Signed Multiplier (Booth)

Nahin Ul Sadad
Lecturer,
CSE, RUET

Signed Number

Here, Sign number will mean 2’s complement number.

For example,

1011 (-5)

is

0100 (1’s complement) + 1

=

0101 (+5) ‘s Negative version.

Signed Multiplication Example

For Signed number,

```
    1011  (-5)
    1101  (-3)
 ________
 00001011
 0000000X
 001011XX
 01011XXX
 ________
 10001111 (-113)
```

The "Binary" Multiplication Table

*	0	1
0	0	0
1	0	1

We can see that multiplying 2 4-bit signed binary numbers do not work like unsigned numbers.

Signed Multiplication Example

For Signed number,

```
1011 (-5)
1101 (-3)
```

This numbers are not signed numbers. Since multiplying 2 4-bit numbers result in 8 bit product, so multiplicand and multiplier must be sign extended.

```
11111011 (-5)
11111101 (-3)
```

Since numbers are negative numbers, they are sign extended with 1s.

Signed Multiplication Example

For Signed number,

```
        11111011  (-5)
        11111101  (-3)
       ----------
        11111011
        0000000X
        111011XX
        11011XXX
        1011XXXX
        101XXXXX
        10XXXXXX
        1XXXXXXX
       ----------
    101|00001111  (+15)
```

The "Binary" Multiplication Table

*	0	1
0	0	0
1	0	1

We can **calculate it like unsigned numbers** just by **sign extending it** and also by **performing twice many steps** (8 steps vs 4 steps) than unsigned numbers.

Booth Multiplication Algorithm

One of most efficient algorithm to perform multiplication with signed numbers is **Booth Multiplication Algorithm**.

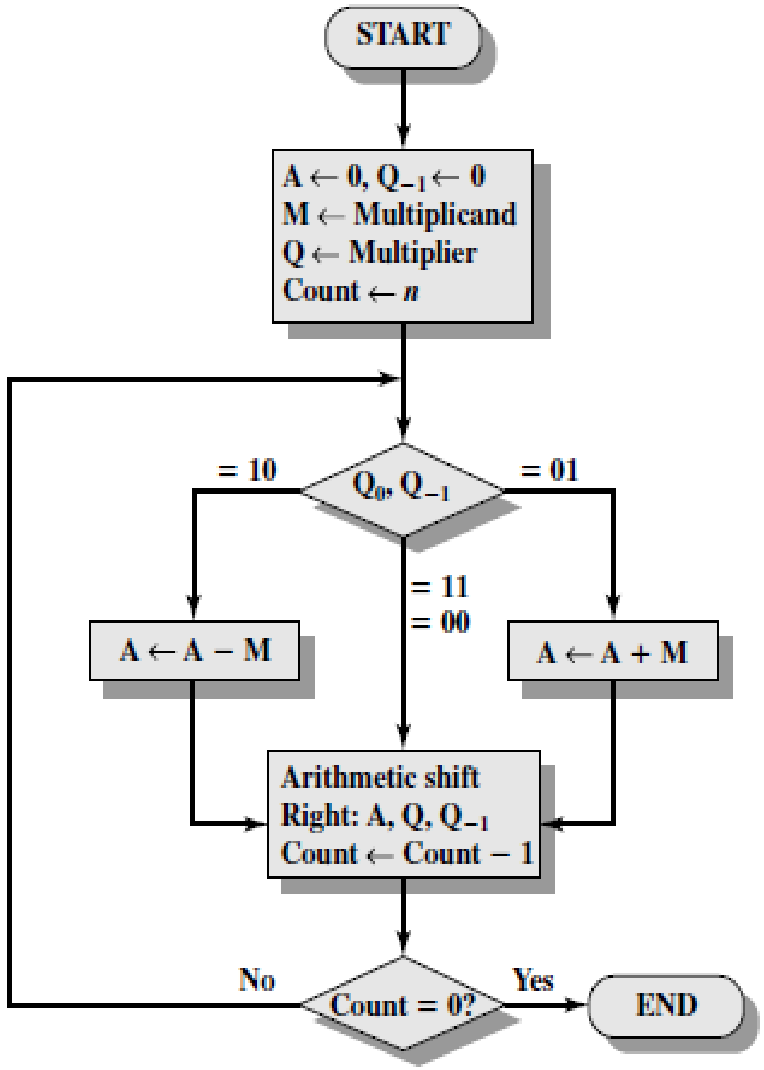


Figure: Booth's Algorithm
for Two's Complement Multiplication

Here,
A-M = **Signed (2's Complement) Subtraction**

Booth Multiplication Example

Table: Example of Booth Multiplication (Type 1)
Multiplicand, **M =** 1011 (4-bit)
And Multiplier, **Q =** 1101 (4-bit)
So, Product, **P = A,Q =** 00001111 (8-bit)

A	Q	Q_{-1}	M	Operation	Cycle
0000	110**1**	**0**	1011	Initial Value	
0101 0010	1101 111**0**	0 **1**	1011 1011	A = A–M Shift A,Q to Right	1
1101 1110	1110 111**1**	1 **0**	1011 1011	A = A+M Shift A,Q to Right	2
0011 0001	1111 111**1**	0 **1**	1011 1011	A = A-M Shift A,Q to Right	3
0001 0000	1111 1111	1 1	1011 1011	A = A Shift A,Q to Right	4

Type - 1

Booth Multiplication Example

Table: Example of Booth Multiplication (Type 1)
Multiplicand, **Y** = 1011 (4-bit)
And Multiplier, **X** = 1101 (4-bit)
So, Product, **P** = 00001111 (8-bit)

P	Y	X, X_{-1}	Operation	Cycle
0000 0000	1111 1011		Initial Value	
0000 0101 0000 0101	1111 1011 1111 0110	110**10**	P = P–Y Shift Y to Left	
1111 1011 1111 1011	1111 0110 1110 1100	11**01**0	P = P+Y Shift Y to Left	**1**
0000 1111 0000 1111	1110 1100 1101 1000	1**10**10	P = P-Y Shift Y to Left	
0000 1111 0000 1111	1101 1000 1011 0000	**11**010	P = P Shift Y to Left	

Type – 2

We are going to create a Booth Multiplier based on Type-2

Booth Multiplier Building Block (Cell B)

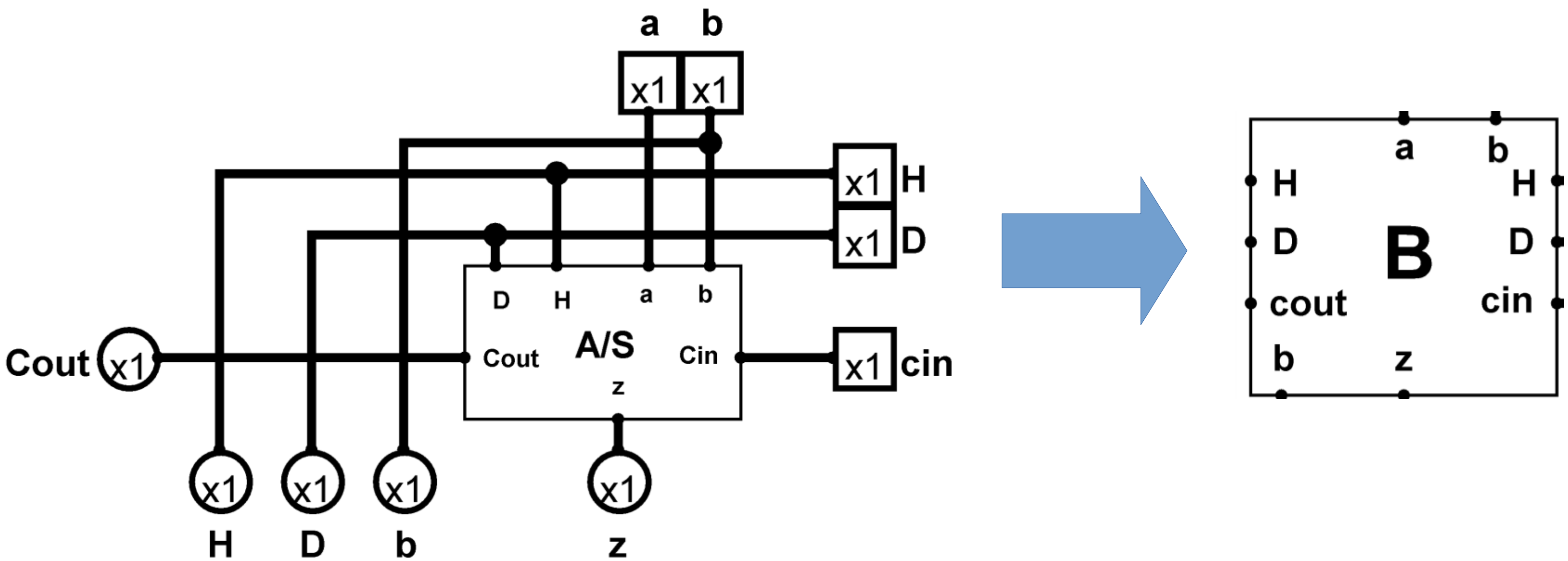


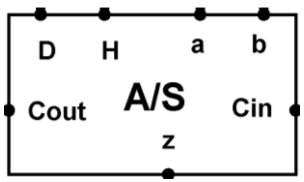
Figure: Cell B (Building Block)

Here,

1. A/S is combination of Full Adder and Full Subtractor and will switch in between them based on H and D value.
2. Value of H, D and b will propagate to next Block which means they are also outputs.

We are going to use a building block (Cell B) to create Booth multiplier

Booth Multiplier Building Block (Cell B)



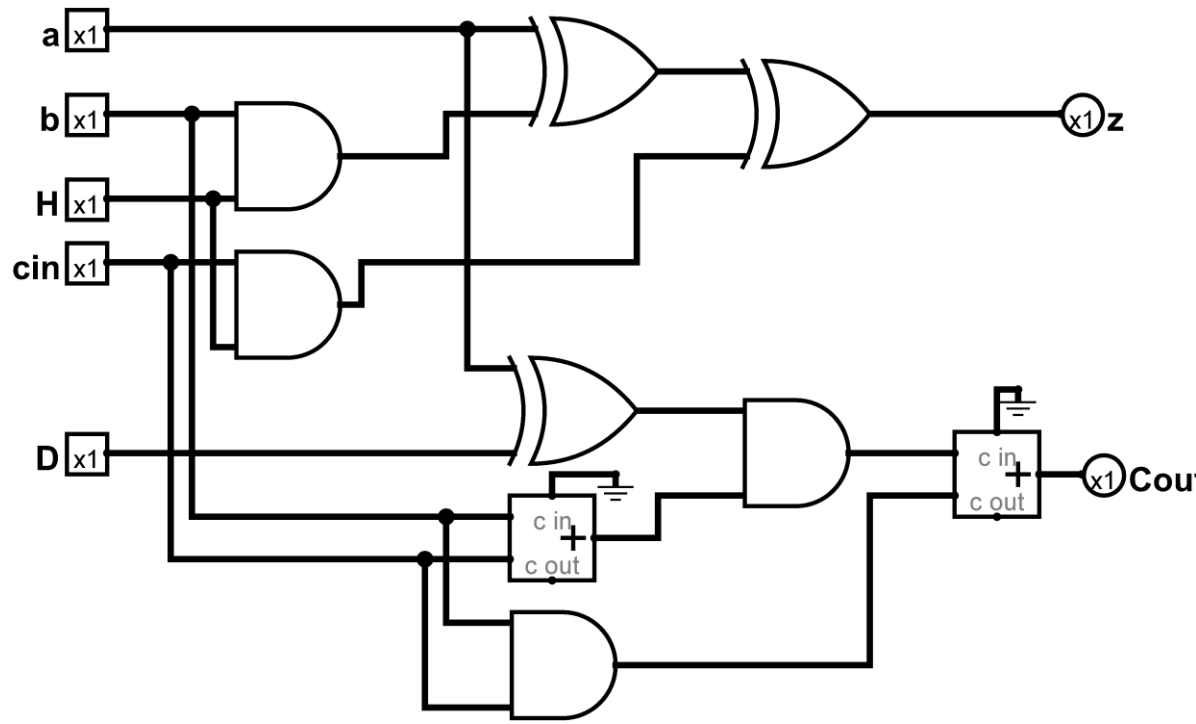


Figure: 1-bit Adder/Subtractor

$$z = a \text{ XOR } (b \text{ AND } H) \text{ XOR } (cin \text{ AND } H)$$
$$C_{out} = (a \text{ XOR } D) \text{ AND } (b \text{ OR } cin) \text{ OR } (b \text{ AND } cin)$$

Here,

1. **`H` and `D` are control inputs which will turn A/S into Full Adder or Full Subtractor.**
2. **Circuit will turn into Full Adder based on `H` and `D` value where `a`, `b` and `cin` (Carry in) are inputs, `cout` (Carry out) and `z` (Sum) are outputs.**
3. **Circuit will also turn into Full Subtractor based on `H` and `D` value where `a`, `b` and `cin` (Borrow in) are inputs, `cout` (Borrow out) and `z` (Difference) are outputs.**

Booth Multiplier Building Block (Cell B)

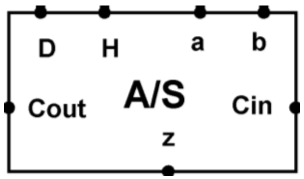


Figure: 1-bit Adder/Subtractor

$$z = a \text{ XOR } (b \text{ AND } H) \text{ XOR } (cin \text{ AND } H)$$
$$C_{out} = (a \text{ XOR } D) \text{ AND } (b \text{ OR } cin) \text{ OR } (b \text{ AND } cin)$$

Table: Function Table for 1-bit Adder/Subtractor

H	D	z and C_out	Function
0	0	z = a XOR 0 XOR 0 = a C_{out} = a AND (b OR cin) OR (b AND cin) = (a AND b) OR (a AND cin) OR (b AND cin)	$z = a$ (no operation)
0	1	z = a XOR 0 XOR 0 = a C_{out} = ~a AND (b OR cin) OR (b AND cin) = (~a AND b) OR (~a AND cin) OR (b AND cin)	$z = a$ (no operation)
1	0	z = a XOR b XOR cin C_{out} = a AND (b OR cin) OR (b AND cin) = (a AND b) OR (a AND cin) OR (b AND cin)	$C_{out}, z = a + b + c$ (add) [Full Adder]
1	1	z = a XOR b XOR cin C_{out} = ~a AND (b OR cin) OR (b AND cin) = (~a AND b) OR (~a AND cin) OR (b AND cin)	$C_{out}, z = a - b - c$ (subtract) [Full (Unsigned) Subtractor] (Not 2's Complement Subtraction)

Booth Multiplier Building Block (Cell C)

Table: Relationship between X_i , X_{i-1} and H, D

X_i	X_{i-1}	H	D	Operation
0	0	0	X	No Operation
0	1	1	0	Add
1	0	1	1	Subtract
1	1	0	X	No Operation

$$H = X_i \text{ XOR } X_{i-1}$$
$$D = X_i \text{ AND } (\sim X_{i-1})$$

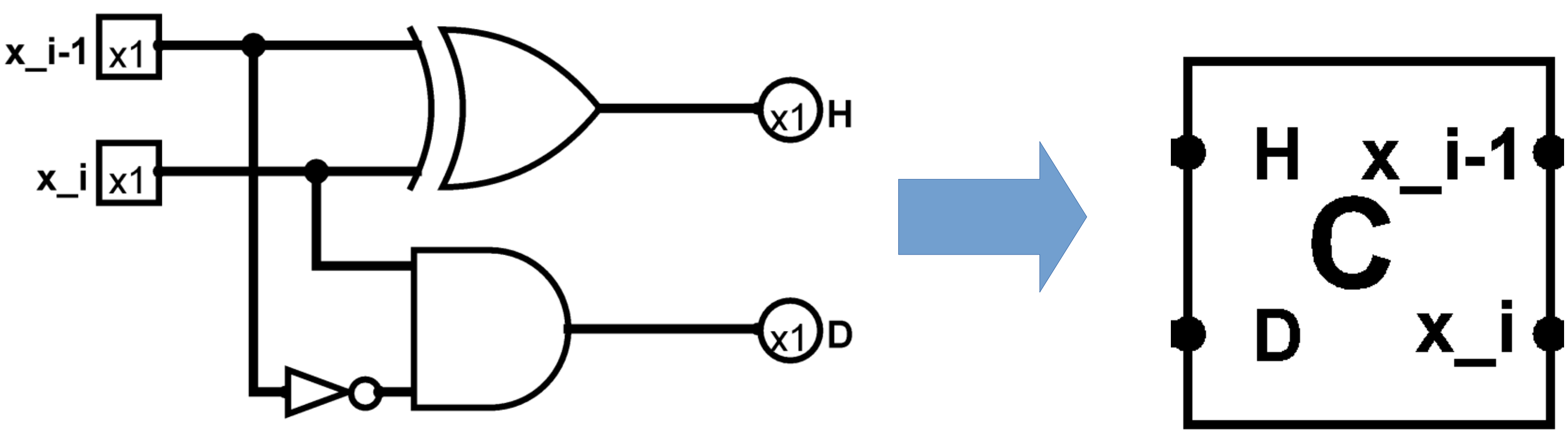


Figure: Cell C

Booth Multiplier

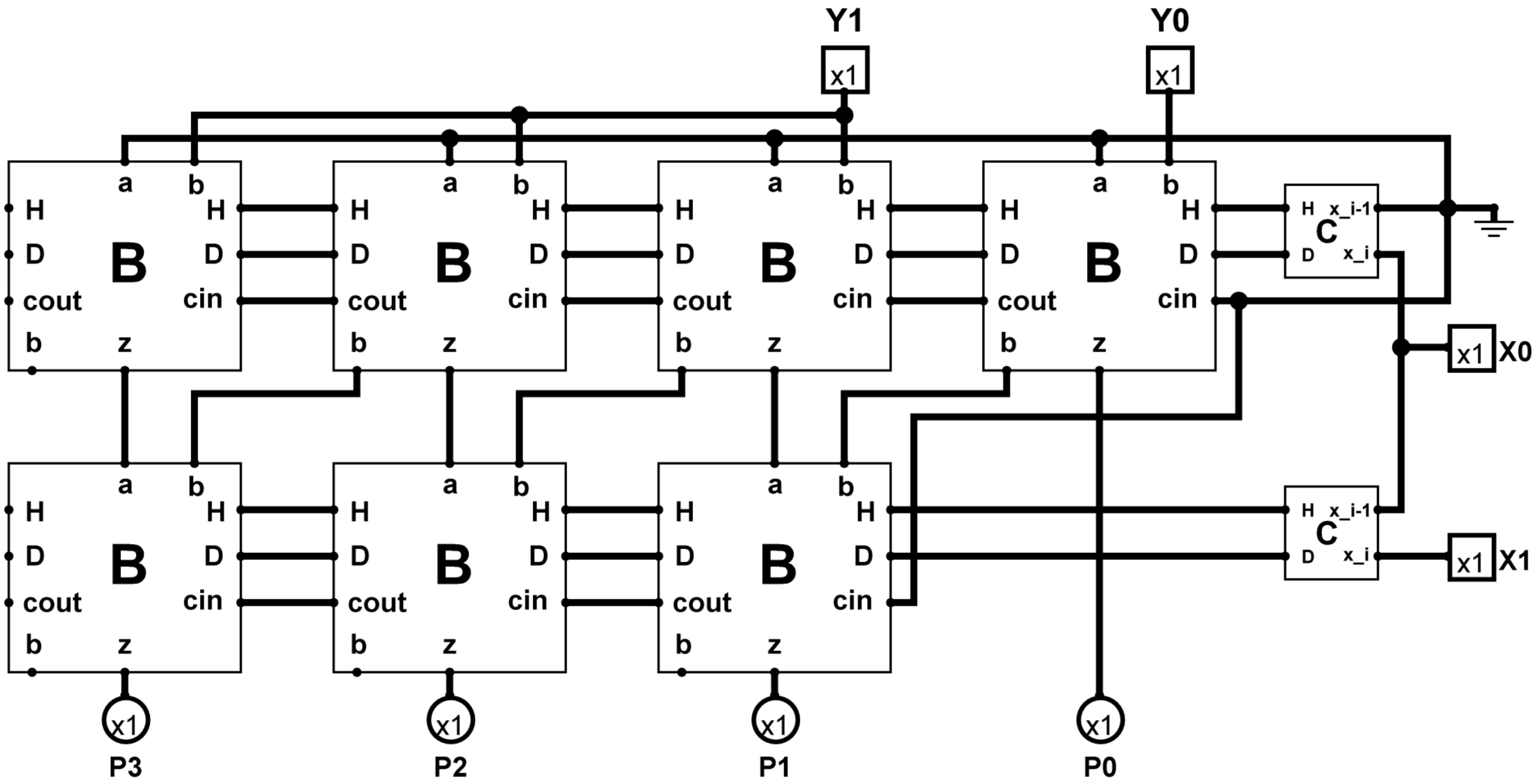


Figure: 2 *2 Booth Multiplier

Booth Multiplier Simulation

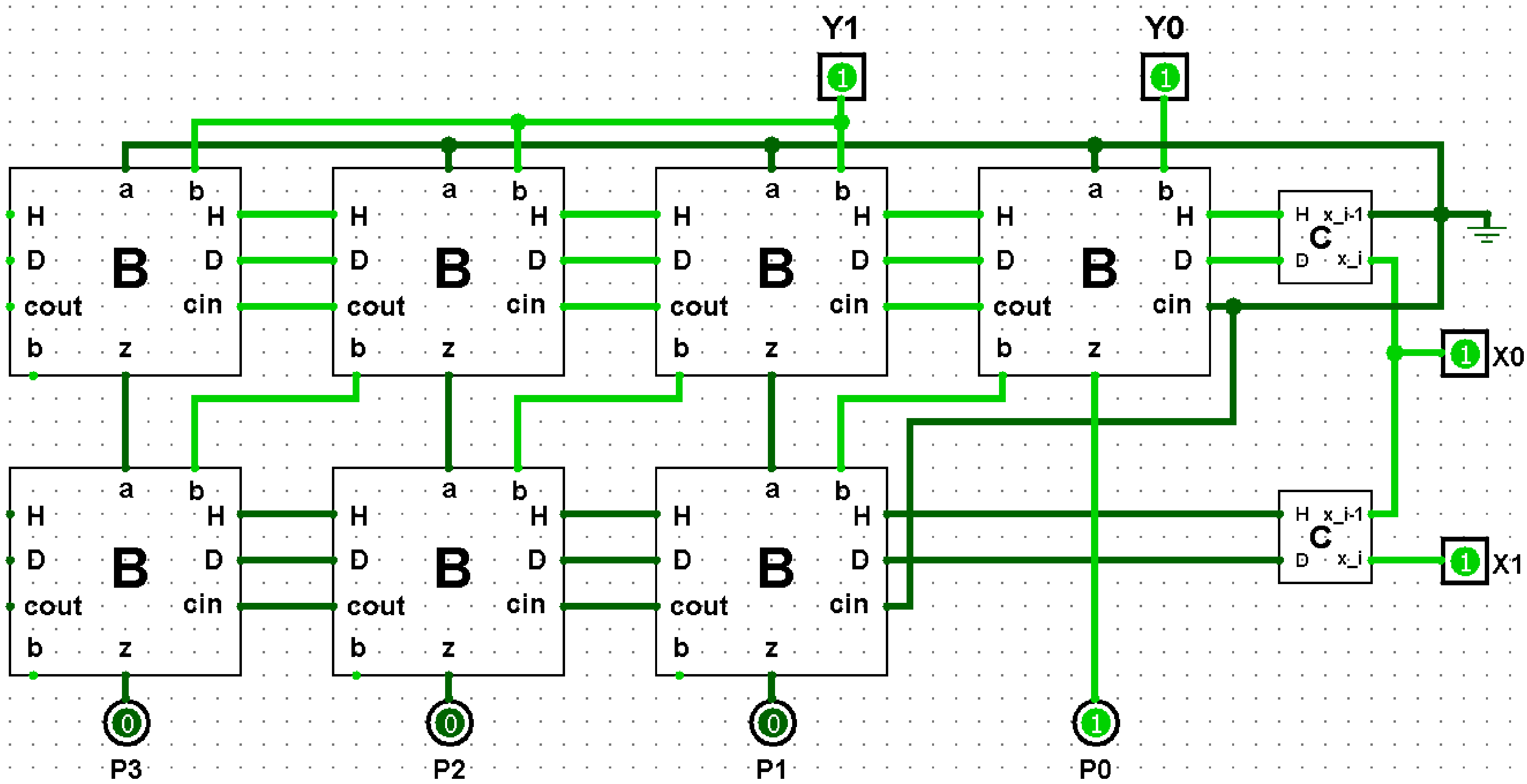


Figure: 2 *2 Booth Multiplier Simulation for input X = 11 (-1) and Y = 11 (-1). Output is P = 0001 (+1)

Booth Multiplier Simulation

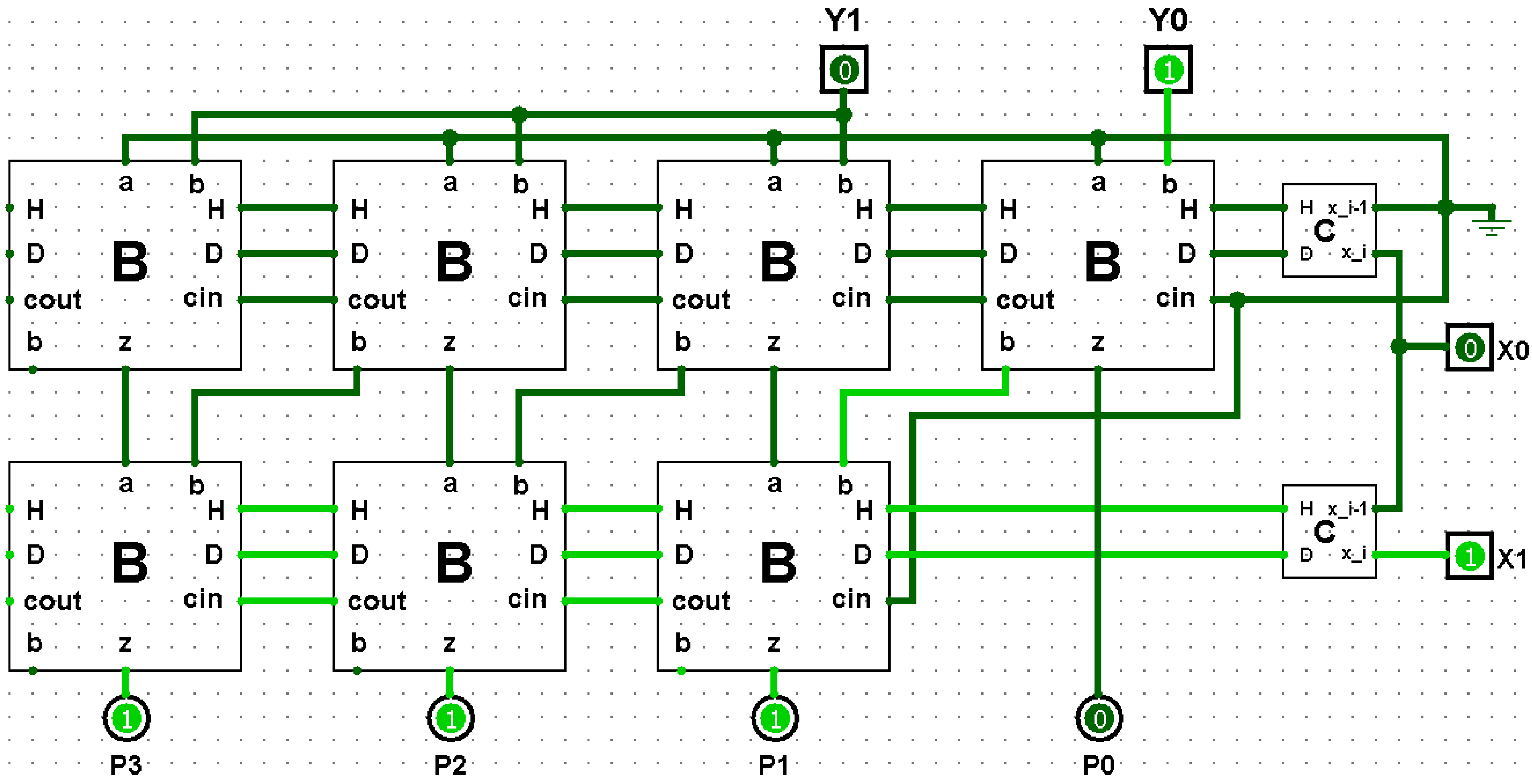


Figure: 2 *2 Booth Multiplier Simulation for input X = 10 (-2) and Y = 01 (+1). Output is P = 1110 (-2)

Booth Multiplier

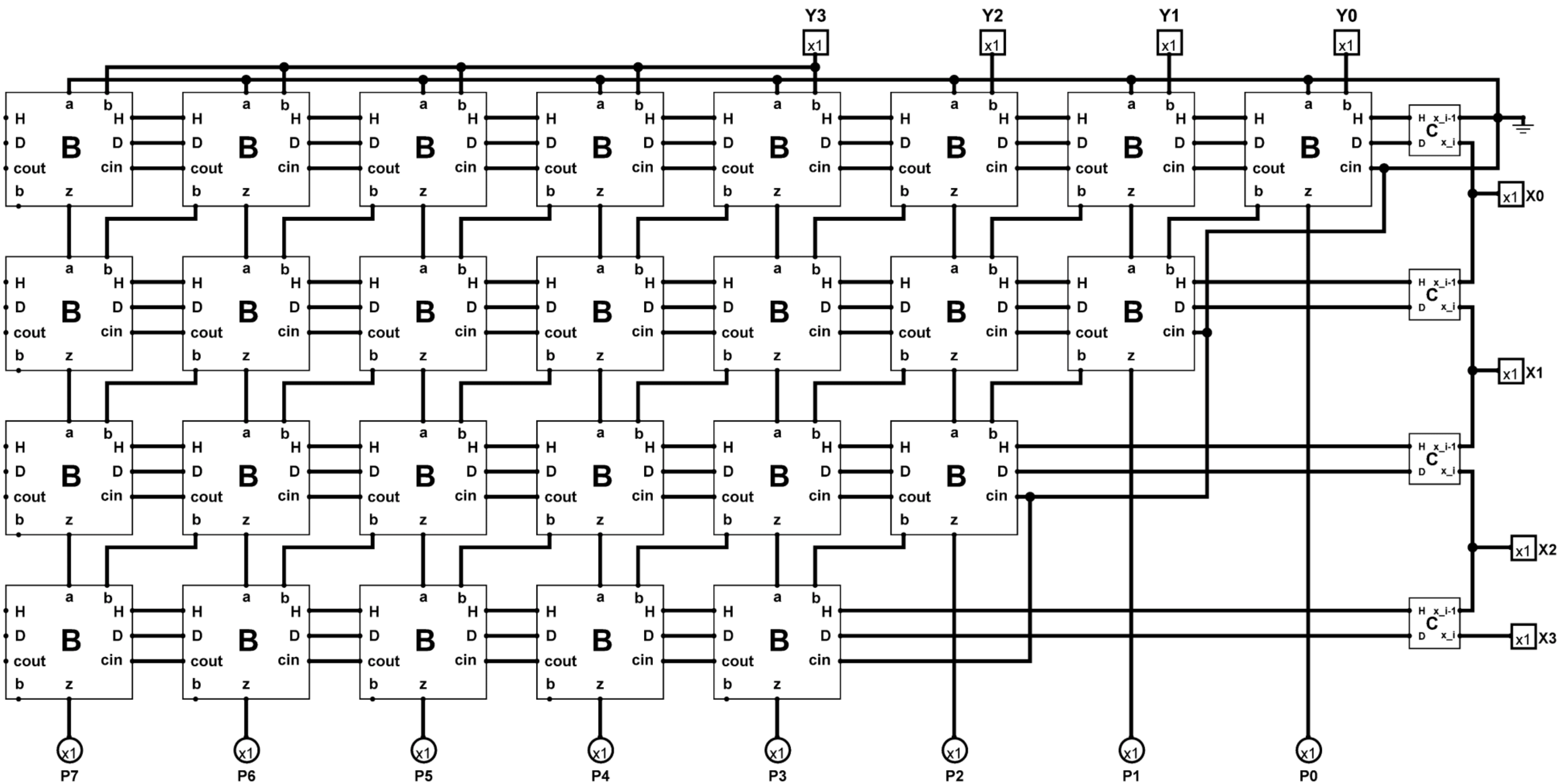


Figure: 4*4 Booth Multiplier

Booth Multiplier Simulation

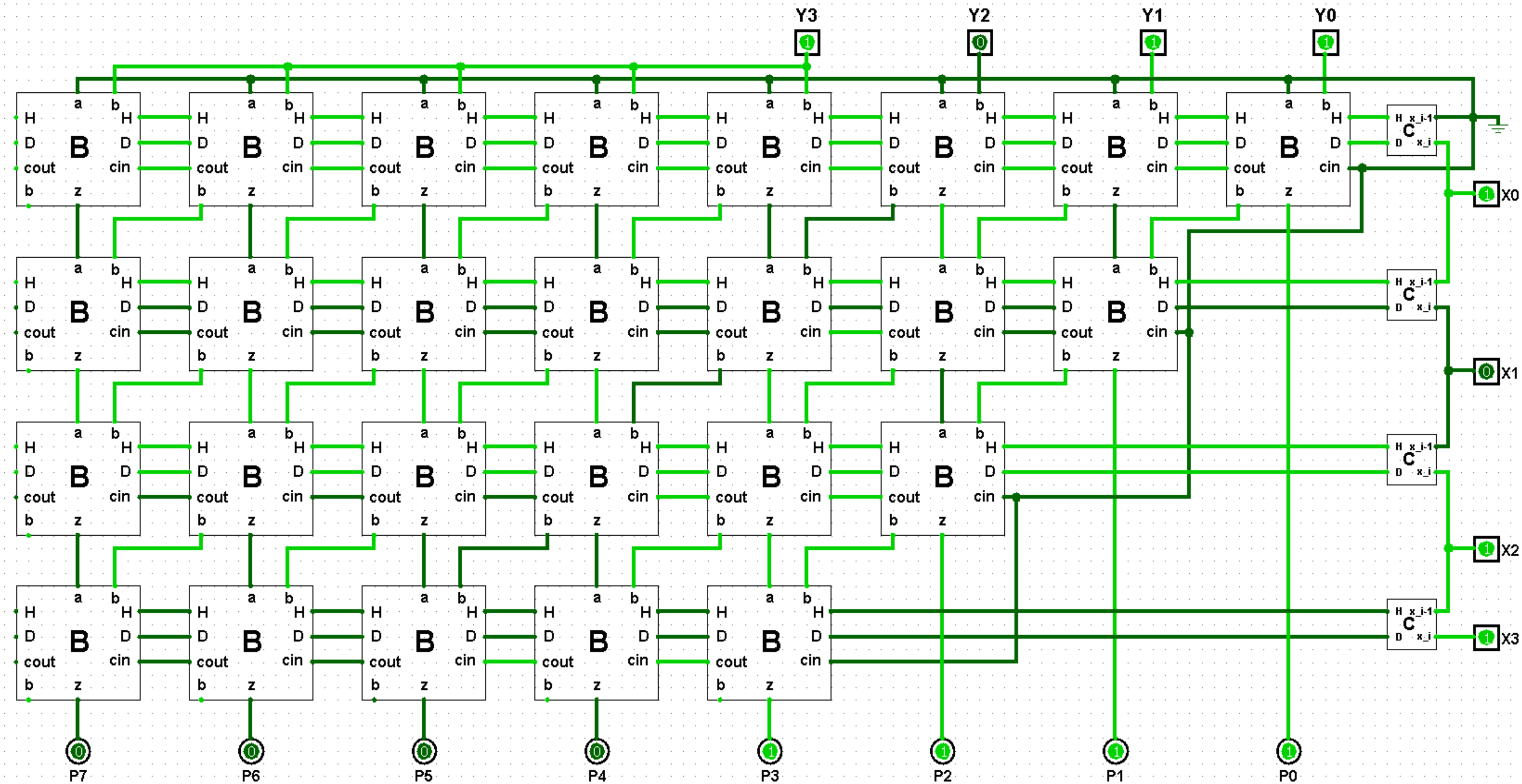


Figure: 2 *2 Booth Multiplier Simulation for input X = 1101 (-3) and Y = 1011 (-5). Output is P = 00001111 (+15)

Booth Multiplier Simulation

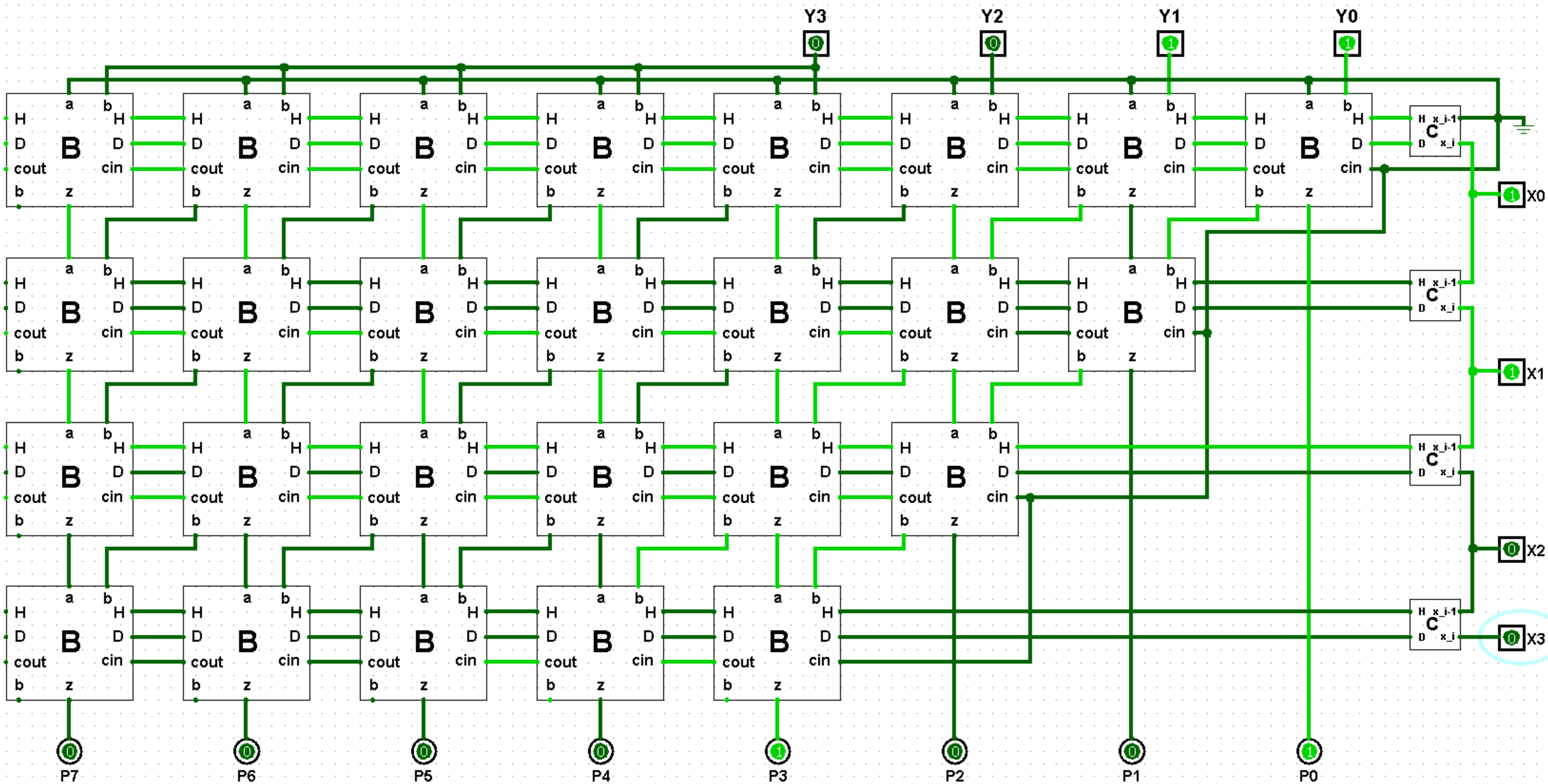


Figure: 2 *2 Booth Multiplier Simulation for input X = 0011 (+3) and Y = 0011 (+3). Output is P = 00001001 (+9)

Excercises

1)Calculate multiplication of two numbers using booth algorithm when

- X = 10 and Y = 11 **OR** X = 111 and Y = 100 **OR** X = 1001 and Y = 1111 OR
- X = 10 and Y = 1111 **OR** X = 1110 and Y = 1 OR
- X = -2 and Y = -3 **OR** X = +2 and Y = -3

2)Multiply two signed (2's complement) binary numbers 1001*1010 and design a circuit which can calculate this.

3)How does your computer do multiplication in program statement,

Z = -X * Y or Z=-2*-3 or Z = 1001 * 1010 (both are signed binary numbers).

Design a circuit and show how it calculates the result in each component.

4)Design a 2/3/4 bit signed/booth multiplier and show output of each circuit when

- X = 10 and Y = 11 **OR** X = 111 and Y = 100 **OR** X = 1001 and Y = 1111 OR
- X = 10 and Y = 1111 **OR** X = 1110 and Y = 1 OR
- X = -2 and Y = -3 **OR** X = +2 and Y = -3