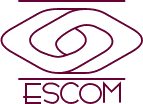
f

**INSTITUTO POLITÉCNICO NACIONAL**

# ESCUELA SUPERIOR DE CÓMPUTO

**DEPARTAMENTO DE INGENIERÍA EN SISTEMAS COMPUTACIONALES**

**PRACTICA No. 6**

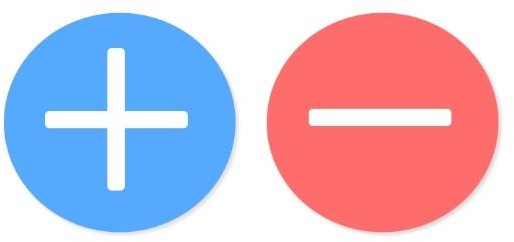
Operaciones Aritméticas

Nombre: Silva Hernandez Noe Jasiel

Grupo: 2CV1

Materia: Fundamentos de Diseño Digital





# Objetivo general.

El alumno diseñará circuitos aritméticos programando en lenguaje VHDL y cargándolo a su GAL 22V10 para verificar físicamente el resultado y la buena operación del circuito.

# Material empleado.

* + 1 Circuito Integrado GAL22V10.
  + 10 LEDS de colores.
  + 10 Resistores de 330Ω.
  + 8 Resistores de 1KΩ.
  + 1 Dip switch de 8.
  + Alambre telefónico.
* 1 Tablilla de Prueba (Protoboard).
* Pinzas de punta.
* Pinzas de corte.
* Cables Banana-Caimán (para alimentar el circuito).

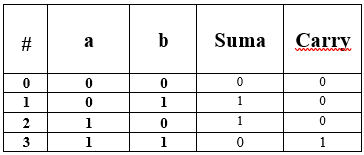
# Equipo empleado.

* + Multímetro.
  + Fuente de Alimentación de 5 Volts.
  + Programador Universal.

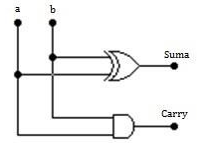
# Desarrollo Experimental y Actividades.

**DESARROLLO**

Un medio sumador es un sumador capaz de sumar dos datos de un sólo bit y producir un bit de acarreo de salida.

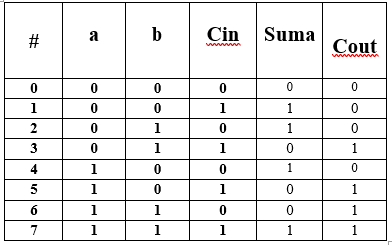
Esto nos deja una tabla de verdad como la siguiente

de este se saca el mapa K para poder hacer su máxima simplificaion, después se procede a realizar el circuito resultante



dibujándolo nos quedaría algo asi… después de esto se procede a programarlo y después a simularlo para comprobar el perfecto funcionamiento… verificándose en la variable de salida S y C que son la suma y el carry comparándose con la tabla de verdad

Sin embargo el medio sumador no puede ser interconectado con otros medios sumadores para formar un sumador más grande, por ello es necesario diseñar un sumador que admita otra entrada aparte de los datos a sumar, es decir, un sumador de 3 datos de 1 bit, éste se le llama sumador completo



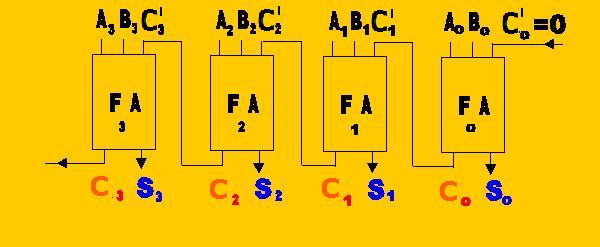
Y de aquí se hace el mismo procedimiento, sacamos la tabla de verdad después hacemos el mapa K de la suma y el carri para poder sacar la máxima simplificación

Después de esto dibujamos el circuito resultante y luego se procede a programas y verificar los resultados como en el anterior

Y asi para el medio restador y restador completo.

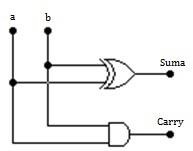
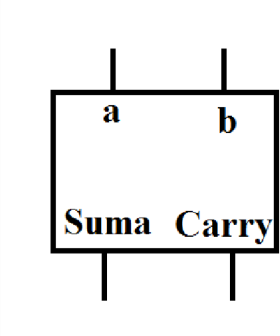
En el caso del sumador de 4 bits se utilizo la dinámica del sumador completo conectados en serie, esto significa que el acarreo de salida del sumador se conecta al acarreo de entrada del siguiente sumador esto lo hacemos 4 veces y nos deja nuestro sumador de 4 bits (como en la imagen), el ultimo bit de acarreo es el que se mostrara al usuario dándonos 2^4 = 16

En cuanto a sus salida es el primer bit de derecha a izquierda es el mas significativo



**4.1.- Medio Sumador**

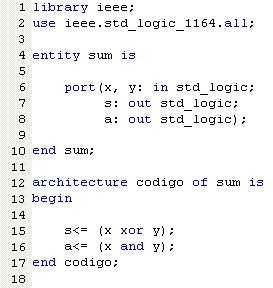
1. Diseñe el siguiente circuito y analícelo para obtener las funciones lógicas de cada una de las salidas.

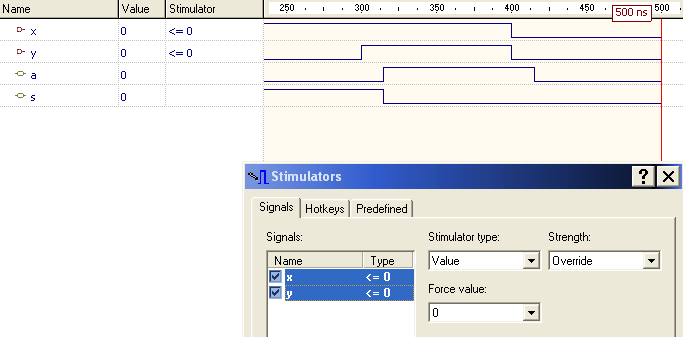


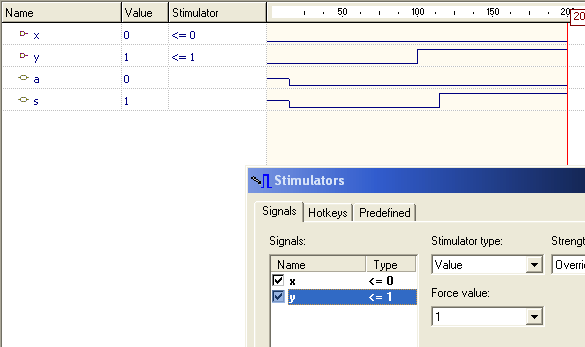
1. Implemente su código en VHDL, junto con el informe de pines RPT.
2. Arme el circuito y compruebe la tabla de verdad.

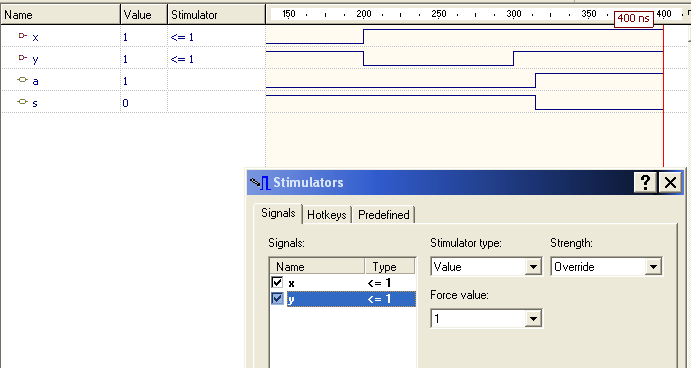
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **a** | **b** | **Suma** | **Carry** |
| **0** | **0** | **0** | 0 | 0 |
| **1** | **0** | **1** | 1 | 0 |
| **2** | **1** | **0** | 1 | 0 |
| **3** | **1** | **1** | 0 | 1 |

Para el caso del medio sumador se agrega el código en VHDL a manera de guía, a partir de la cual el alumno hará las adecuaciones pertinentes para los siguientes casos.



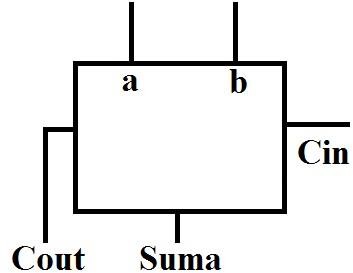
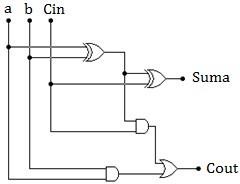






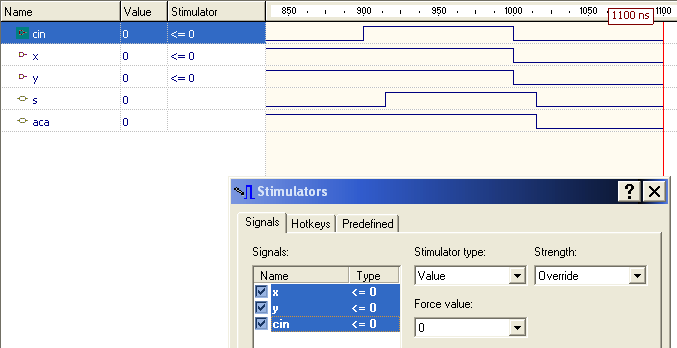
# 4.2.- Sumador Completo.

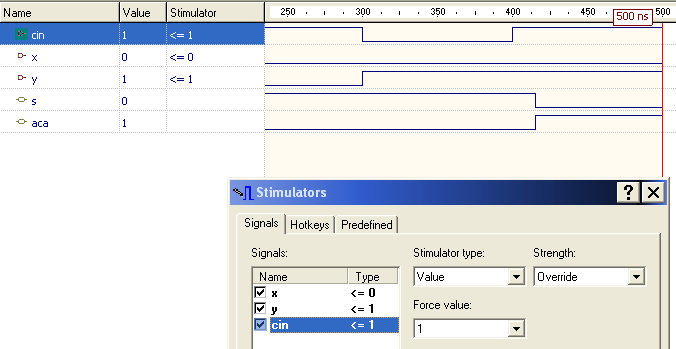
1. Diseñe el siguiente circuito y analícelo para obtener las funciones lógicas de cada una de las salidas.

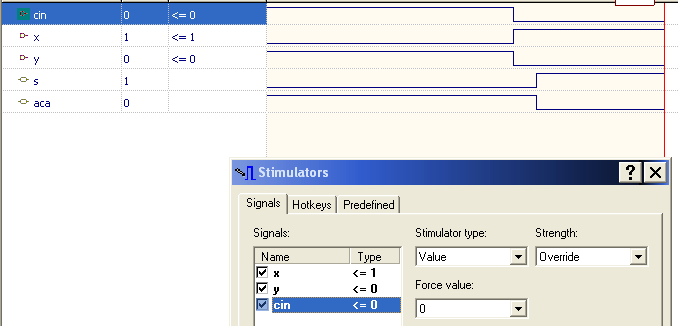


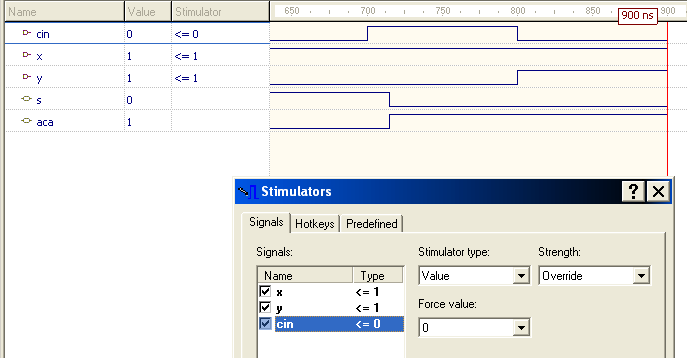
1. Implemente su código en VHDL, junto con el informe de pines RPT.
2. Arme el circuito y compruebe la tabla de verdad.

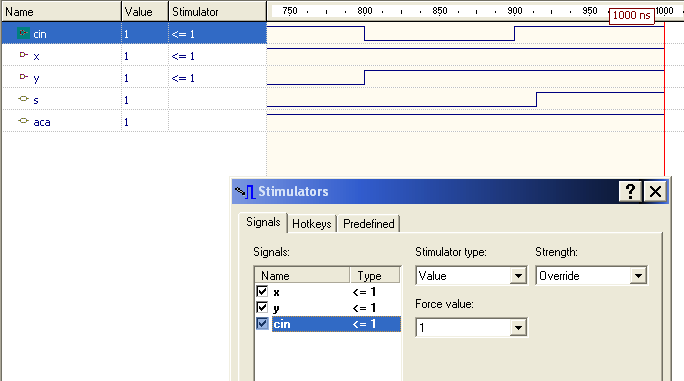
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **a** | **b** | **Cin** | **Suma** | **Cout** |
| **0** | **0** | **0** | **0** | 0 | 0 |
| **1** | **0** | **0** | **1** | 1 | 0 |
| **2** | **0** | **1** | **0** | 1 | 0 |
| **3** | **0** | **1** | **1** | 0 | 1 |
| **4** | **1** | **0** | **0** | 1 | 0 |
| **5** | **1** | **0** | **1** | 0 | 1 |
| **6** | **1** | **1** | **0** | 0 | 1 |
| **7** | **1** | **1** | **1** | 1 | 1 |





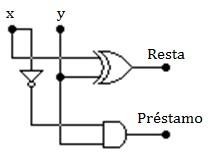
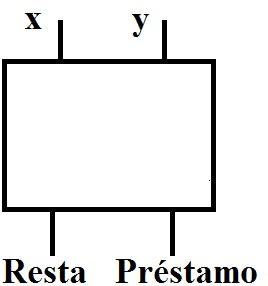






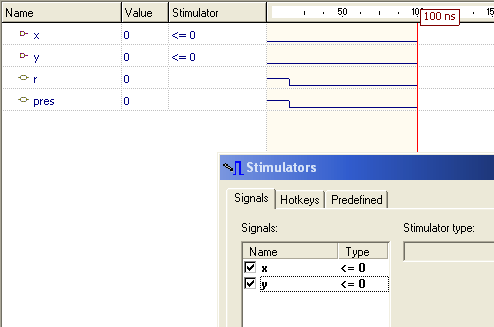
# 4.3.- Medio Restador.

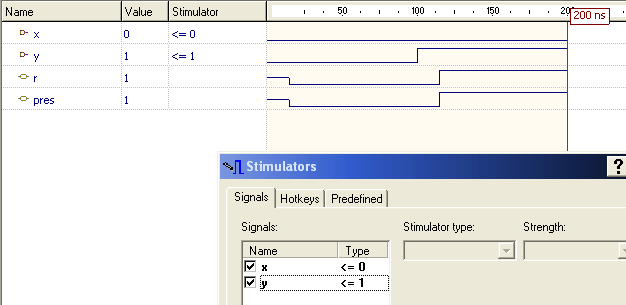
1. Diseñe el siguiente circuito y analícelo para obtener las funciones lógicas de cada una de las salidas.

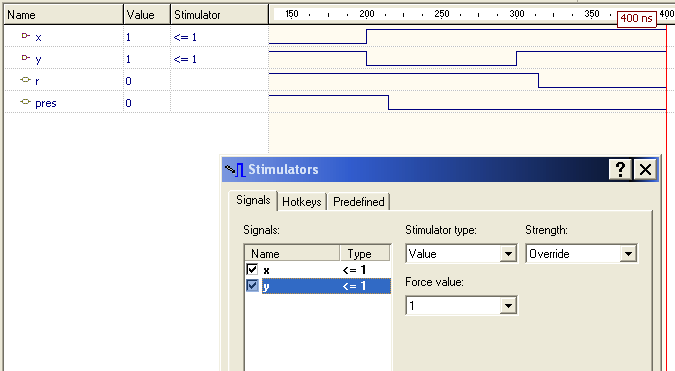


1. Implemente su código en VHDL, junto con el informe de pines RPT.
2. Arme el circuito y compruebe la tabla de verdad.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **a** | **b** | **Resta** | **Prestamo** | |
| **0** | **0** | **0** | 0 | 0 |
| **1** | **0** | **1** | 1 | 1 |
| **2** | **1** | **0** | 1 | 0 |
| **3** | **1** | **1** | 0 | 0 |



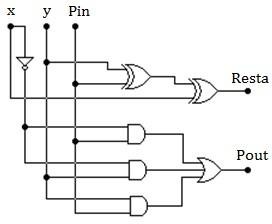
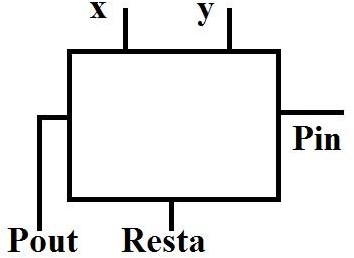




|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **x** | **y** | **Resta** | **Préstamo** |
| **0** | **0** | **0** |  |  |
| **1** | **0** | **1** |  |  |
| **2** | **1** | **0** |  |  |
| **3** | **1** | **1** |  |  |

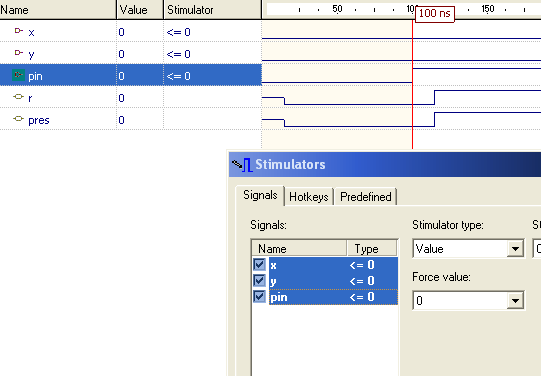
# 4.4.- Restador Completo.

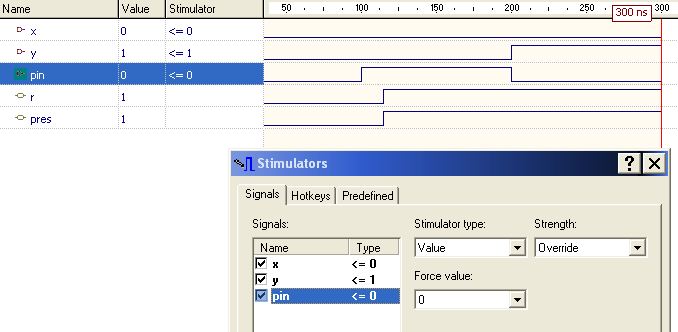
a) Diseñe el siguiente circuito y analícelo para obtener las funciones lógicas de cada una de las salidas.

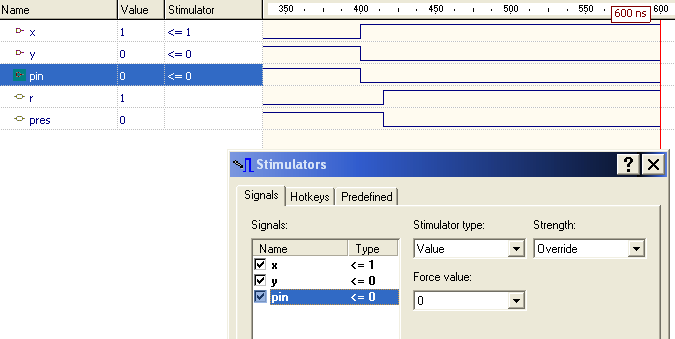


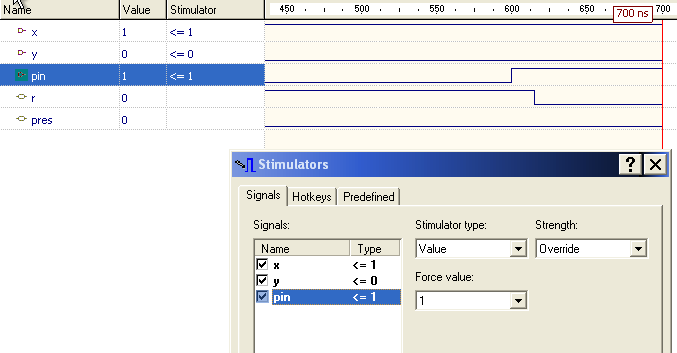
1. Implemente su código en VHDL, junto con el informe de pines RPT.
2. Arme el circuito y compruebe la tabla de verdad.

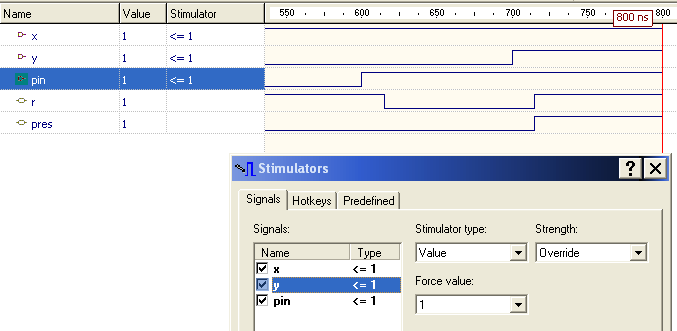
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **x** | **y** | **Pin** | **Resta** | **Pout** |
| **0** | **0** | **0** | **0** | 0 | 0 |
| **1** | **0** | **0** | **1** | 1 | 1 |
| **2** | **0** | **1** | **0** | 1 | 1 |
| **3** | **0** | **1** | **1** | 0 | 1 |
| **4** | **1** | **0** | **0** | 1 | 0 |
| **5** | **1** | **0** | **1** | 0 | 0 |
| **6** | **1** | **1** | **0** | 0 | 0 |
| **7** | **1** | **1** | **1** | 1 | 1 |





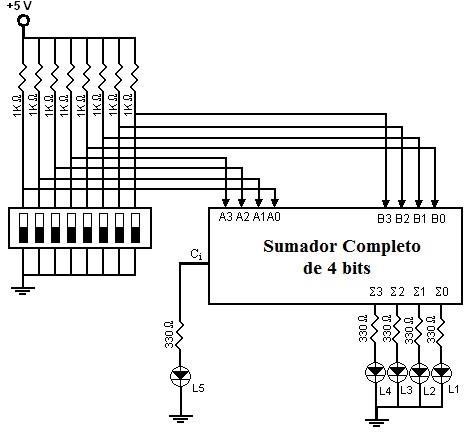






# 4.5.- Sumador Completo.

1. Diseñe el siguiente circuito y analícelo para obtener las funciones lógicas de cada una de las salidas.
2. Arme el circuito y compruebe la tabla de verdad.

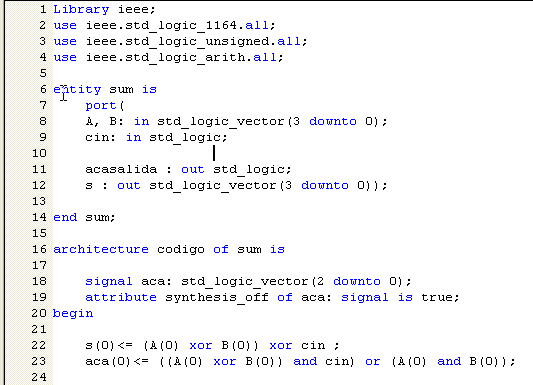


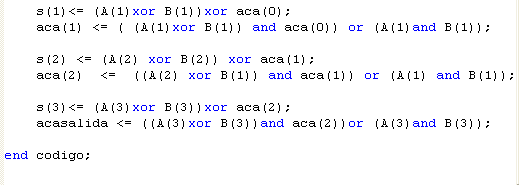
# Verifique algunas sumas que usted establezca y confirme sus resultados

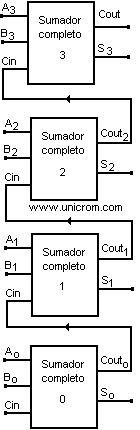
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Ci** | **A3** | **A2** | **A1** | **A0** | **B3** | **B2** | **B1** | **B0** | **Co** | **Σ3** | **Σ2** | **Σ1** | **Σ0** |
| **0** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| **2** | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| **3** | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| **4** | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| **5** | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| **6** | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| **7** | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| **8** | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| **9** | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| **10** | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |

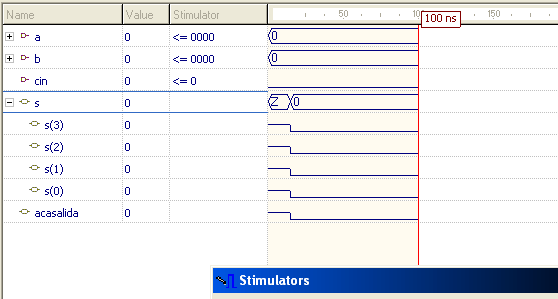
**SUMADOR DE 4 BITS**

**CODIGO----**

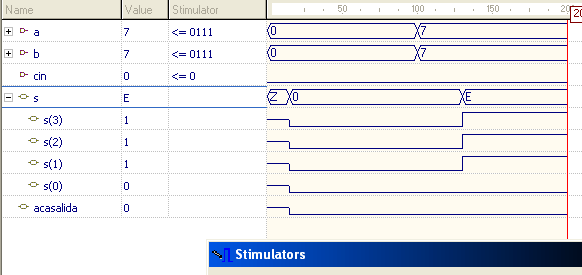


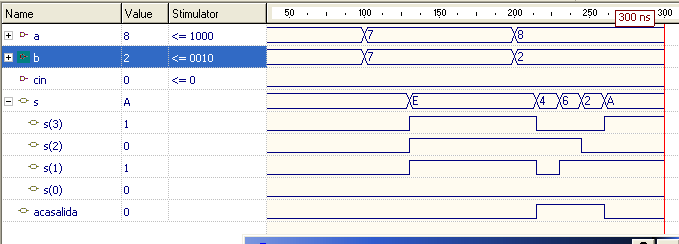


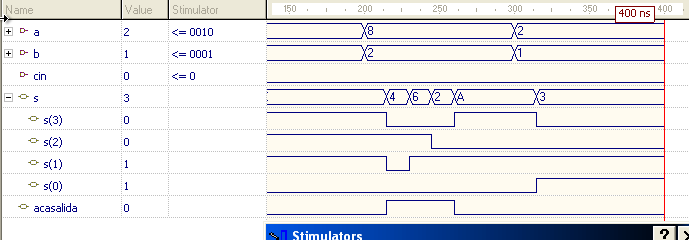


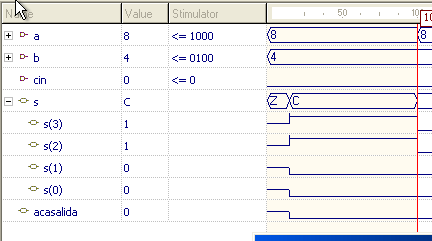


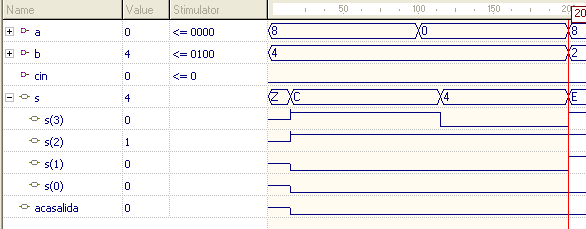
**En este tenemos 7 + 7 = 14 en hexadecimal es E**

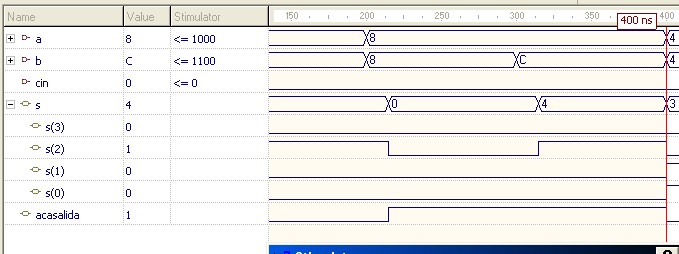


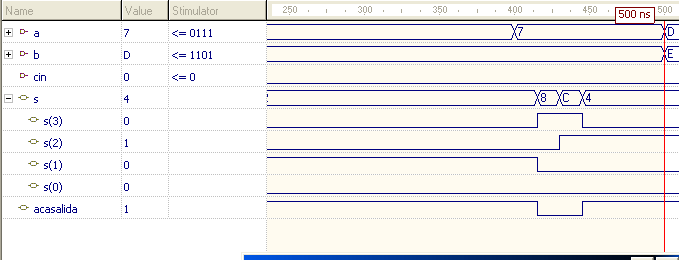


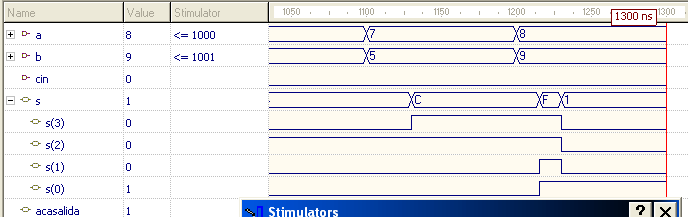












1. **Conclusiones Individuales.**

El sumador binario es la parte fundamental de todos los circuitos, ya que mediante sumas es posible realizar restas, multiplicaciones y divisiones,

Al programar este codigo nos ahorrara muchísimo tiempo al querer hacer alguna resta o suma si bien para nosotros o en algún proyecto individual seria mucho trabajo… pero hablando de algo mas elaborado que requiera sumas o restas constantemente este codigo seria super efectivo…

Como en el caso del sumador de 4 bits podemos representar hasta 2^4 = 16 con el bit de acarreo.