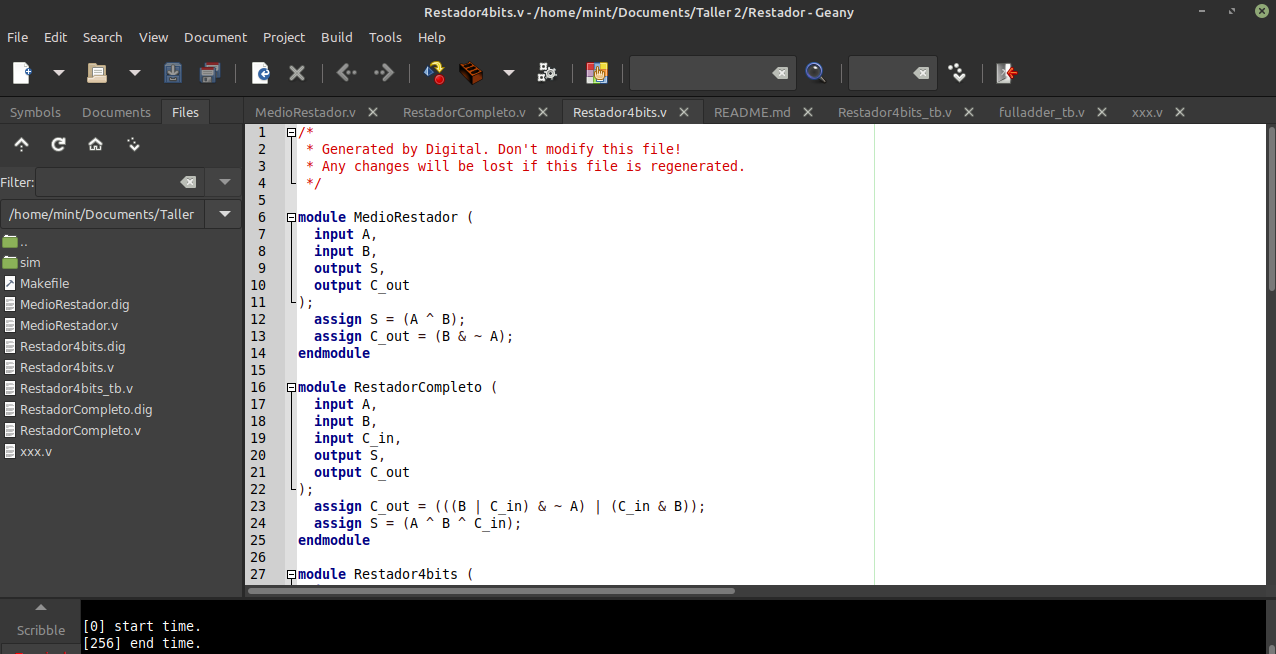


Fig 6. Archivo verilog exportado del Restador 4 bits.

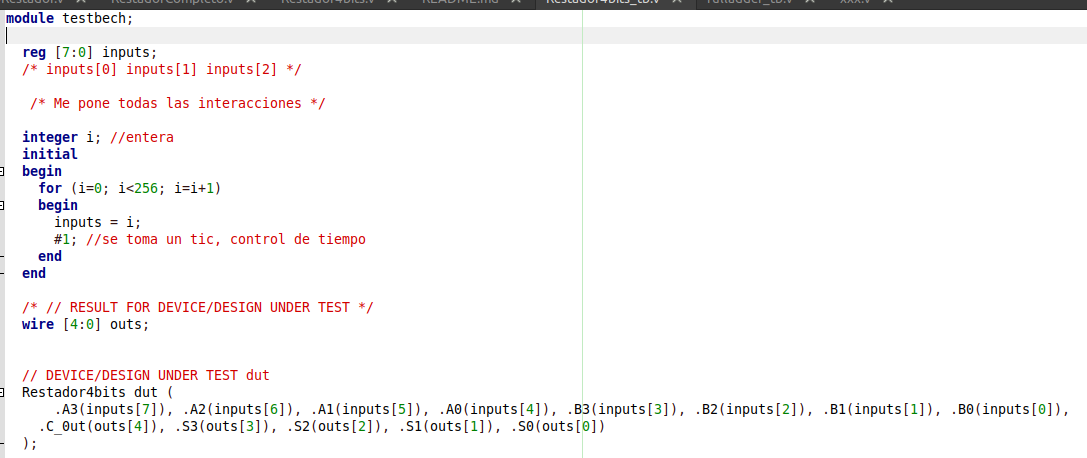


Fig 7. Código Verilog del testbech del Restador de 4 bits.

2.3 Finalmente, teniendo el testbech correctamente, se usan los códigos siguientes para crear el archivo que se abrirá en gtkwave.

// iverilog -o top.vvp Restador4bits\_tb.v Restador4bits.v

// vvp top.vvp

// gtkwave top.vcd

1. **Implementaciòn de los circuitos en gtkwave**

Finalmente, con el código anterior se nos abre la ventana de gtkwave, aquí podemos observar la señal binaria correspondiente. Ver figura 8.

Como podemos observar, se presenta el caso en el que el negativo es mayor que el positivo y ahì tenemos un C\_0ut que està asignado en outs[4]= 1 y el resto de valores en 1, esto quiere decir que deben ser convertidos de complemento a dos a su forma binaria. Que da como resultado 0001 es decir 1, y el c\_out indica menos 1.

Obsérvese, que el bit para el vector A, de menor valor es inputs[4] y para el vector B es inputs[0], así entonces obtenemos.

Vector A = [0000] B =[0001] lo que en decimal sería A-B= -1 = 0 - 1=-1.

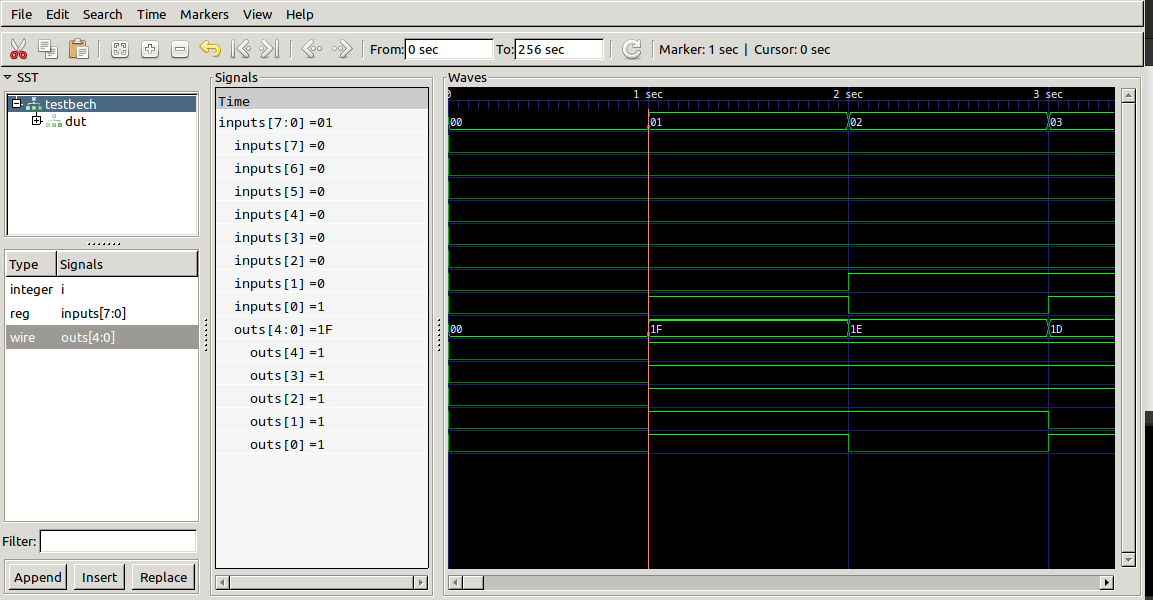


Fig 8. Imagen de gtkwave circuito Restador de 4 bits, C\_out negativo.

Por otro lado, tenemos el caso en el que, el C\_out da positivo, indicando que el número mayor es positivo, por ejemplo veamos el ejemplo de la figura 9. tenemos:

A=[1010] y B= [0100] asi entonces A-B= 10 - 4 = 6 positivo.

Si nos fijamos en los Outs, el outs[4] que corresponde al C\_out està en 0, el numero no debe ser convertido de complemento a dos. el resultado de la salida es:

S= [00110] = 6.

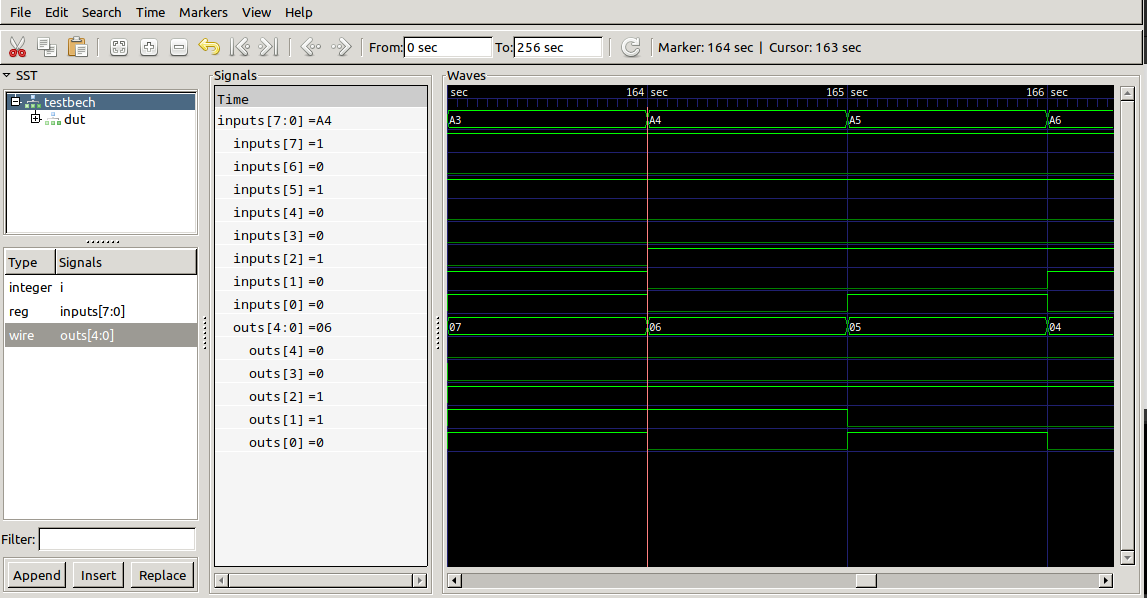


Fig 9. Imagen de gtkwave circuito Restador de 4 bits, C\_out positivo.

Una cosa curiosa para notar, es que en la gráfica el gtkwave escribe los valores en Hexadecimal.