INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

INGENIERIA EN SISTEMAS COMPUTACIONALES

MATERIA: DISEÑO DE SISTEMAS DIGITALES

PROFESOR: TESTA NAVA ALEXIS

PRESENTA:

RAMIREZ BENITEZ BRAYAN

GRUPO: 2CV18

PRACTICA 7:

ALGORITMO DE BOOTH

CIUDAD DE MEXICO JUNIO DE 2021

Para esta práctica utilizaremos una CPLD la PSI C25G01K100 en el Simulador Galaxy, esta práctica consistió en diseñar un circuito que realiza operaciones de números binarios con 3 bits haciendo uso del algoritmo de booth dónde mostrara en el resultado de esta operación en la salida R, a continuación, la información vista en clase, las salidas y entradas que ocupamos y el código correspondiente para la práctica.

Algoritmo de Booth.

El algoritmo de Booth es un método rápido y sencillo para obtener el producto de dos números binarios con signo en notación complemento a dos. El algoritmo de Booth examina pares adyacentes de bits del multiplicador Y de N-bits en la representación de complemento a dos con signo, incluyendo un bit implícito debajo del bit menos significativo, y-1 = 0. Para cada bit yi, para i corriendo desde 0 hasta N-1, los bits yi e yi-1 son considerados. Cuando estos dos bits son iguales, el acumulador del producto P es dejado sin cambios. Cuando yi = 0 e yi-1 = 1, el multiplicando multiplicado por 2i es agregado a P; y cuando yi = 1 e yi-1 = 0, el multiplicando multiplicado por 2i es restado de P. El valor final de P es el producto con signo.

Bit 1	Bit 2	Operación
0	0	No hacer nada
0	1	P + A
1	0	P + S
1	1	No hacer nada

Procedimiento

Supongamos dos números, multiplicando y multiplicador, con longitudes en bits, x para el primero, e Y para el segundo:

•Construimos una matriz de tres filas y x+y+1 columnas. Identificaremos las filas como, A la primera, S la segunda y P la tercera.

•Se inician los x primeros bits de cada fila con:

- •A, el multiplicando.
- •S, el complemento a dos del multiplicando.
- •P, ceros.

•Los siguientes y bits se completan con:

•A, ceros.

•S, ceros.

•P, el multiplicador.

•Para finalizar la matriz, se inician a 0 todos los valores de la última columna. Una vez iniciada esta matriz, se realiza el algoritmo.

•Se realizan y iteraciones del siguiente bucle.

1. Comparar los dos bits menos significativos de P, para realizar la siguiente acción:
   •00 o 11: no se hace nada.
   •01: P = P + A. Se ignora el desbordamiento (overflow).
   •10: P = P + S. Se ignora el desbordamiento.
2. Desplazamiento aritmético de P a la derecha (se conserva el bit de signo).

•Finalmente, tras y iteraciones, se elimina el último bit de la derecha (menos significativo), obteniendo el resultado.

Ejemplo:

1) N1= 0101 -> multiplicando, longitud x= 4
   N2= 0011 -> multiplicador, longitud, y= 4

2) Se crea la matriz de 3 filas por x+y+1= 9 columnas.

3) Se nombran las filas A, S, P

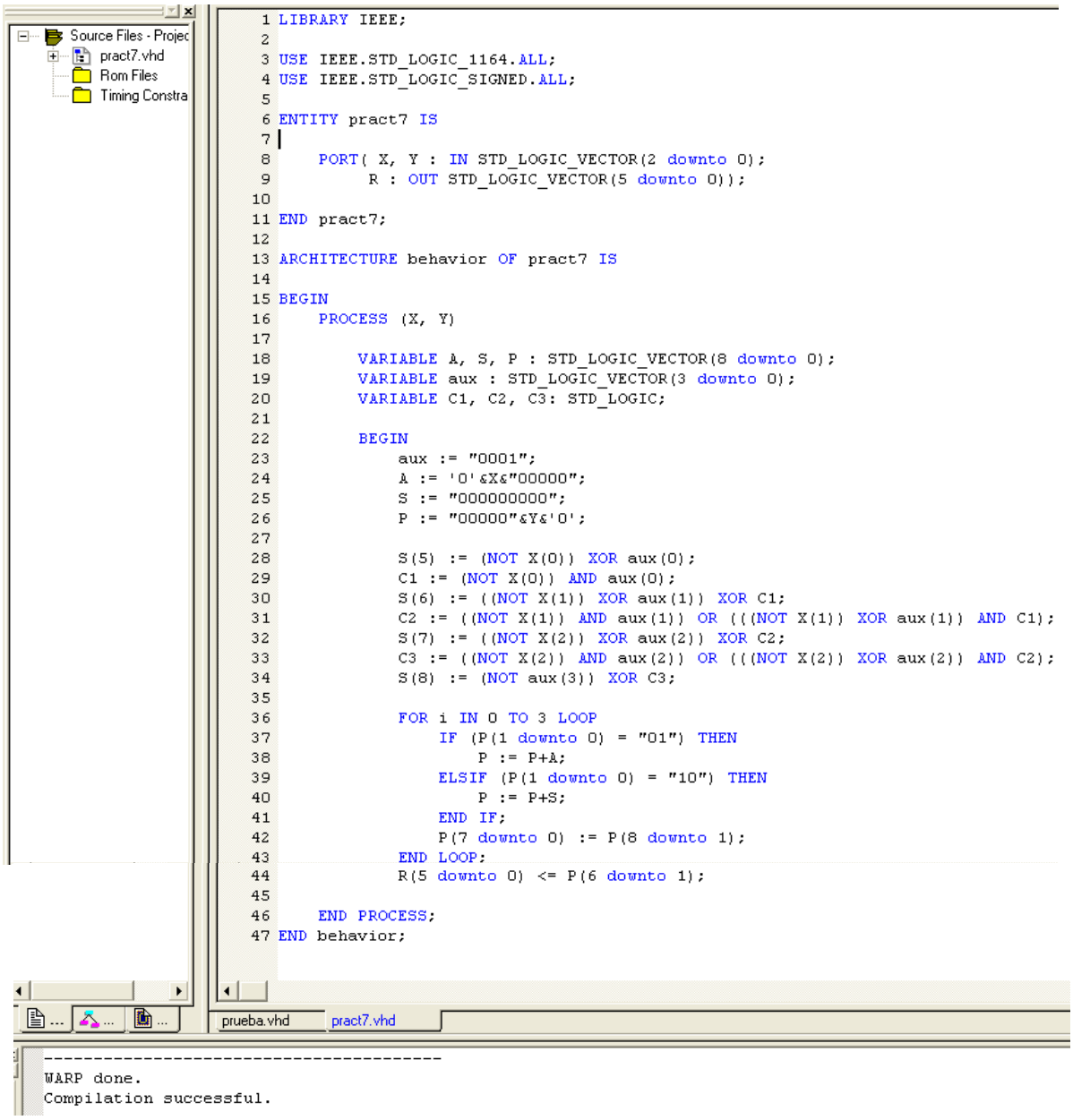
4) Se colocan los valores de N1, el complemento a dos de N1 y N2.

5) número de iteraciones= 4

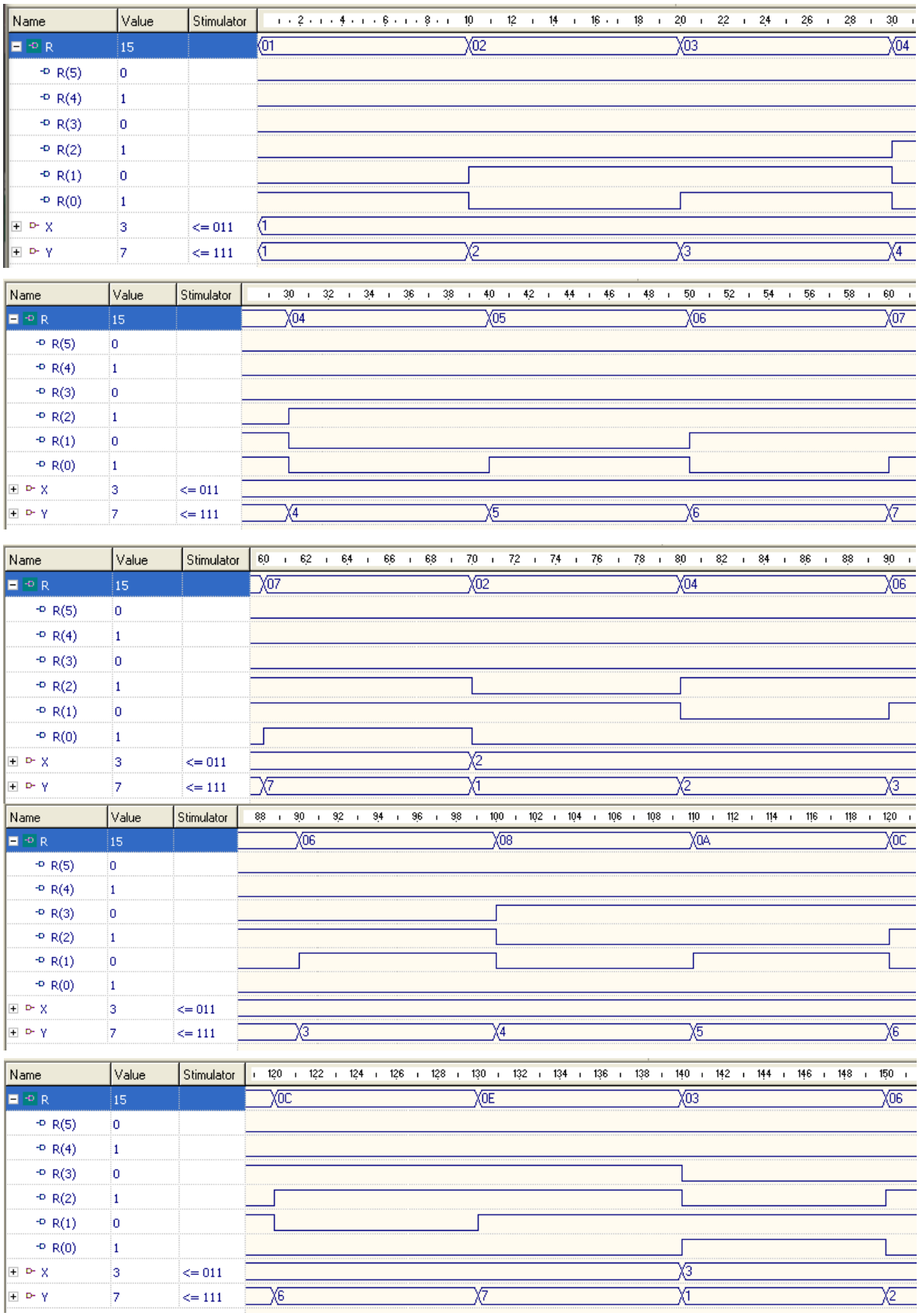
A	0	1	0	1	0	0	0	0	0	
S	1	0	1	1	0	0	0	0	0	
P'	0	0	0	0	0	0	1	1	0	P+S
P'	1	0	1	1	0	0	1	1	0	→
P'	1	1	0	1	1	0	0	1	1	→
P'	1	1	1	0	1	1	0	0	1	P+A
P'	0	0	1	1	1	1	0	0	1	→
P'	0	0	0	1	1	1	1	0	0	→
P	0	0	0	0	1	1	1	1	0	fin

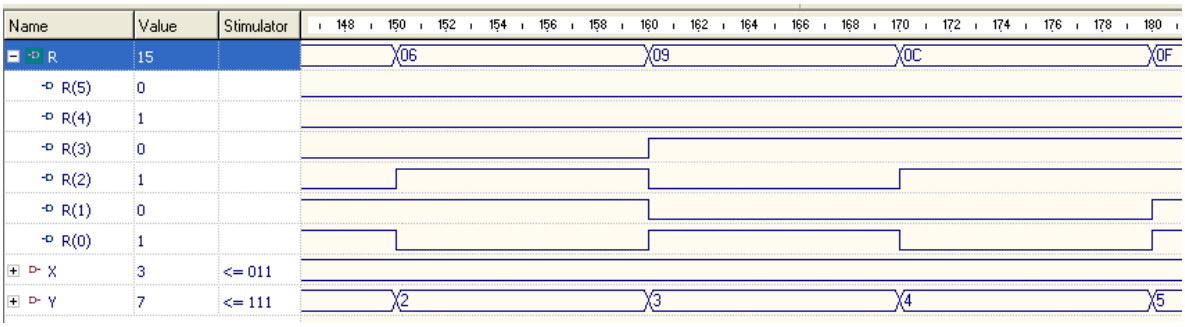
En amarillo se representa cada iteración

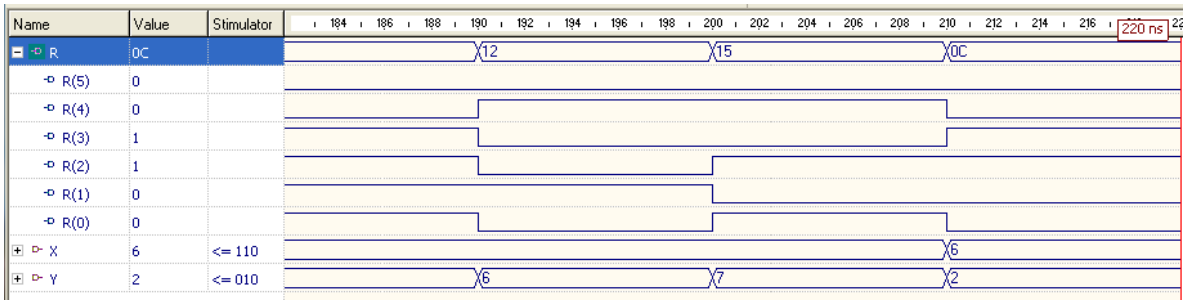
Código y simulación.




```
LIBRARY IEEE;

USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_SIGNED.ALL;

ENTITY pract7 IS

    PORT( X, Y : IN STD_LOGIC_VECTOR(2 downto 0);
          R : OUT STD_LOGIC_VECTOR(5 downto 0));

END pract7;

ARCHITECTURE behavior OF pract7 IS

BEGIN
    PROCESS (X, Y)

        VARIABLE A, S, P : STD_LOGIC_VECTOR(8 downto 0);
        VARIABLE aux : STD_LOGIC_VECTOR(3 downto 0);
        VARIABLE C1, C2, C3: STD_LOGIC;

        BEGIN
            aux := "0001";
            A := '0'&X&"00000";
            S := "000000000";
            P := "00000"&Y&'0';

            S(5) := (NOT X(0)) XOR aux(0);
            C1 := (NOT X(0)) AND aux(0);
            S(6) := ((NOT X(1)) XOR aux(1)) XOR C1;
            C2 := ((NOT X(1)) AND aux(1)) OR (((NOT X(1)) XOR aux(1)) AND C1);
            S(7) := ((NOT X(2)) XOR aux(2)) XOR C2;
            C3 := ((NOT X(2)) AND aux(2)) OR (((NOT X(2)) XOR aux(2)) AND C2);
            S(8) := (NOT aux(3)) XOR C3;

            FOR i IN 0 TO 3 LOOP
                IF (P(1 downto 0) = "01") THEN
                    P := P+A;
                ELSIF (P(1 downto 0) = "10") THEN
                    P := P+S;
                END IF;
                P(7 downto 0) := P(8 downto 1);
            END LOOP;
            R(5 downto 0) <= P(6 downto 1);

    END PROCESS;
END behavior;
```


Name
Value
Stimulator
R
15
R(5)
0
R(4)
1
R(3)
0
R(2)
1
R(1)
0
R(0)
1
X
3
<= 011
Y
7
<= 111
01
02
03
04
1
2
3
4
05
06
07
5
6
08
0A
0C
0E





Observaciones y conclusiones.

Durante el desarrollo de esta practica tuve bastantes dificultades puesto que encontrar una manera de simular este algoritmo sin utilizar las librerías SIGNED es complicado, pero al usar esta librería generaba errores por la cantidad de operaciones que debía ejecutar en la GAL C22V10, ya que la gal no tenía la capacidad suficiente para implementar esta práctica entonces utilicé una CPLD para hacer uso de esta librería, aun así para encontrar el complemento a 2 de X lo implemente sin hacer uso de esta librería para ejemplificar que si es posible hacer una suma, donde primero negamos X y luego le sumamos un 1 binario y de esta manera encontramos el complemento a 2 de X, si observamos este código es un poco extenso para 4 bits entonces hacerlo para 9 bits es posible pero bastante extenso, es por esto que lo hice de esta manera haciendo uso del operado '+' además que es más entendible y sencillo.

Además, la gal C22V10 genera los siguientes errores al compilarlo

```
Error: Logic equation has too many product terms on signal r(5).
Error: Logic equation has too many product terms on signal r(4).
Error: Logic equation has too many product terms on signal r(3).
Error: Logic equation has too many product terms on signal r(2).
Error: Logic equation has too many product terms on signal \MODULE_17:ss_7\.
Error: Logic equation has too many product terms on signal \MODULE_17:ss_6\.
Error: Logic equation has too many product terms on signal \MODULE_15:ss_6\.
```

Anexos y bibliografía

Referencias bibliográficas:

Floyd, T. L. (2021). *Fundamentos De Sistemas Digitales* (9.ª ed.) [Libro electrónico]. PRENTICE HALL/PEARSON. https://www.academia.edu/34699883/Libro_fundamentos_de_sistemas_digitales_floyd_9ed_PDF

Maxinez, D. (2013). *Programación de sistemas digitales con VHDL* (1.ª ed.). Grupo Editorial Patria. https://editorialpatria.com.mx/pdffiles/9786074386219.pdf

Referencias electrónicas:

Algoritmos de Booth. (s. f.). matematicas-discreta. Recuperado 27 de mayo de 2021, de https://trabajoenequipoitq.wixsite.com/matematicas-discreta/14-algoritmos-de-booth

Algoritmos de Booth para la multiplicación y división en binario. - Matematicas Discretas evz. (s. f.). Algoritmos de Booth para la multiplicación y división en binario. Recuperado 27 de mayo de 2021, de https://sites.google.com/site/matematicasdiscretasevz/1-4-algoritmos-de-booth-para-la-multiplicacion-y-division-en-binario