

# Digital Logic III

Michael C. Hackett  
Assistant Professor, Computer Science

Community  
College  
*of* Philadelphia

# Lecture Topics

- Combinational Circuits
  - Adders
    - Half Adder
    - Full Adder
  - Subtractors
    - Half Subtractor
    - Full Subtractor
  - Multipliers

# Adders

- **Adders** are combinational logic circuits capable of performing addition
- A **half adder** has two inputs (the two digits to add) and two outputs (the sum and the carry).

0	1	0	1 $\leftarrow X_0$
<u>+ 0</u>	<u>+ 0</u>	<u>+ 1</u>	<u>+ 1</u> $\leftarrow X_1$
00	01	01	10
		$C$ ( <i>Carry</i> )	$S$ ( <i>Sum</i> )

# Half Adders

- Half adder truth table:

$X_1$	$X_0$	$C$	$S$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$\begin{array}{r} 0 \\ + 0 \\ \hline 00 \end{array} \quad \begin{array}{r} 1 \\ + 0 \\ \hline 01 \end{array} \quad \begin{array}{r} 0 \\ + 1 \\ \hline 01 \end{array} \quad \begin{array}{r} 1 \\ + 1 \\ \hline 10 \end{array}$$

$C$  (*Carry*)       $S$  (*Sum*)

Diagram illustrating the half adder operation for the input pair (1, 1). The inputs are  $X_1 = 1$  and  $X_0 = 1$ . The sum is 10, where the leftmost digit (1) is the carry ( $C$ ) and the rightmost digit (0) is the sum ( $S$ ).

# Half Adders

SOP Expressions:

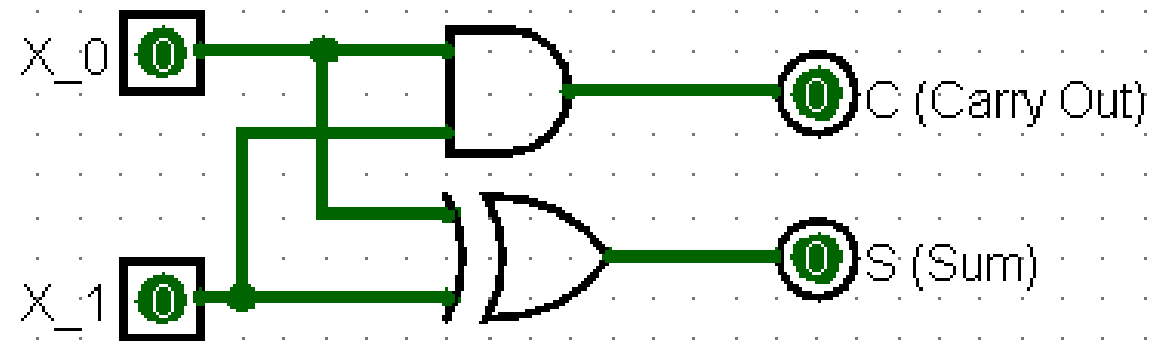
$$C = X_1X_0$$

$$S = \overline{X_1}X_0 + X_1\overline{X_0} = X_1 \oplus X_0$$

$X_1$	$X_0$	$C$	$S$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

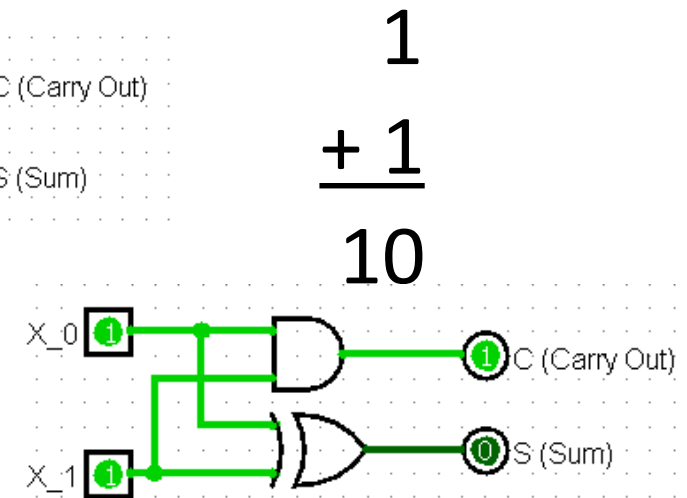
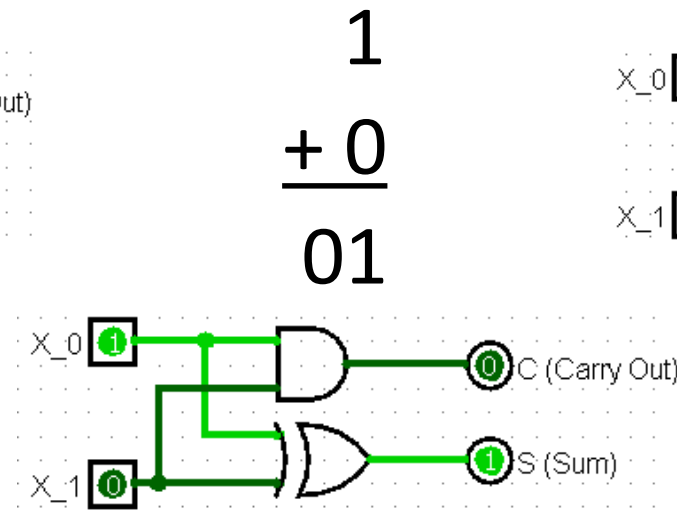
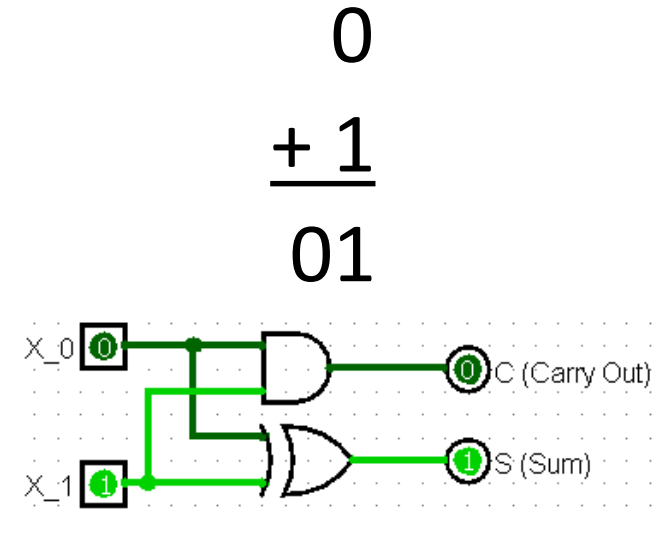
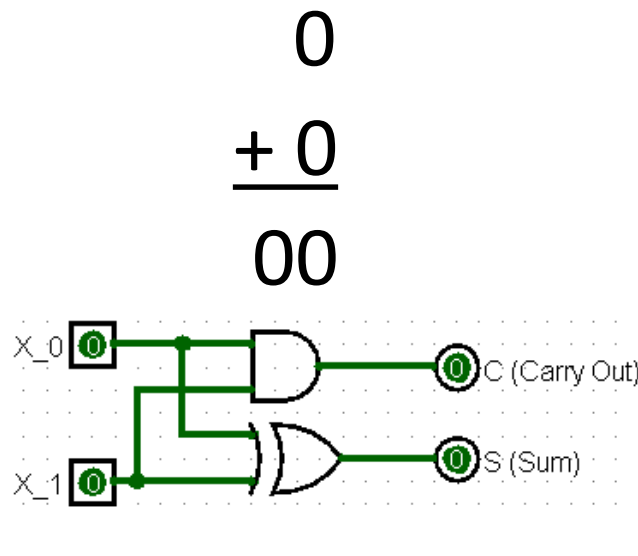
# Half Adders

Half Adder Logic Circuit:



# Half Adders

Half Adder Logic Circuit:



# Full Adders

- A **full adder** has three inputs (the two digits to add, plus a value *carried in*) and two outputs (the sum and the carry).

The diagram shows a truth table for a full adder. The inputs are labeled as  $C_{IN}$  (Carry In),  $X_1$ , and  $X_0$ . The outputs are labeled as  $C_{OUT}$  (Carry Out) and  $S$  (Sum). Blue arrows point from the labels to the corresponding values in the equations. The equations are arranged in a column, showing the sum of the three inputs for each combination of  $X_1$  and  $X_0$ .

$C_{IN}$	$X_1$	$X_0$	$C_{OUT}$	$S$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



# Full Adders

- Full adder truth table:

$C_{IN}$	$X_1$	$X_0$	$C_{OUT}$	$S$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

# Full Adders

SOP Expressions:

$$C_{OUT} = \overline{C_{IN}}X_1X_0 + C_{IN}\overline{X_1}X_0 + C_{IN}X_1\overline{X_0} + C_{IN}X_1X_0$$

$$S = \overline{C_{IN}}\overline{X_1}X_0 + \overline{C_{IN}}X_1\overline{X_0} + C_{IN}\overline{X_1}\overline{X_0} + C_{IN}X_1X_0$$

$C_{IN}$	$X_1$	$X_0$	$C_{OUT}$	$S$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

# Full Adders

Simplifying:

$$C_{OUT} = \overline{C_{IN}}X_1X_0 + C_{IN}\overline{X_1}X_0 + C_{IN}X_1\overline{X_0} + \mathbf{C_{IN}X_1X_0}$$

$$C_{OUT} = \mathbf{X_1X_0}(\overline{C_{IN}} + \mathbf{C_{IN}}) + C_{IN}\overline{X_1}X_0 + C_{IN}X_1\overline{X_0}$$

$$C_{OUT} = \mathbf{X_1X_0(1)} + C_{IN}\overline{X_1}X_0 + C_{IN}X_1\overline{X_0}$$

$$C_{OUT} = X_1X_0 + \mathbf{C_{IN}\overline{X_1}X_0} + \mathbf{C_{IN}X_1\overline{X_0}}$$

$$C_{OUT} = X_1X_0 + C_{IN}(\overline{X_1}X_0 + X_1\overline{X_0})$$

$$C_{OUT} = X_1X_0 + C_{IN}(X_0 \oplus X_1)$$

# Full Adders

Simplifying:

$$S = \underbrace{\overline{C_{IN}} \overline{X_1} X_0 + \overline{C_{IN}} X_1 \overline{X_0}}_{\text{XOR}} + \underbrace{C_{IN} \overline{X_1} \overline{X_0} + C_{IN} X_1 X_0}_{\text{AND}}$$

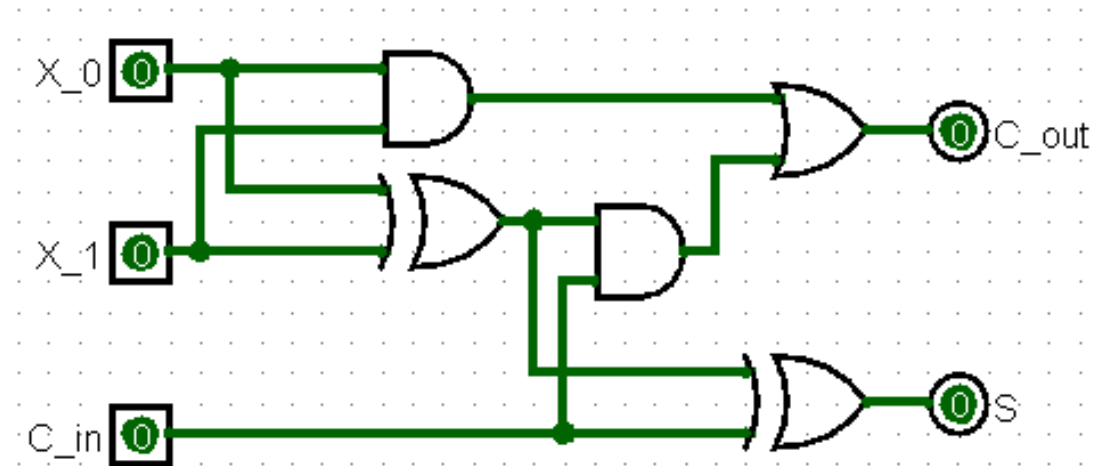
$$S = \overline{C_{IN}} (X_0 \oplus X_1) + C_{IN} (X_0 \odot X_1)$$

$$S = \underbrace{\overline{C_{IN}} (X_0 \oplus X_1) + C_{IN} (\overline{X_0 \oplus X_1})}_{\text{XOR}} \quad \bar{X}Y + X\bar{Y} = X \oplus Y$$

$$S = C_{IN} \oplus (X_0 \oplus X_1) = C_{IN} \oplus X_0 \oplus X_1$$

# Full Adders

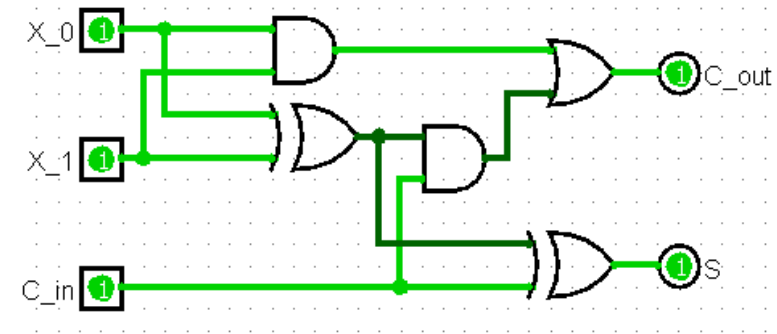
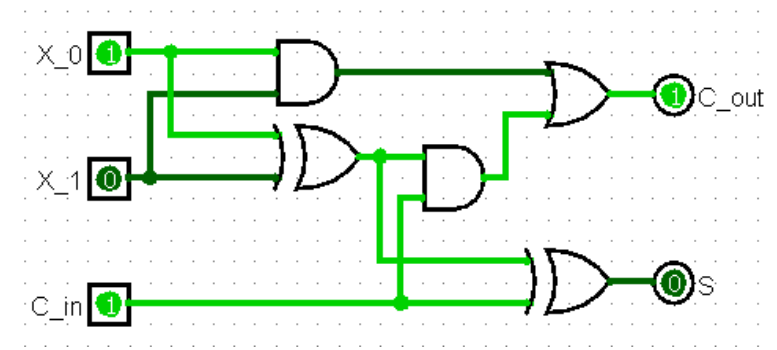
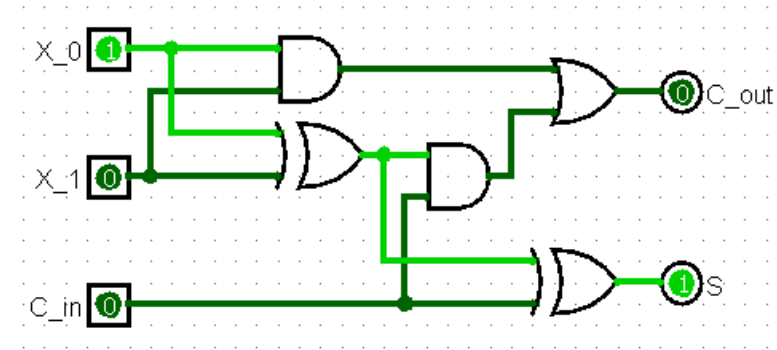
Full Adder Logic Circuit:



# Full Adders

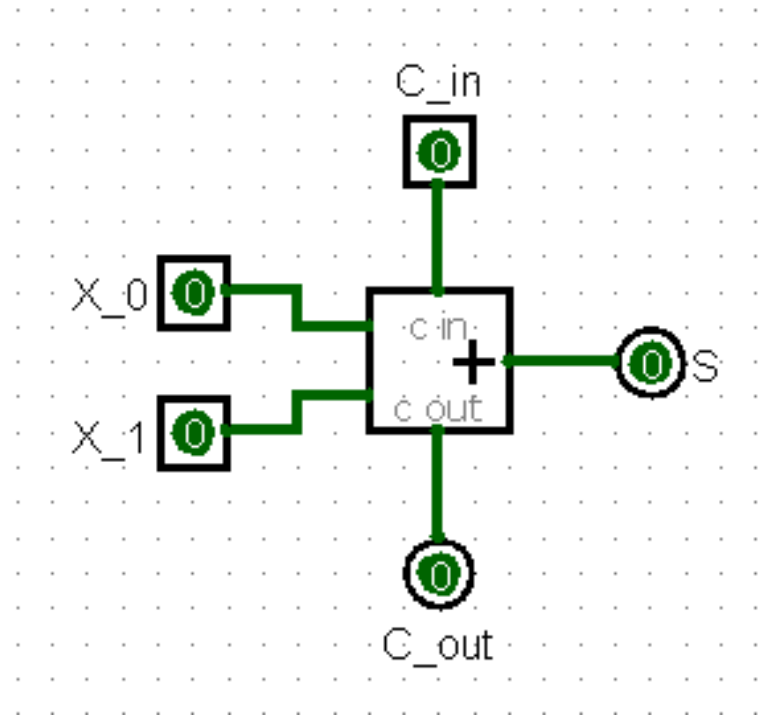
Full Adder Logic Circuit:

$C_{IN}$	$X_1$	$X_0$	$C_{OUT}$	$S$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



# Full Adders

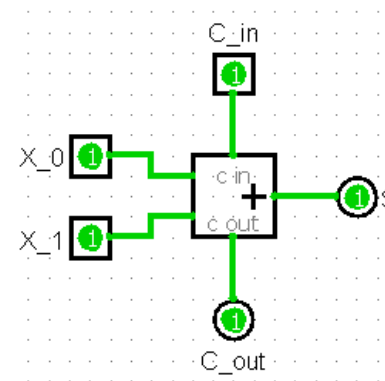
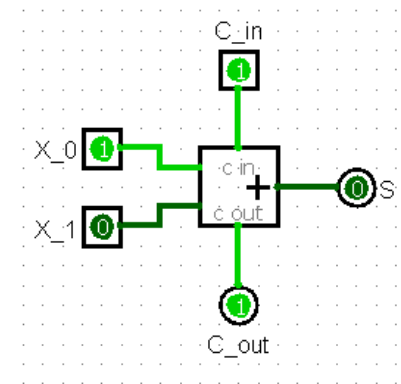
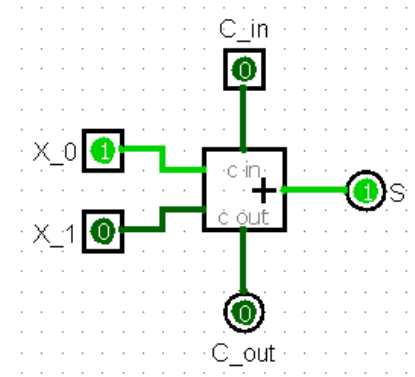
- Abstracted Full Adder:



# Full Adders

- Abstracted Full Adder:

$C_{IN}$	$X_1$	$X_0$	$C_{OUT}$	$S$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1





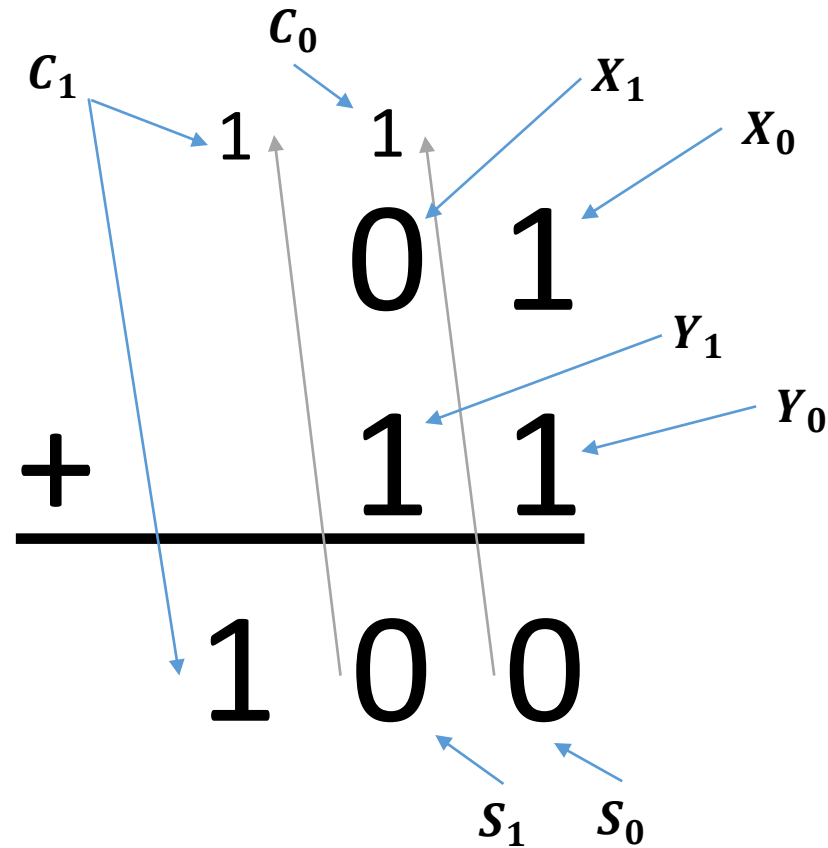
# Full Adders

- So far, we've seen only how adders can add single digits together
  - $1+1+1$  or  $0+1$  or  $1+0+1$  is no problem
  - What if we wanted to add  $10+11$  or  $11101+10101$ ?
- Full adders can work together by providing the carry out of one full adder as the carry in for a second adder
  - The technique shown next is a *ripple-carry adder*

$$\begin{array}{r} 001 \\ + 011 \\ \hline 100 \end{array}$$

# Full Adders

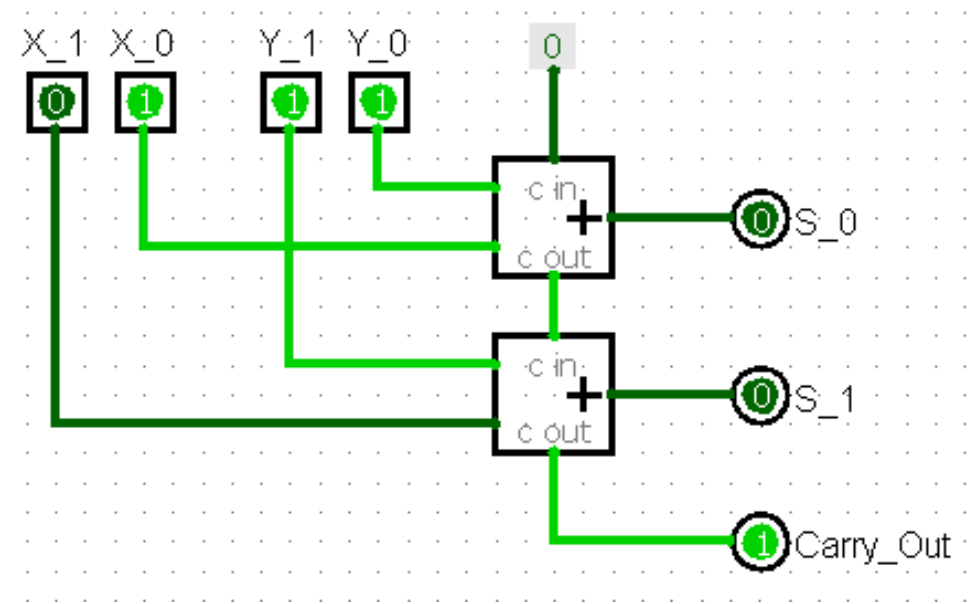
- For example:



# Full Adders

- A 2-bit Adder:

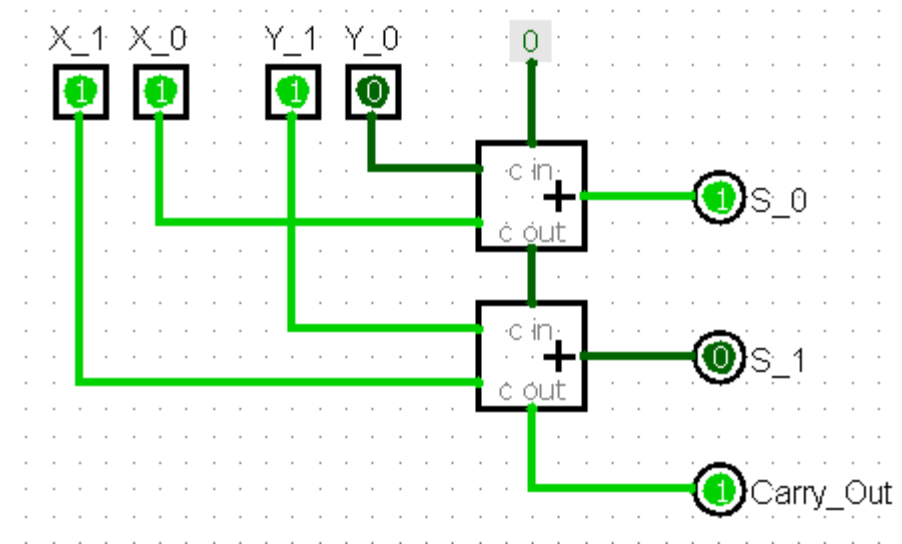
$$\begin{array}{r} \phantom{+} 01 \\ + 11 \\ \hline 100 \end{array}$$



# Full Adders

- Another example:

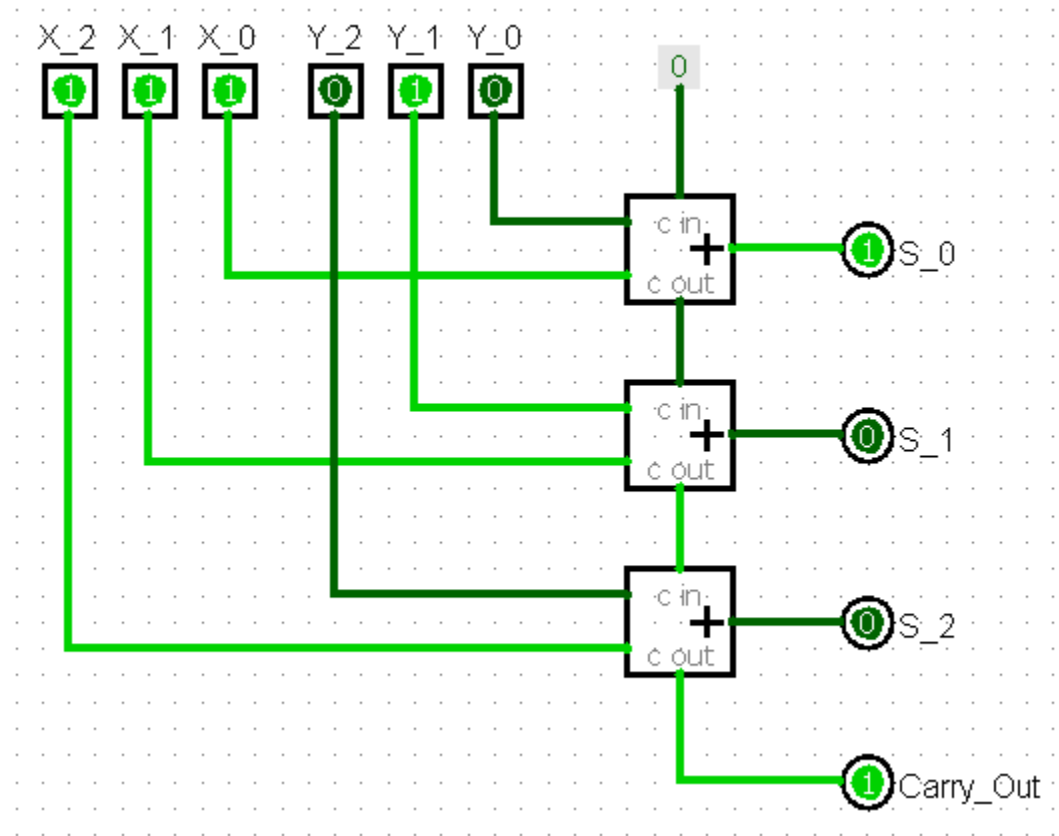
$$\begin{array}{r} \phantom{+} \phantom{+} \phantom{+} 1 \phantom{+} 1 \\ + \phantom{+} 1 \phantom{+} 0 \\ \hline 1 \phantom{+} 0 \phantom{+} 1 \end{array}$$



# Full Adders

- A 3-bit Adder:

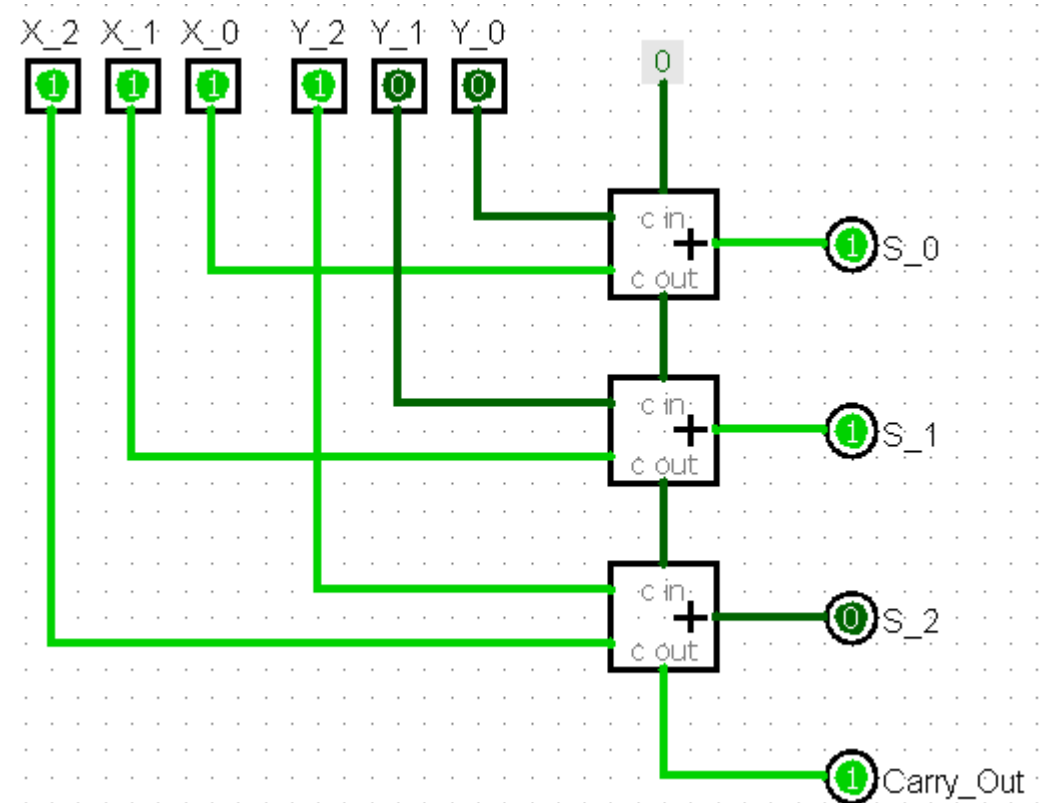
$$\begin{array}{r} \phantom{+} 111 \\ + 010 \\ \hline 1001 \end{array}$$



# Full Adders

- Another example:

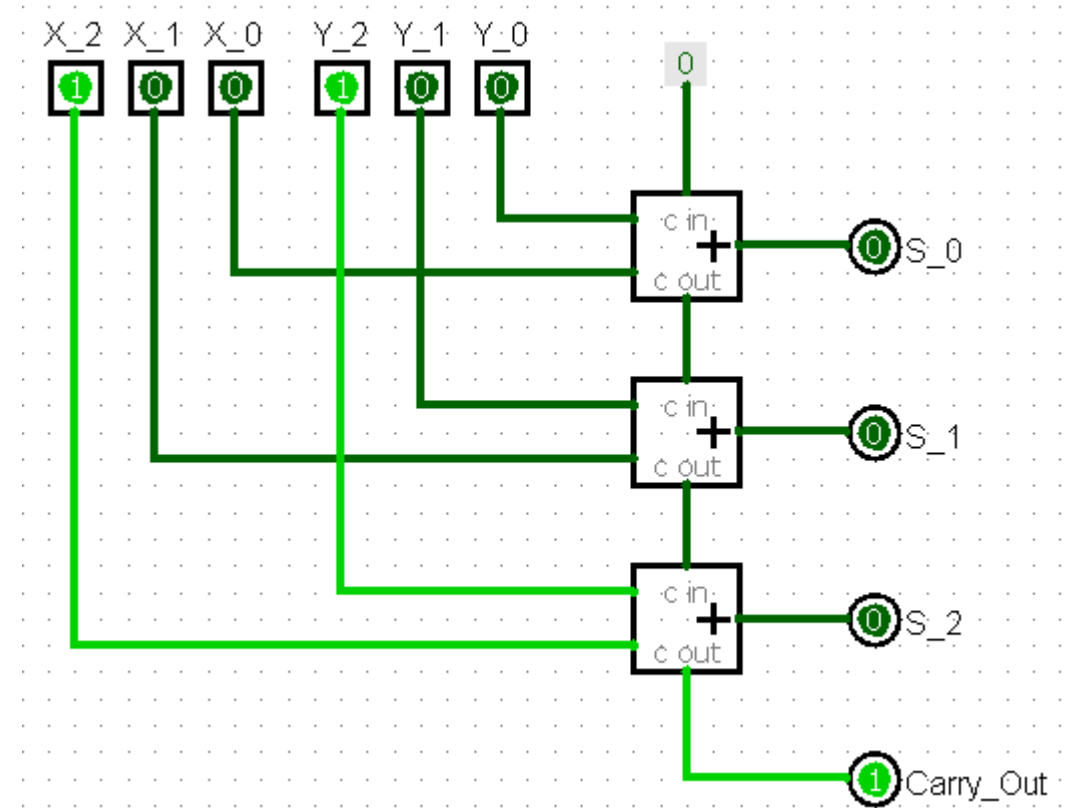
$$\begin{array}{r} \phantom{+} \phantom{+} \phantom{+} \phantom{+} \\ \phantom{+} \phantom{+} \phantom{+} \phantom{+} \\ + \phantom{+} \phantom{+} \phantom{+} \phantom{+} \\ \hline 1 \phantom{+} \phantom{+} \phantom{+} \phantom{+} \end{array}$$



# Full Adders

- Another example:

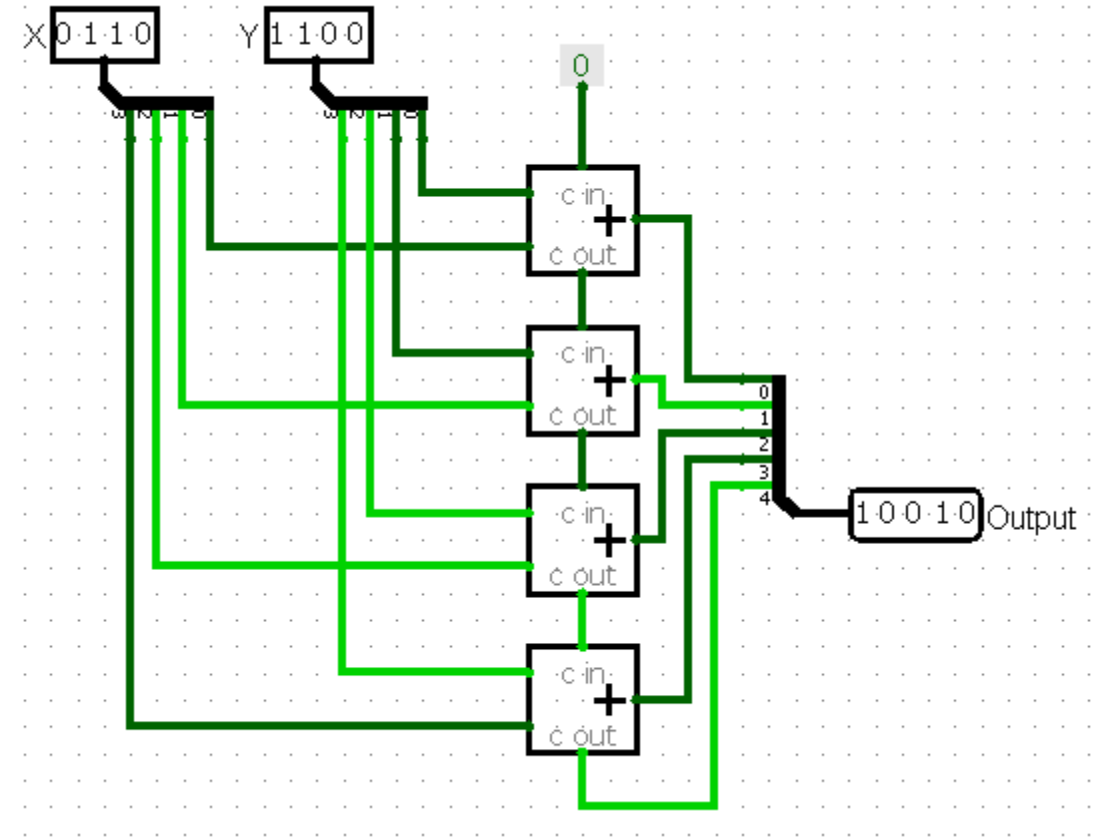
$$\begin{array}{r} \phantom{+} 100 \\ + 100 \\ \hline 1000 \end{array}$$



# Full Adders

- A 4-bit Adder (with buses):

$$\begin{array}{r} 0110 \\ + 1100 \\ \hline 10010 \end{array}$$

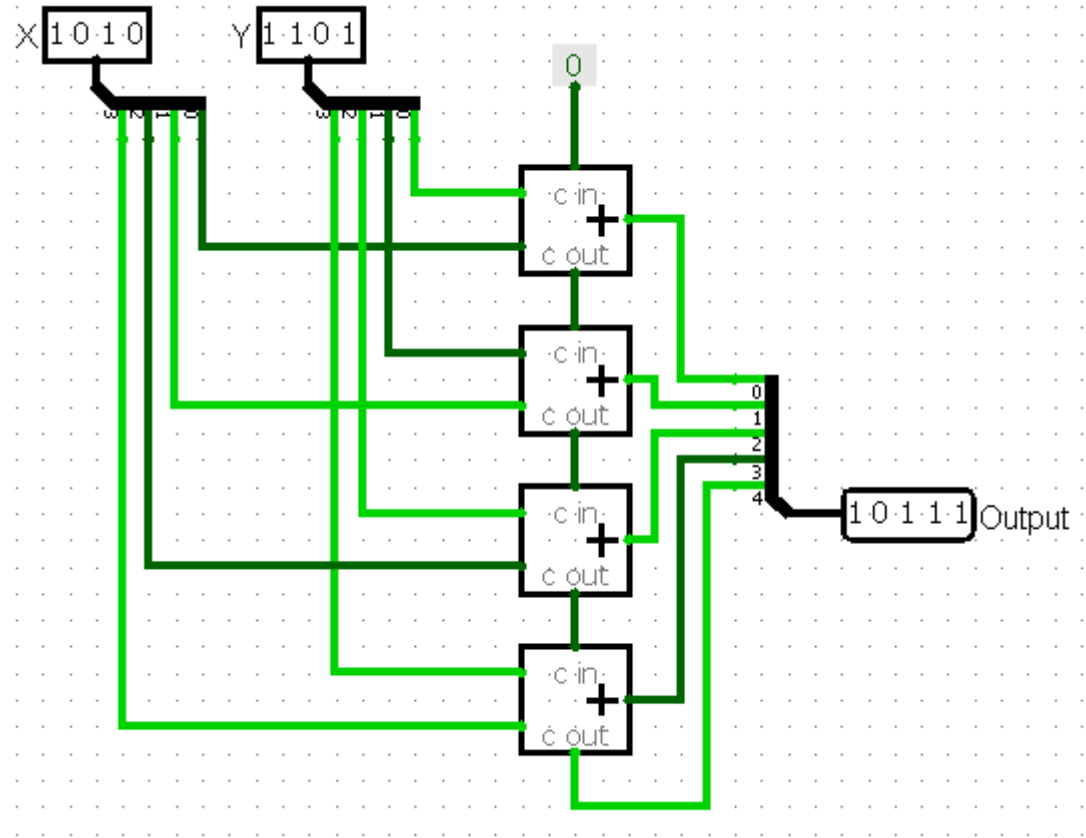




# Full Adders

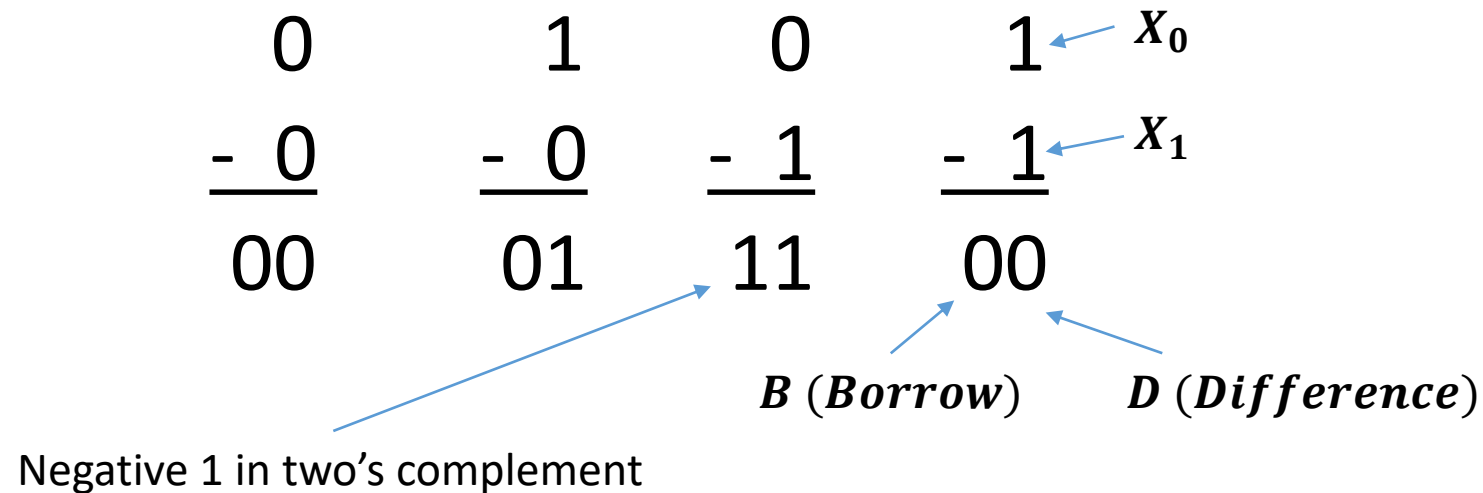
- Another example

$$\begin{array}{r} 1010 \\ + 1101 \\ \hline 10111 \end{array}$$



# Subtractors

- **Subtractors** are combinational logic circuits capable of performing subtraction
- A **half subtractor** has two inputs (the two digits to subtract) and two outputs (the difference and the borrow).



# Half Subtractor

- Half subtractor truth table:

$X_1$	$X_0$	$B$	$D$
0	0	0	0
0	1	0	1
1	0	1	1
1	1	0	0

$$\begin{array}{r} 0 \\ - 0 \\ \hline 00 \end{array} \quad \begin{array}{r} 1 \\ - 0 \\ \hline 01 \end{array} \quad \begin{array}{r} 0 \\ - 1 \\ \hline 11 \end{array} \quad \begin{array}{r} 1 \\ - 1 \\ \hline 00 \end{array}$$

$B$  (*Borrow*)       $D$  (*Difference*)

Diagram illustrating the half subtractor operation for the case  $X_1=1, X_0=1$ . The inputs are  $X_1$  and  $X_0$ . The output is the Borrow ( $B$ ) and the Difference ( $D$ ). The diagram shows the subtraction of  $1$  from  $1$  at the  $X_0$  position, resulting in a Borrow of  $1$  and a Difference of  $0$ .

# Half Subtractors

SOP Expressions:

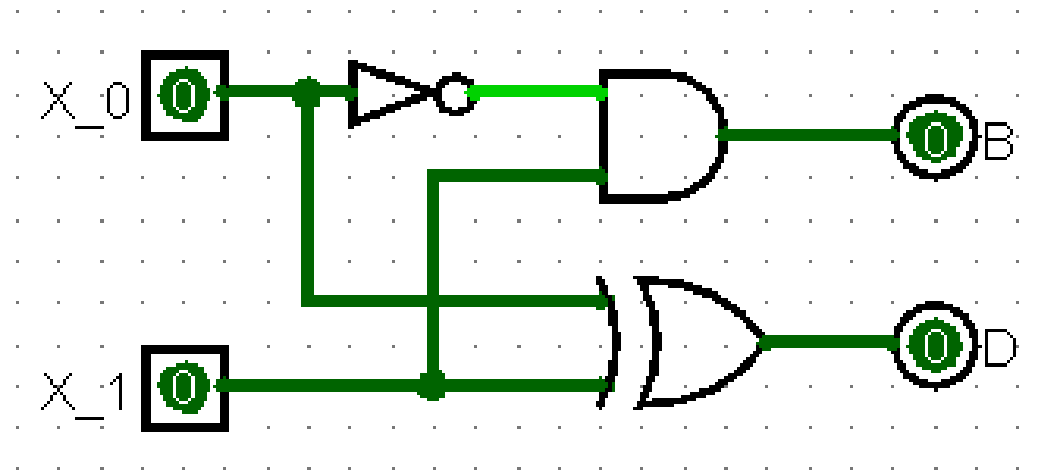
$$B = X_1 \overline{X_0}$$

$$D = \overline{X_1} X_0 + X_1 \overline{X_0} = X_1 \oplus X_0$$

$X_1$	$X_0$	$B$	$D$
0	0	0	0
0	1	0	1
1	0	1	1
1	1	0	0

# Half Subtractor

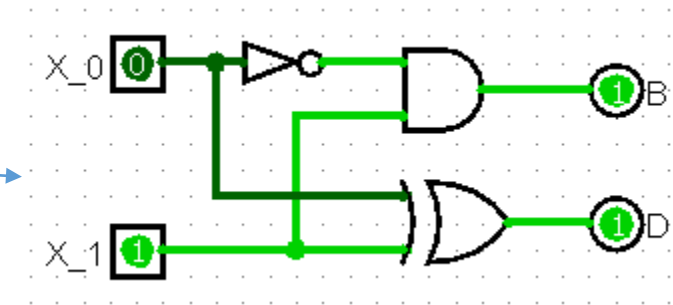
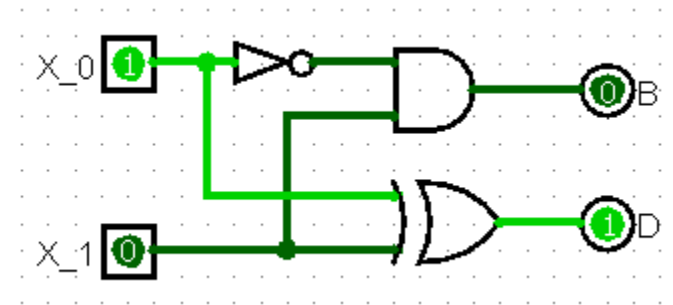
Half Subtractor Logic Circuit:



# Half Subtractor

Half Subtractor Logic Circuit:

$X_1$	$X_0$	$B$	$D$
0	0	0	0
0	1	0	1
1	0	1	1
1	1	0	0



# Full Subtractors

- A **full subtractor** has three inputs (the two digits to subtract, plus a value *borrowed in*) and two outputs (the difference and the borrow).

The diagram shows a truth table for a full subtractor. The inputs are labeled  $B_{IN}$  (Borrow In),  $X_1$ , and  $X_0$ . The outputs are labeled  $B_{OUT}$  (Borrow Out) and  $D$  (Difference). Blue arrows point from the labels to the corresponding columns in the table. The table lists all possible combinations of the three inputs and the resulting difference and borrow-out values.

$B_{IN}$	$X_1$	$X_0$	$B_{OUT}$	$D$
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

# Full Subtractors

SOP Expressions:

$$B_{OUT} = \overline{B_{IN}} \overline{X_1} X_0 + \overline{B_{IN}} X_1 \overline{X_0} + \overline{B_{IN}} X_1 X_0 + B_{IN} X_1 X_0$$

$$D = \overline{B_{IN}} \overline{X_1} X_0 + \overline{B_{IN}} X_1 \overline{X_0} + B_{IN} \overline{X_1} \overline{X_0} + B_{IN} X_1 X_0$$

$B_{IN}$	$X_1$	$X_0$	$B_{OUT}$	$D$
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



# Full Subtractors

Simplifying:

$$B_{OUT} = \overline{B_{IN}} \overline{X_1} X_0 + \overline{B_{IN}} X_1 \overline{X_0} + \overline{B_{IN}} X_1 X_0 + B_{IN} X_1 X_0$$

$$B_{OUT} = \overline{B_{IN}} X_1 + \overline{B_{IN}} X_0 + X_1 X_0$$

		$X_0$	
		0	1
$B_{IN} \ X_1$	00	0 000	1 001
	01	1 010	1 011
	11	0 110	1 111
	10	0 100	0 101

# Full Subtractors

Simplifying:

$$D = \underbrace{\overline{B_{IN}} \overline{X_1} X_0 + \overline{B_{IN}} X_1 \overline{X_0}}_{\text{XOR}} + \underbrace{B_{IN} \overline{X_1} \overline{X_0} + B_{IN} X_1 X_0}_{\text{XOR}}$$

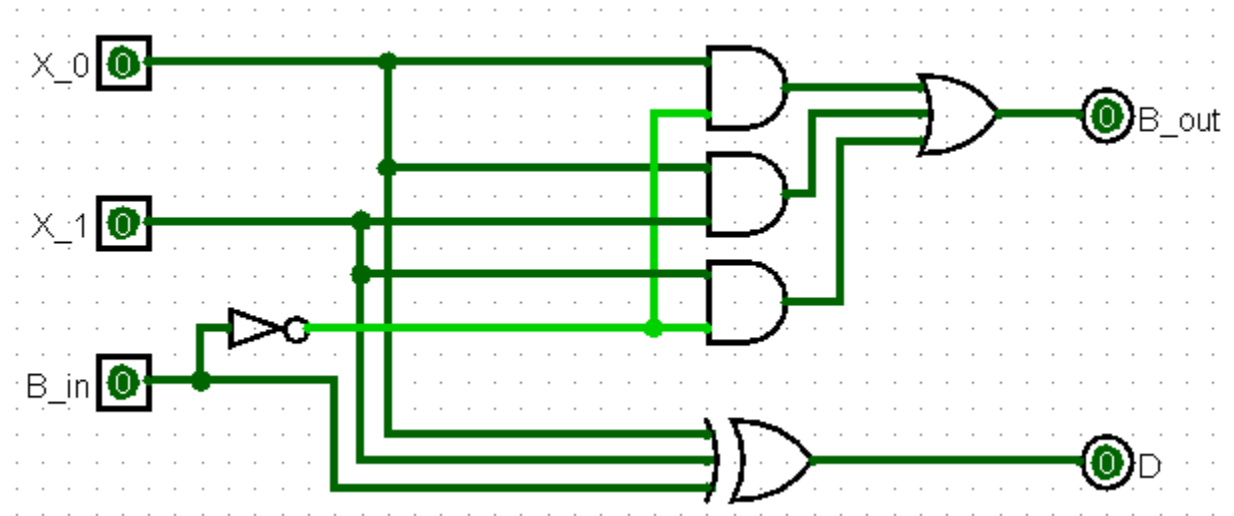
$$D = \overline{B_{IN}}(X_0 \oplus X_1) + B_{IN}(\underbrace{X_0 \odot X_1}_{\text{AND}})$$

$$D = \underbrace{\overline{B_{IN}}(X_0 \oplus X_1) + B_{IN}(\overline{X_0 \oplus X_1})}_{\text{XOR}} \quad \bar{X}Y + X\bar{Y} = X \oplus Y$$

$$D = B_{IN} \oplus (X_0 \oplus X_1) = B_{IN} \oplus X_0 \oplus X_1$$

# Full Subtractor

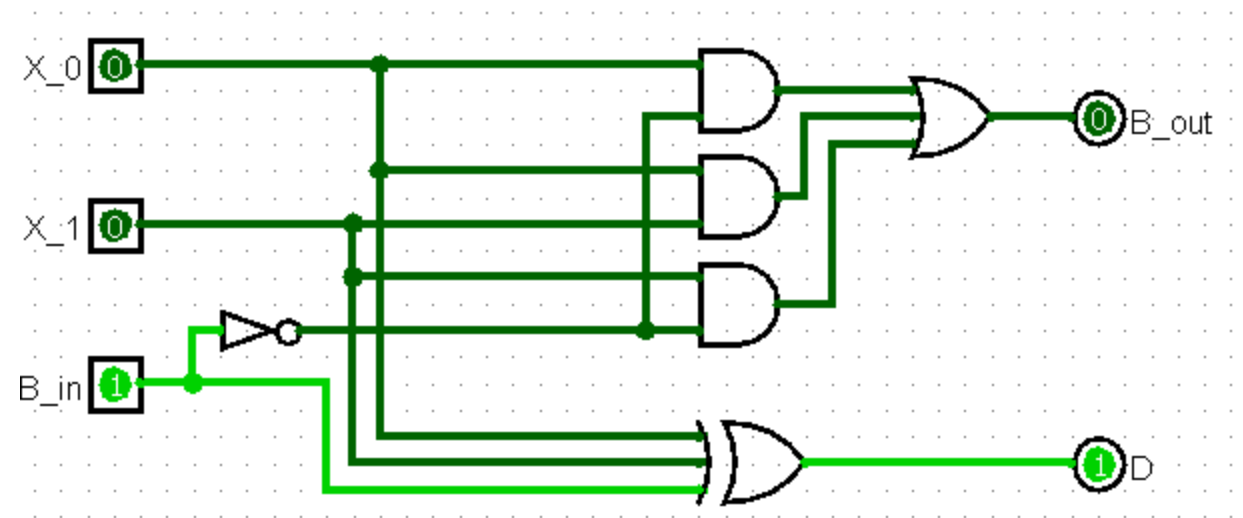
Full Subtractor Logic Circuit:



# Full Subtractor

Full Subtractor Logic Circuit:

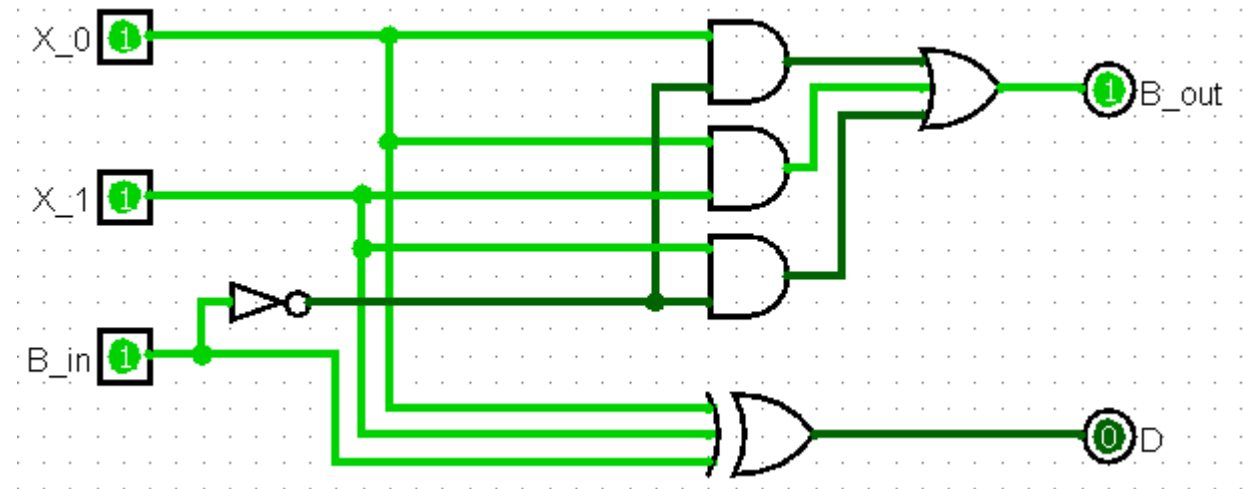
$B_{IN}$	$X_1$	$X_0$	$B_{OUT}$	$D$
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



# Full Subtractor

Full Subtractor Logic Circuit:

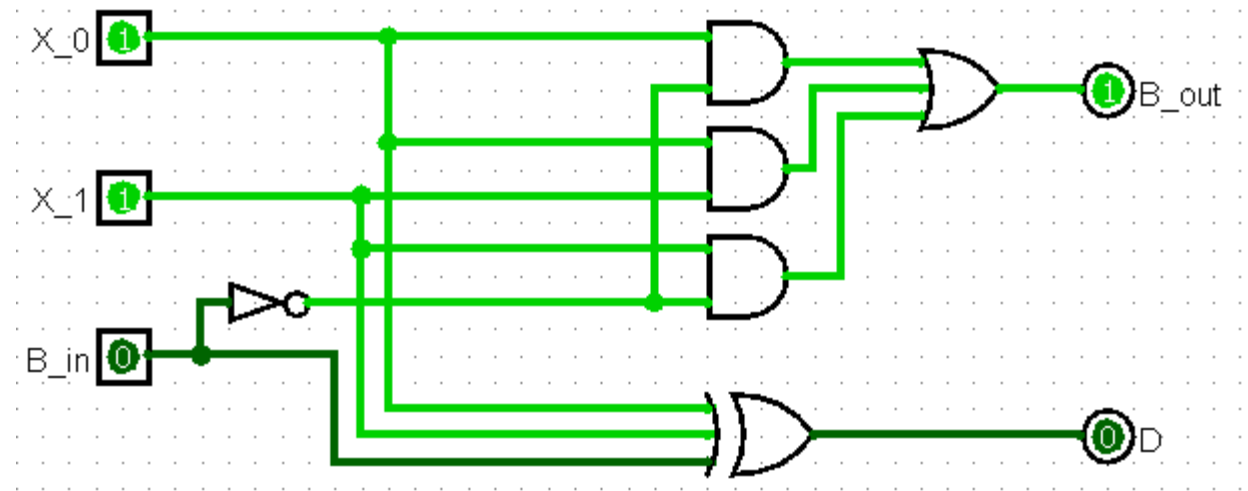
$B_{IN}$	$X_1$	$X_0$	$B_{OUT}$	$D$
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



# Full Subtractor

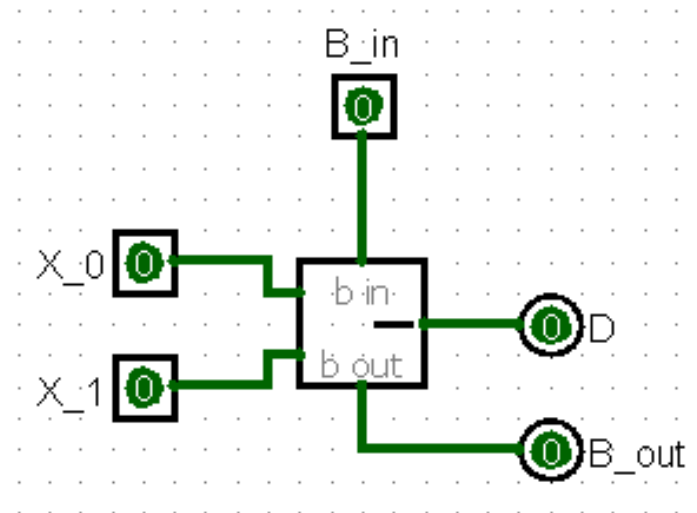
Full Subtractor Logic Circuit:

$B_{IN}$	$X_1$	$X_0$	$B_{OUT}$	$D$
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



# Full Subtractors

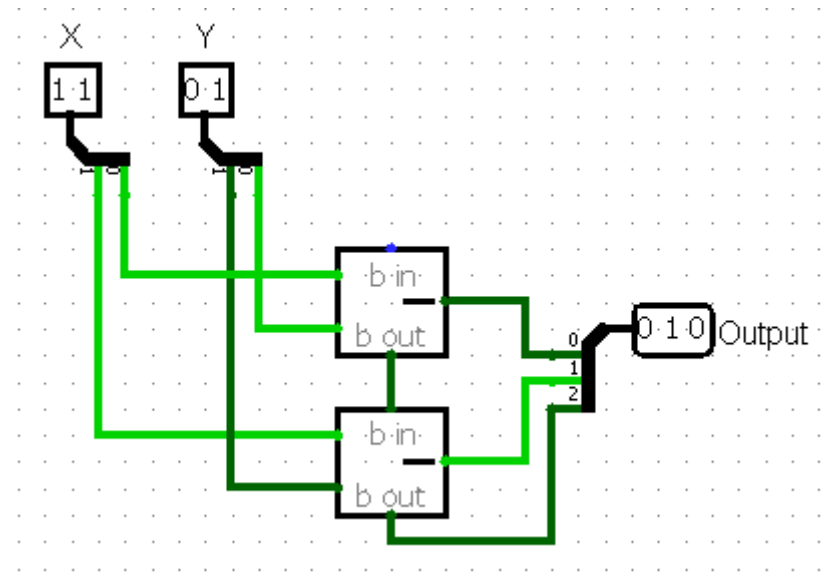
- Abstracted Full Subtractor:



# Full Subtractors

- Like full adders, full subtractors can work together by providing the borrow out of one full subtractor as the borrow in for a second subtractor

$$\begin{array}{r} 11 \\ -01 \\ \hline 010 \end{array}$$

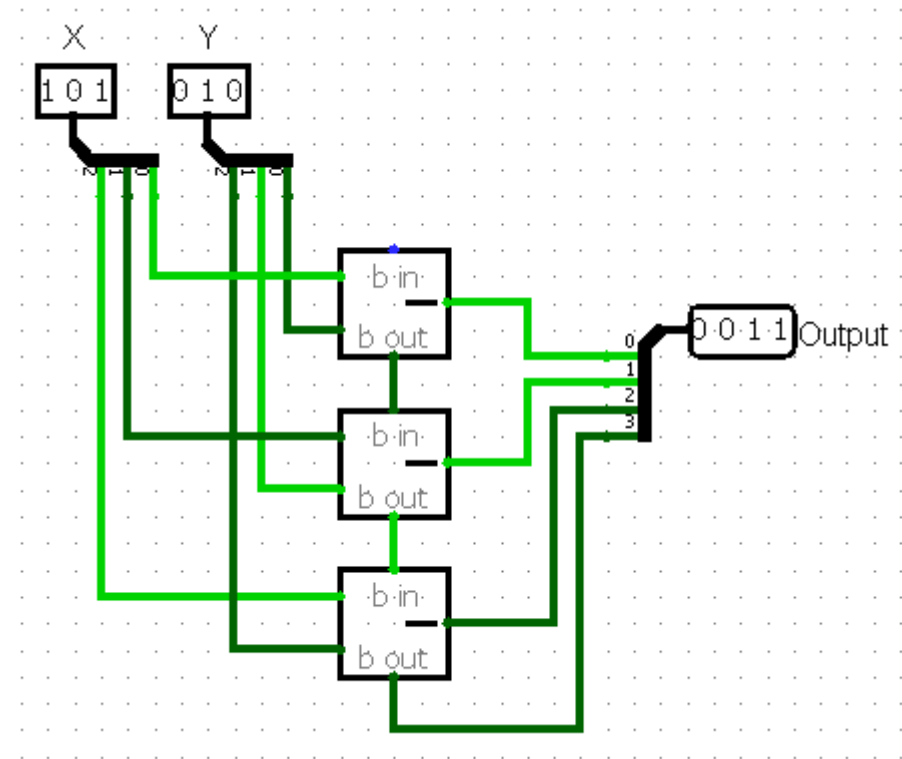




# Full Subtractors

- A 3-bit Subtractor:

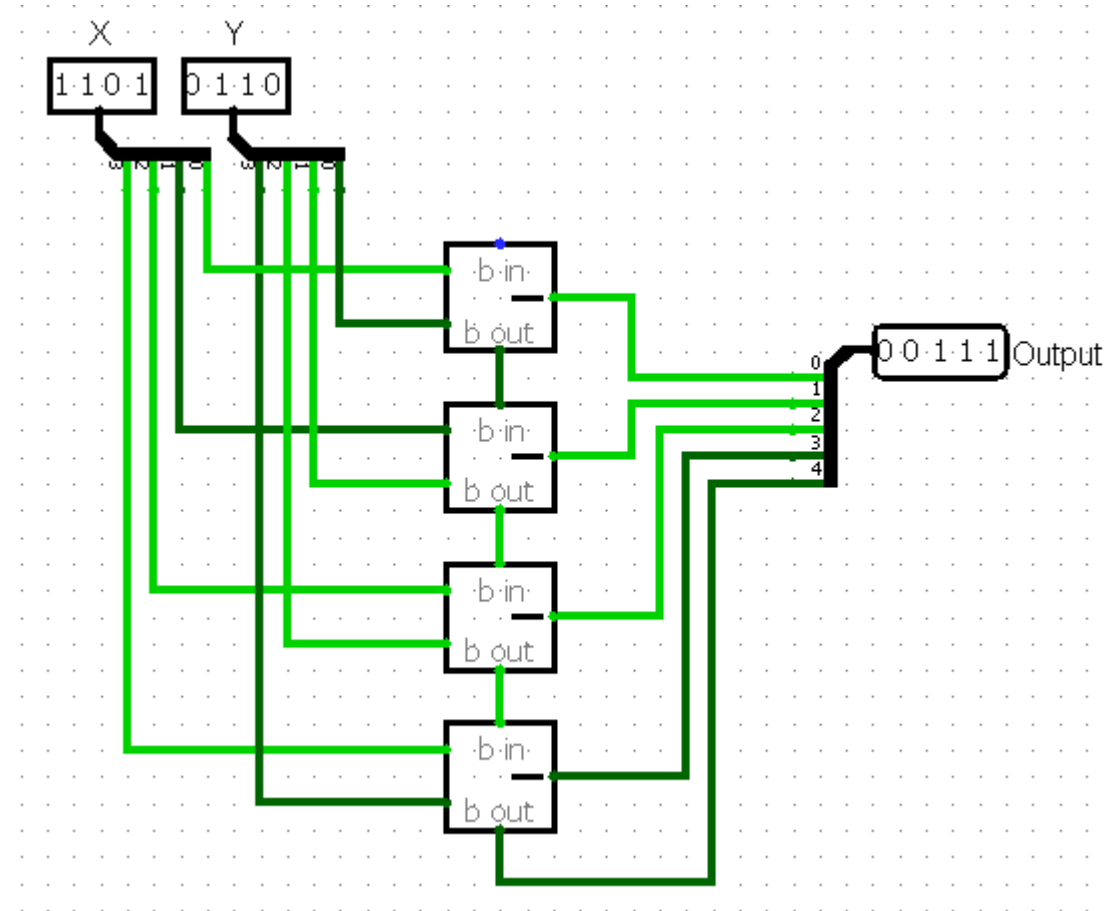
$$\begin{array}{r} 101 \\ - 010 \\ \hline 0011 \end{array}$$



# Full Subtractors

- A 4-bit Subtractor:

$$\begin{array}{r} 1101 \\ - 0110 \\ \hline 0011 \end{array}$$



# Multipliers

- **Multipliers** (not to be confused with multiplexers) are combinational logic circuits capable of performing multiplication
- Note that the multiplication of two 1-bit numbers is a simple *and* operation

$$\begin{array}{r} 0 \\ \times 0 \\ \hline 0 \end{array} \quad \begin{array}{r} 1 \\ \times 0 \\ \hline 0 \end{array} \quad \begin{array}{r} 0 \\ \times 1 \\ \hline 0 \end{array} \quad \begin{array}{r} 1 \\ \times 1 \\ \hline 1 \end{array}$$

$X_0$   
 $Y_0$

$X_0$	$Y_0$	$X_0 \times Y_0$	$X_0 \cdot Y_0$
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	0

# Multipliers

- However, addition will be required when multiplying numbers that are two or more bits.
- We will see how to construct multipliers using full and half adders.
- The largest product of multiplying two, 2-bit numbers is 9:
  - $11 \times 11 = 1001$  ( $3 \times 3 = 9$ )
  - Thus, our circuit must have 4 outputs
    - $P_0$  through  $P_3$

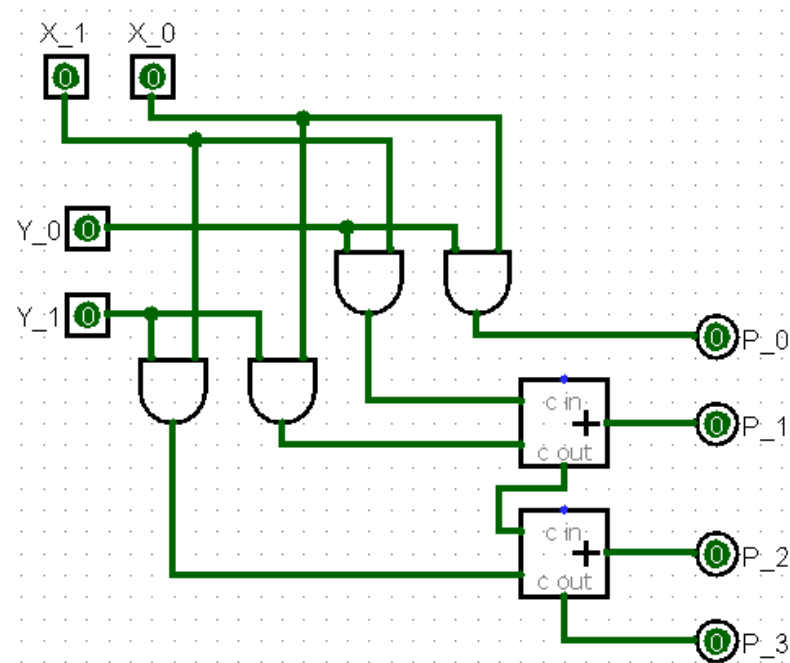
A handwritten binary multiplication diagram. The first number is 11, with bits labeled  $X_1$  (for the left '1') and  $X_0$  (for the right '1'). The second number is 10, with bits labeled  $Y_1$  (for the left '1') and  $Y_0$  (for the right '0'). A horizontal line separates the multiplicand from the multiplier. Below the line, the first partial product is 00, corresponding to  $X_0 \times Y_1$ . The second partial product is 11, corresponding to  $X_1 \times Y_1$ . A horizontal line separates the partial products from the final sum. The final sum is 110, with bits labeled  $P_2$  (for the left '1'),  $P_1$  (for the middle '1'), and  $P_0$  (for the right '0').

$$\begin{array}{r} \begin{array}{cc} X_1 & X_0 \\ 1 & 1 \end{array} \\ \times \begin{array}{cc} Y_1 & Y_0 \\ 1 & 0 \end{array} \\ \hline 00 \\ + 11 \\ \hline 110 \\ \begin{array}{ccc} \nearrow & \nearrow & \nearrow \\ P_2 & P_1 & P_0 \end{array} \end{array}$$

# Multipliers

## 2-bit Multiplier Logic Circuit:

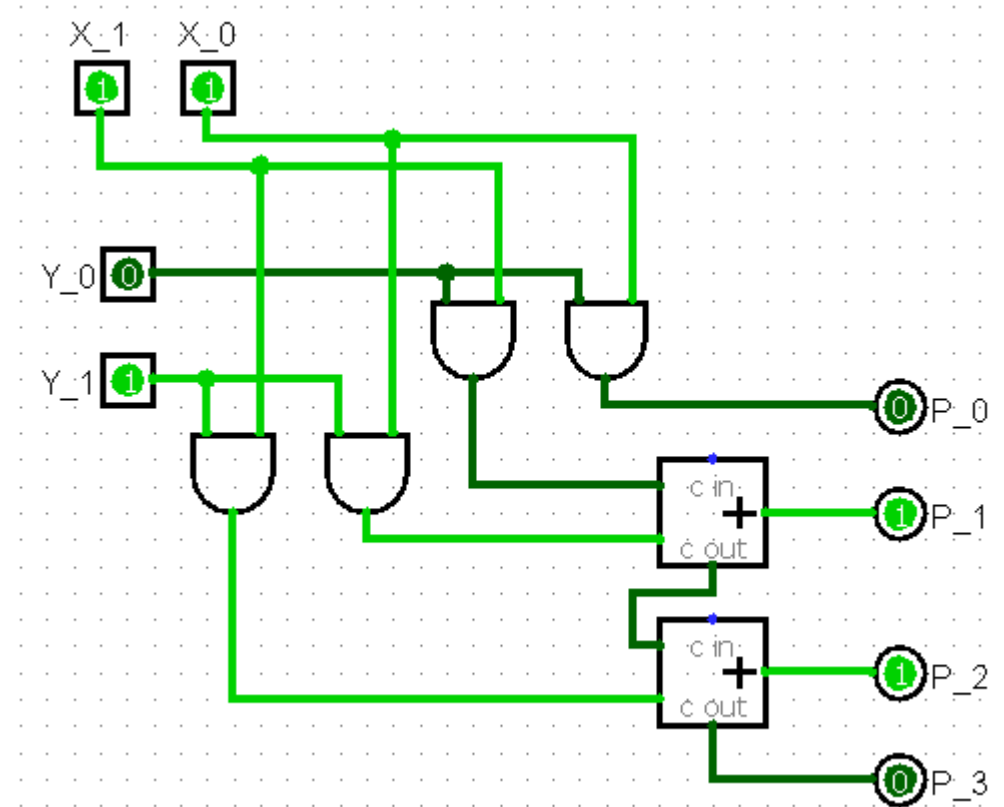
- (Uses 2 half adders)



# Multipliers

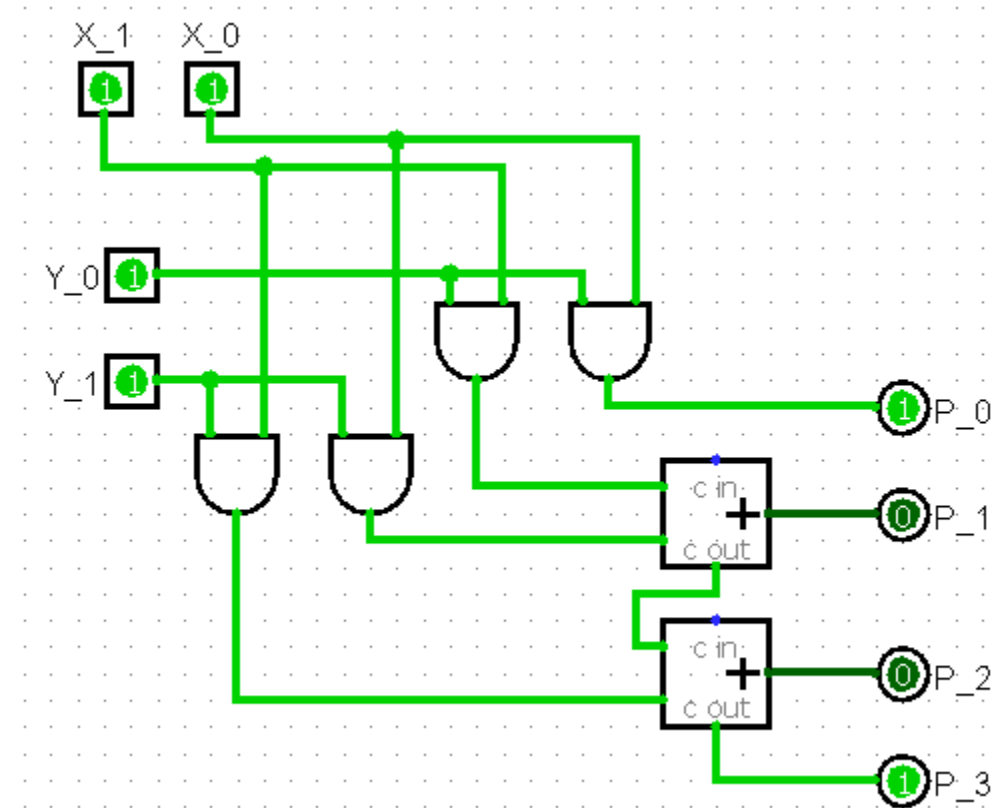
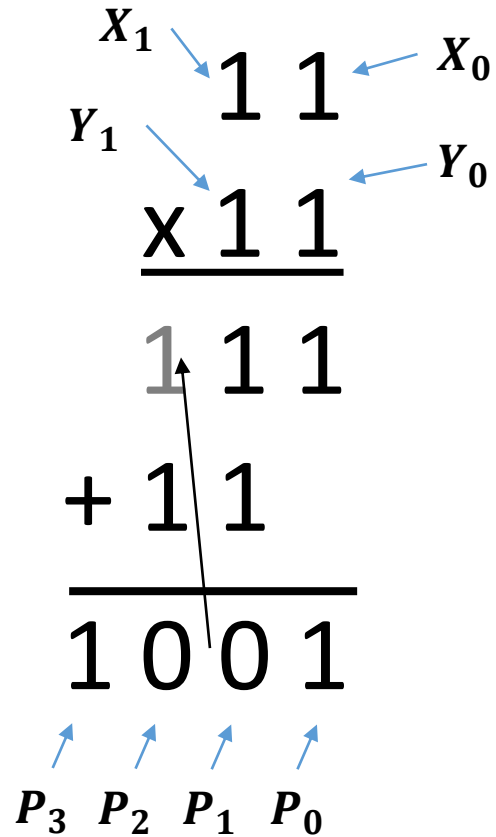
2-bit Multiplier Logic Circuit:

$$\begin{array}{r} X_1 \swarrow \quad \nwarrow X_0 \\ 1 \ 1 \\ Y_1 \swarrow \quad \nwarrow Y_0 \\ \times 1 \ 0 \\ \hline 0 \ 0 \\ + \ 1 \ 1 \\ \hline 1 \ 1 \ 0 \\ \swarrow \quad \swarrow \quad \swarrow \\ P_2 \ P_1 \ P_0 \end{array}$$



# Multipliers

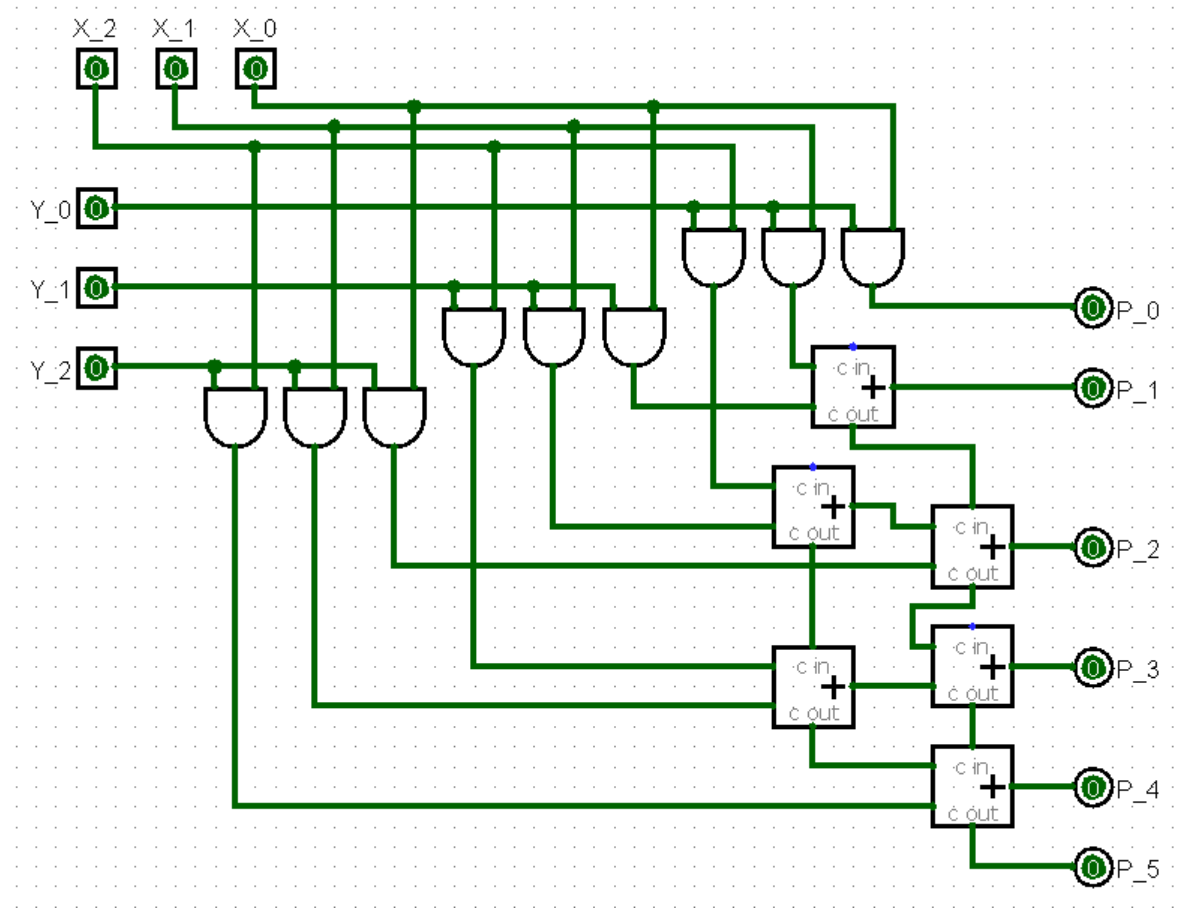
2-bit Multiplier Logic Circuit:



# Multipliers

## 3-bit Multiplier Logic Circuit:

- (Uses 2 half adders)
- (Uses 3 full adders)

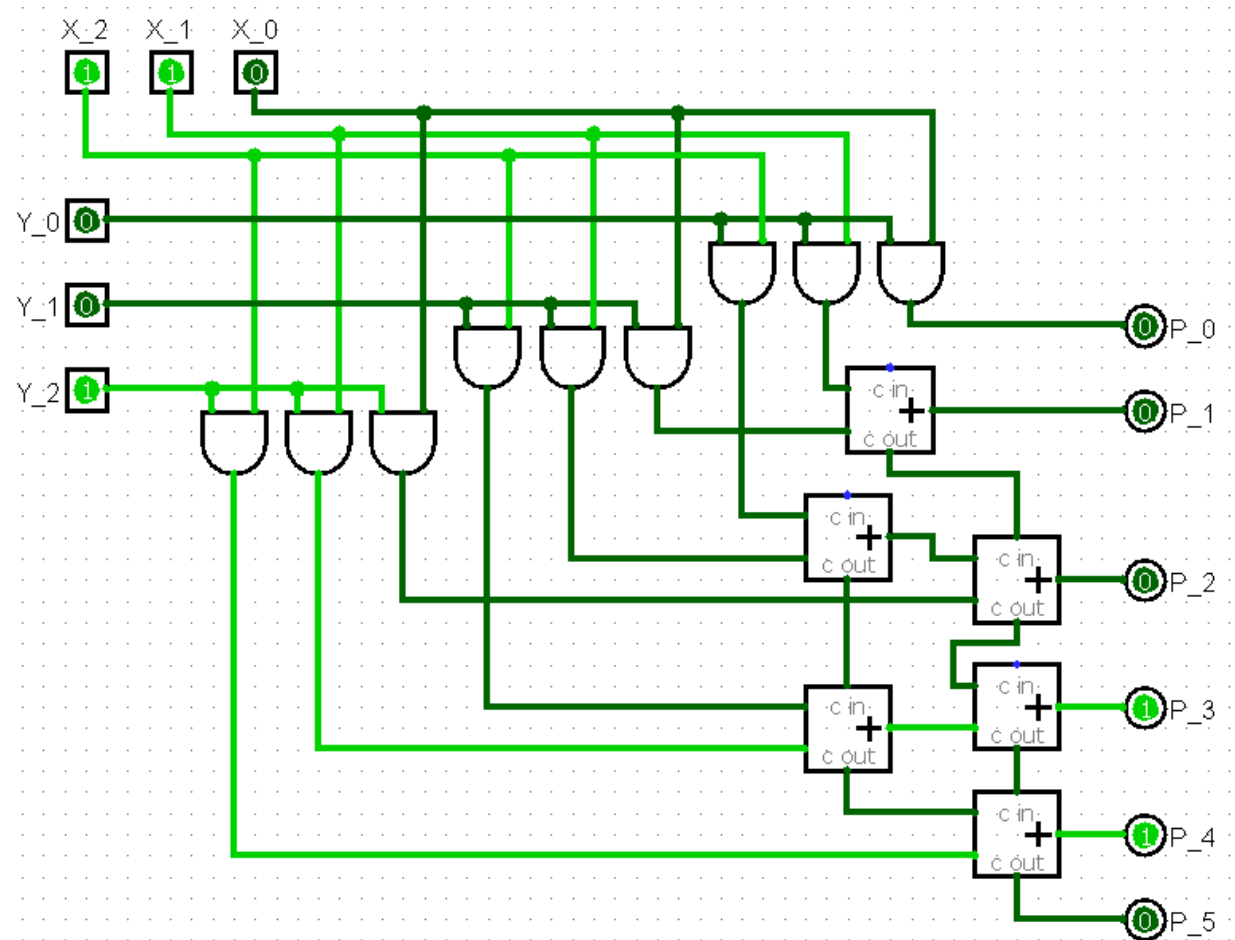




# Multipliers

3-bit Multiplier Logic Circuit:

$$\begin{array}{r} 110 \\ \times 100 \\ \hline 000 \\ 000 \\ + 110 \\ \hline 011000 \end{array}$$



# Multipliers

3-bit Multiplier Logic Circuit:

$$\begin{array}{r} 111 \\ \times 111 \\ \hline 111 \\ 111 \\ + 111 \\ \hline 110001 \end{array}$$

