Revised: 2/25/2021

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).

$$\begin{array}{rrrrl} 0 & 1 & 0 & 1 & \boldsymbol{X_0} \\ \underline{+\,0} & \underline{+\,0} & \underline{+\,1} & \underline{+\,1} & \boldsymbol{X_1} \\ 00 & 01 & 01 & 10 & \end{array}$$

$\boldsymbol{C\ (Carry)}$ $\boldsymbol{S\ (Sum)}$

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

$$\begin{array}{r} 0 \\ +\,0 \\ \hline 00 \end{array} \qquad \begin{array}{r} 1 \\ +\,0 \\ \hline 01 \end{array} \qquad \begin{array}{r} 0 \\ +\,1 \\ \hline 01 \end{array} \qquad \begin{array}{rl} 1 & \leftarrow X_0 \\ +\,1 & \leftarrow X_1 \\ \hline 10 & \end{array}$$

$C\ (Carry)$ $\quad$ $S\ (Sum)$

Half Adders

SOP Expressions:

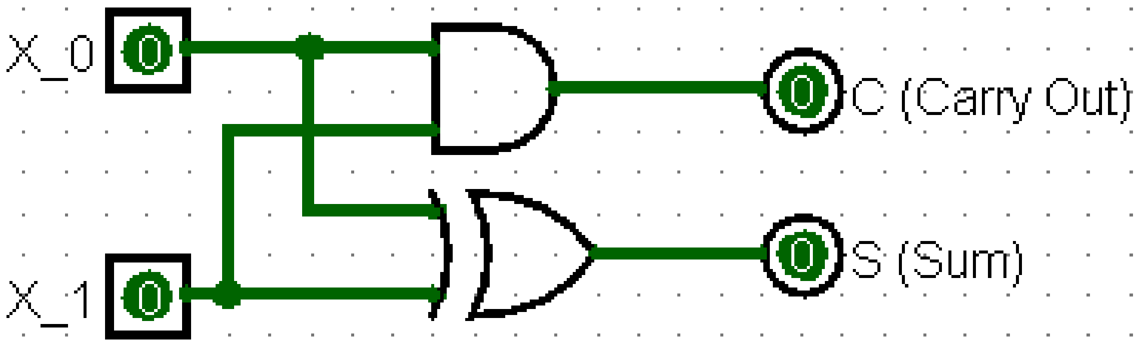
$$C = X_1 X_0$$

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

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

Half Adders

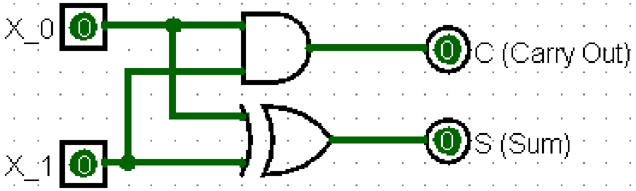
Half Adder Logic Circuit:



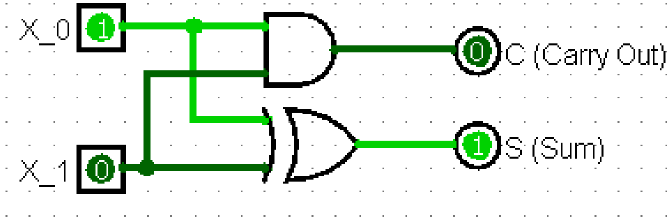
Half Adders

Half Adder Logic Circuit:

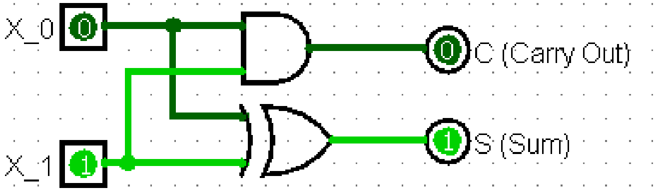
0
<u>+ 0</u>
00



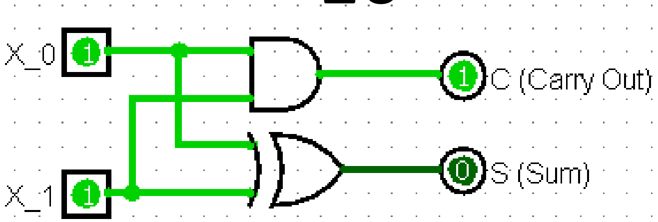
1
<u>+ 0</u>
01



0
<u>+ 1</u>
01



1
<u>+ 1</u>
10



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).

$C_{IN}(Carry\ In)$ X_1 X_0 $C_{OUT}(Carry\ Out)$ $S\ (Sum)$

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:

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

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

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

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

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

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

Full Adders

Simplifying:

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

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

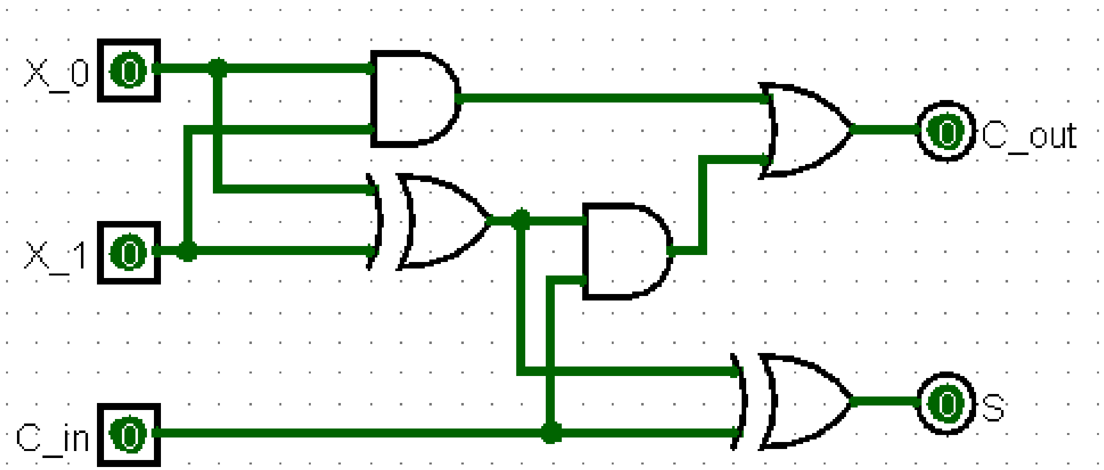
$$S = \overline{C_{IN}}\,(X_0 \oplus X_1) + C_{IN}(\overline{X_0 \oplus X_1})$$

$$\bar{X}Y + X\bar{Y} = X \oplus Y$$

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

Full Adders

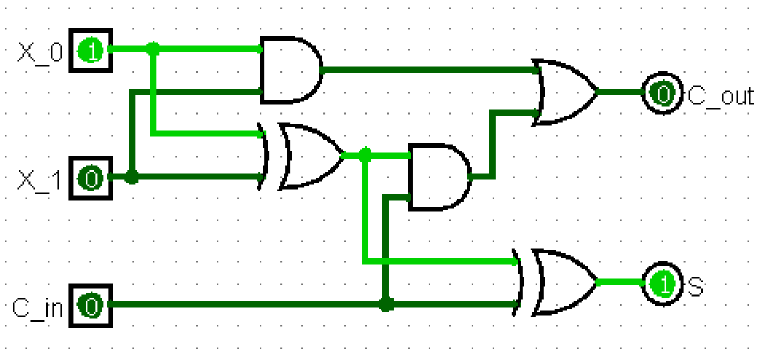
Full Adder Logic Circuit:

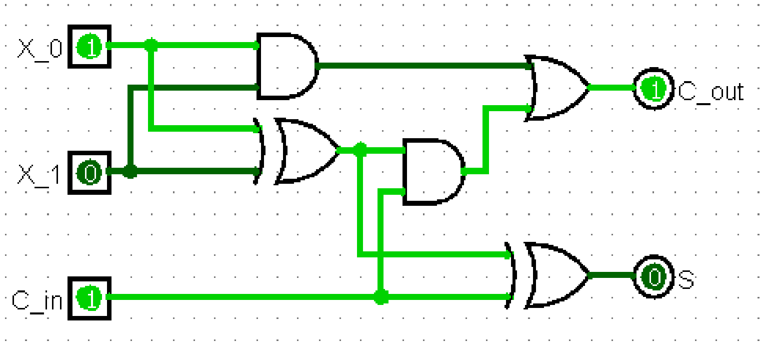


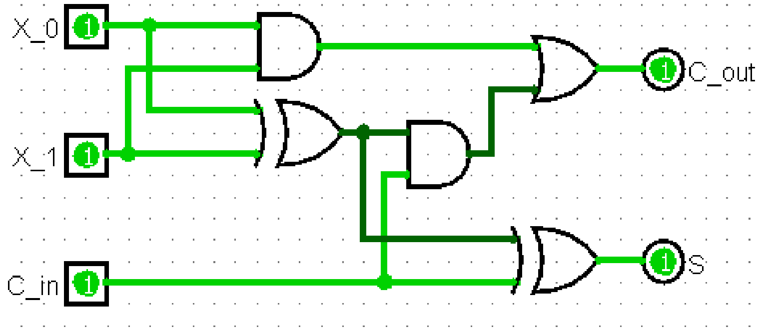
Full Adders

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**

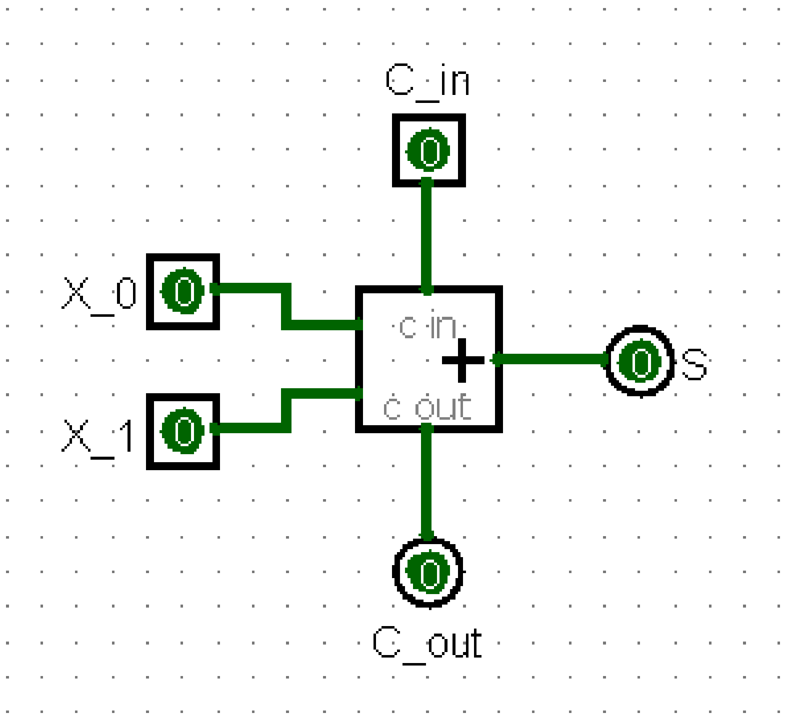






Full Adders

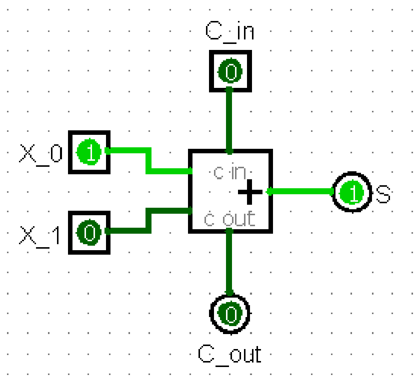
- Abstracted Full Adder:

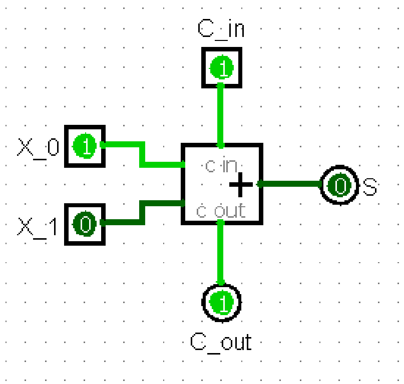


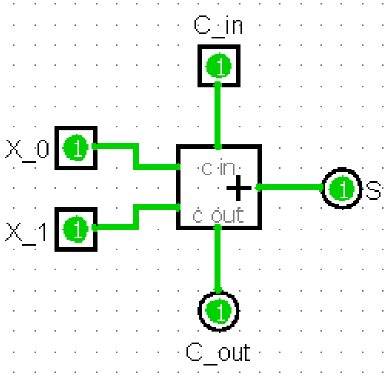
Full Adders

- 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

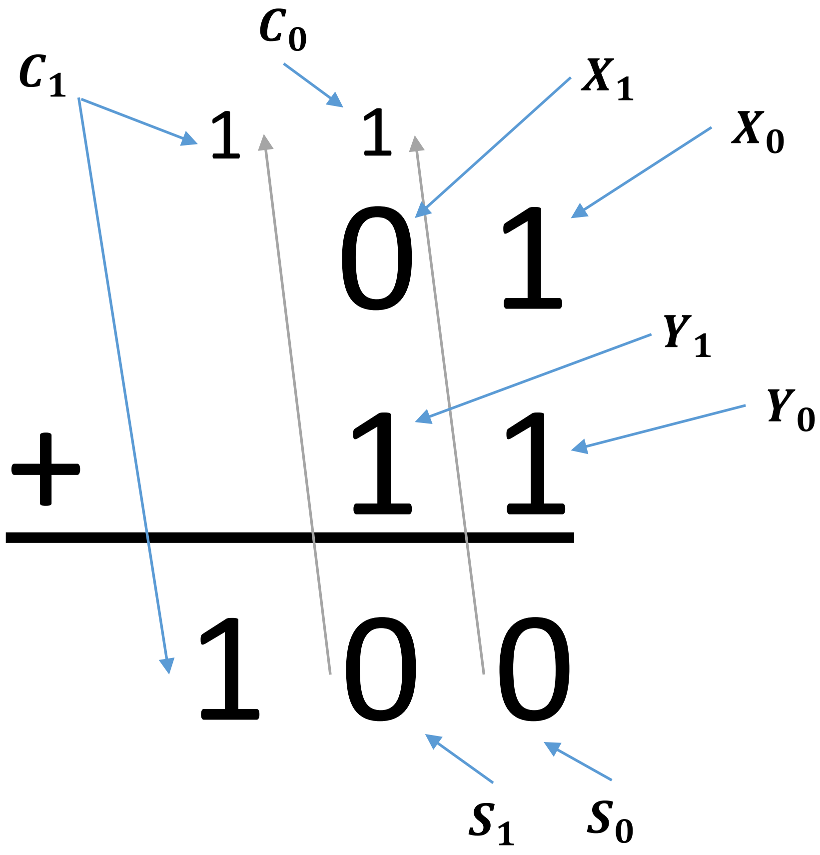
- So far, we've seen only how adders can add single digits together
  - 1+1+1 or 0+1 or 1+0+1 is no problem
  - What if we wanted to add 10+11 or 11101+10101?

- 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*

$$
\begin{array}{r}
0\ 0\ 1 \\
+\ \underline{0\ 1\ 1} \\
1\ 0\ 0
\end{array}
$$

Full Adders

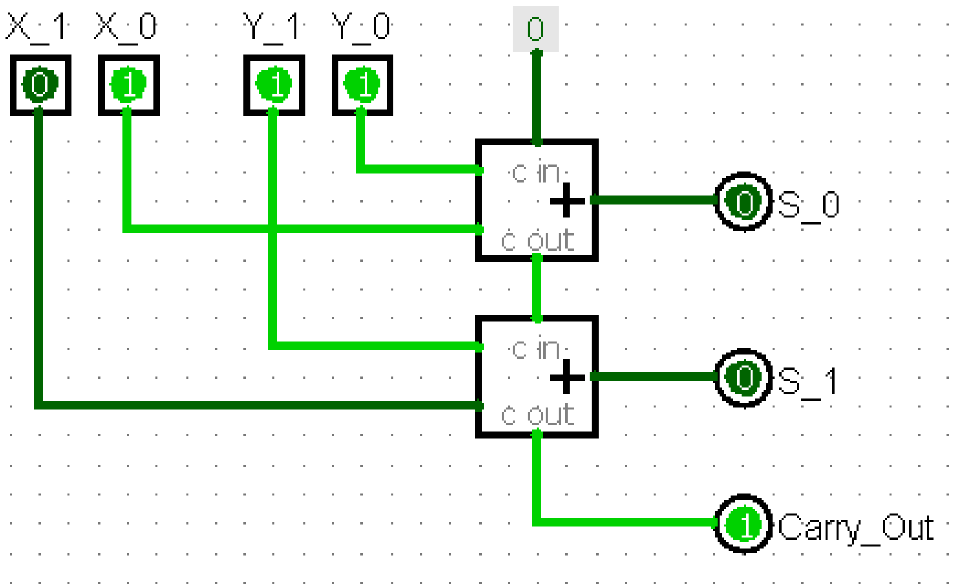
- For example:



Full Adders

- A 2-bit Adder:

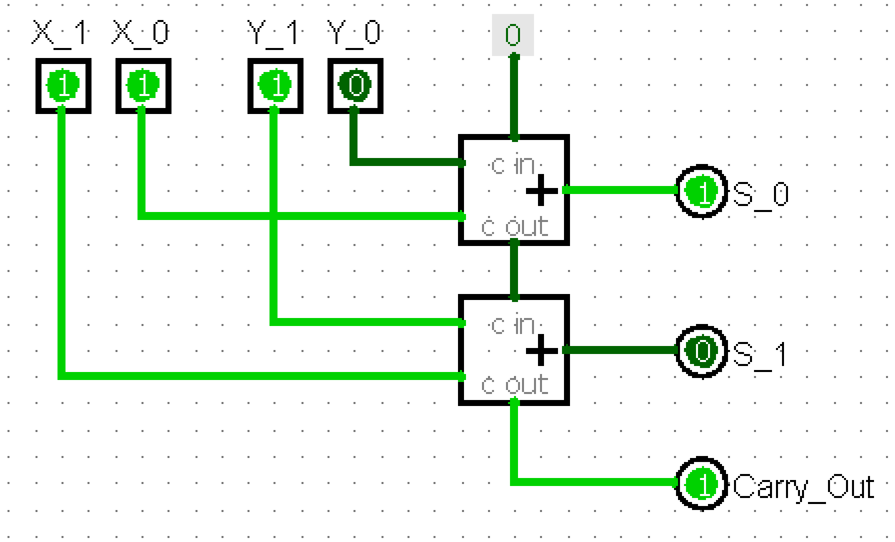
$$\begin{array}{r} 0\ 1 \\ +\quad 1\ 1 \\ \hline 1\ 0\ 0 \end{array}$$



Full Adders

- Another example:

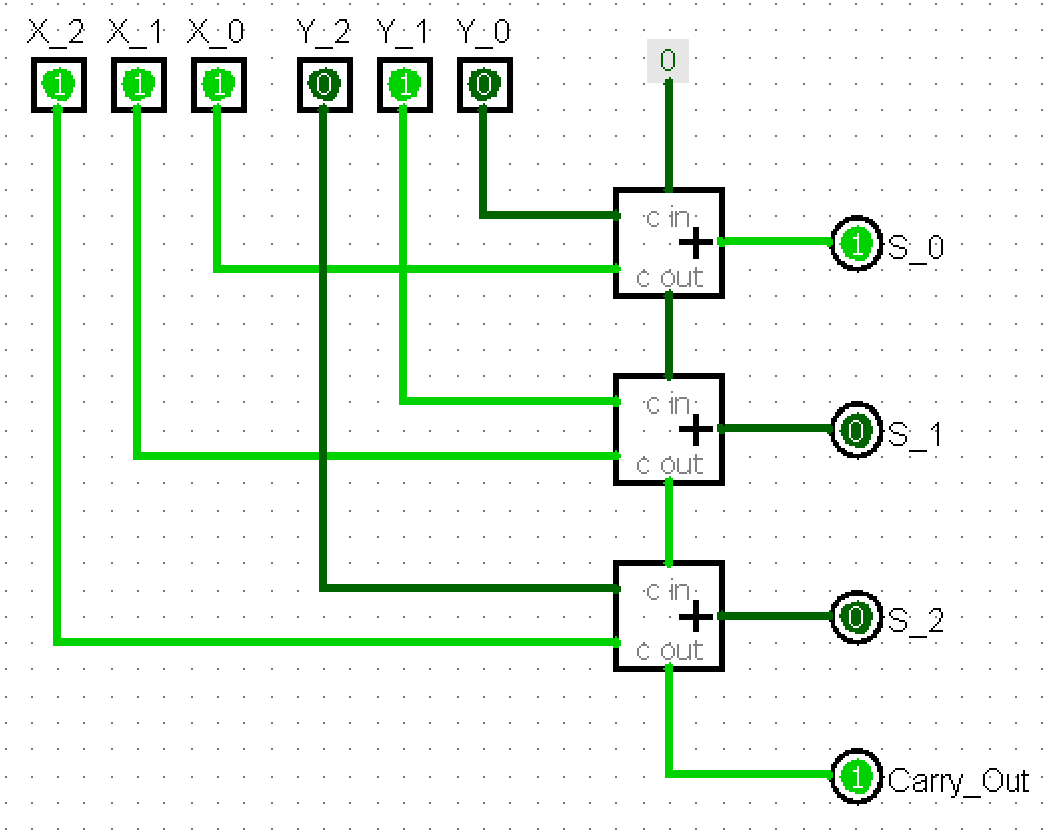
$$\begin{array}{rr} & 1\ 1 \\ + & 1\ 0 \\ \hline & 1\ 0\ 1 \end{array}$$



Full Adders

- A 3-bit Adder:

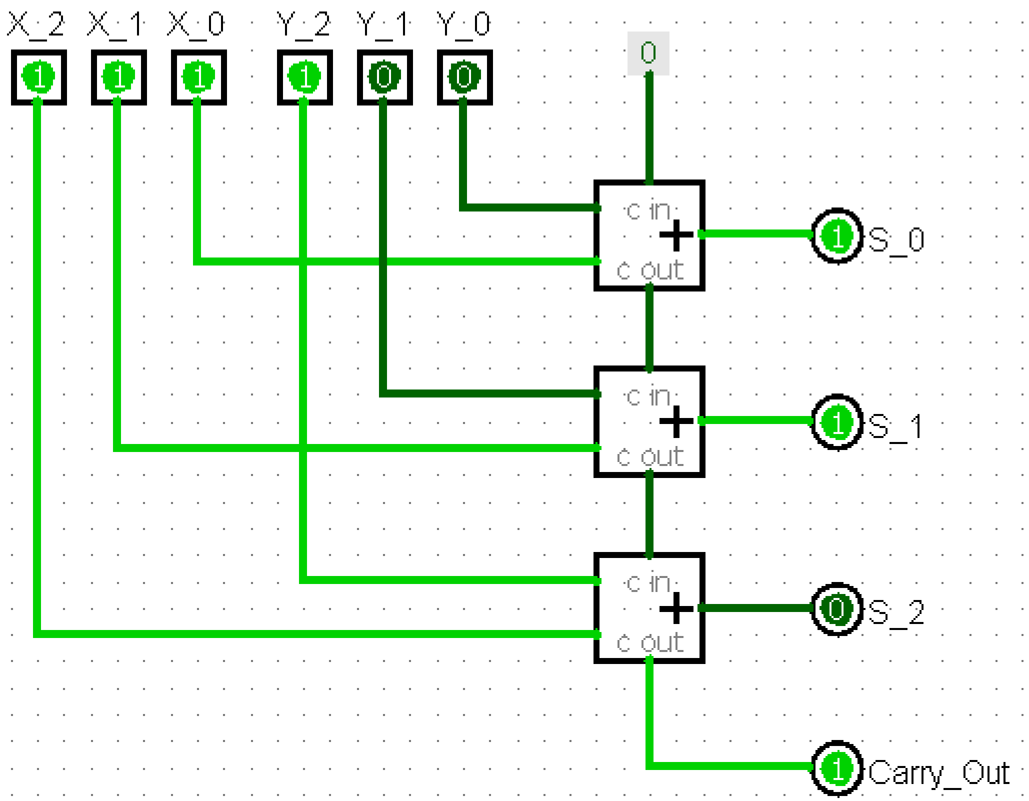
$$\begin{array}{r} 1\;1\;1 \\ +\quad 0\;1\;0 \\ \hline 1\;0\;0\;1 \end{array}$$



Full Adders

- Another example:

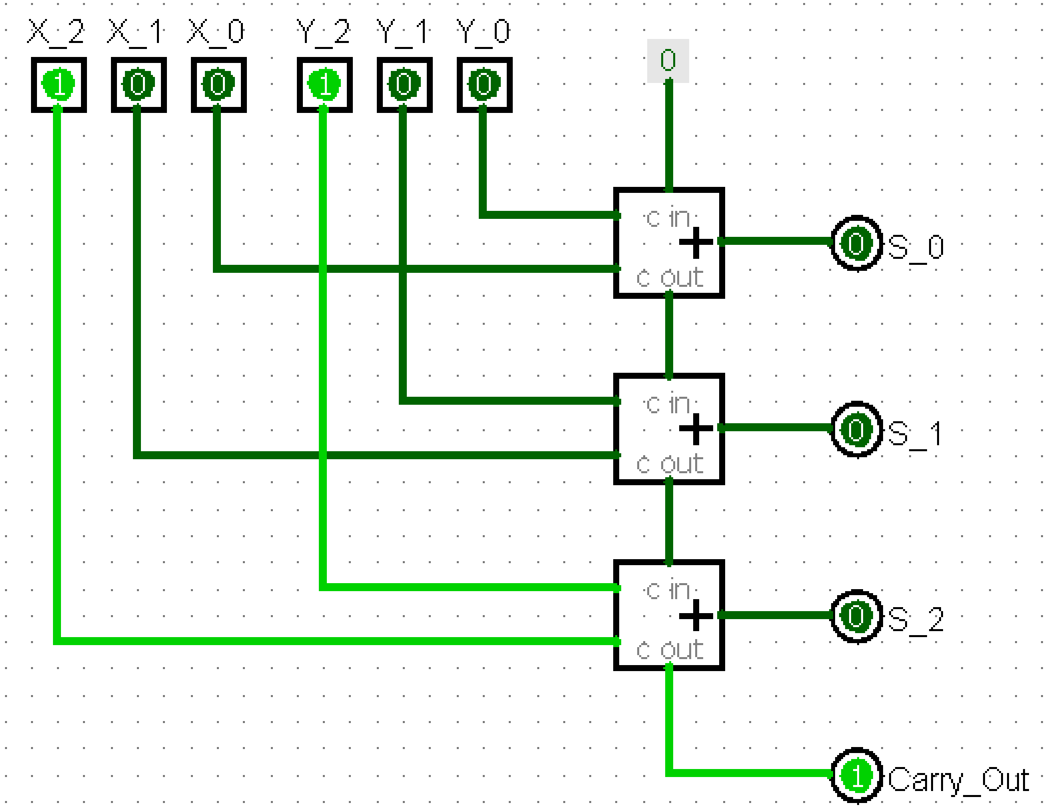
$$
\begin{array}{r}
1\ 1\ 1 \\
+ \quad 1\ 0\ 0 \\
\hline
1\ 0\ 1\ 1
\end{array}
$$



Full Adders

- Another example:

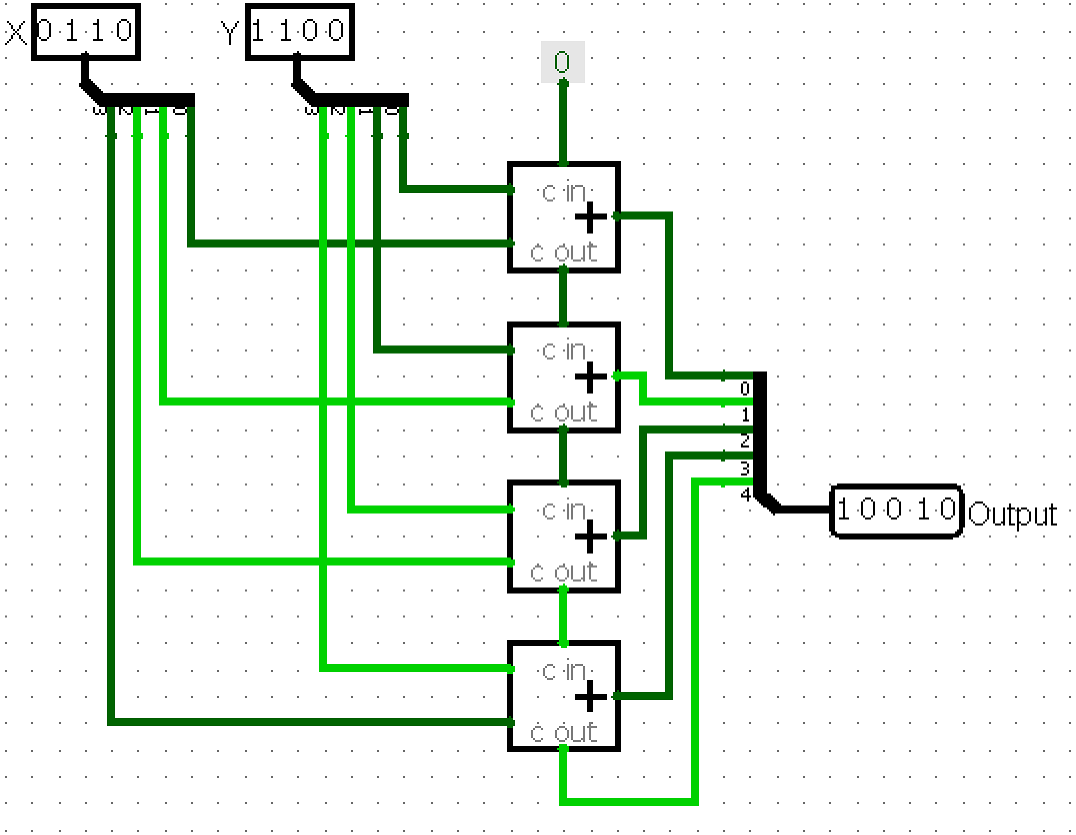
$$\begin{array}{r} 1\ 0\ 0 \\ +\quad 1\ 0\ 0 \\ \hline 1\ 0\ 0\ 0 \end{array}$$



Full Adders

- A 4-bit Adder (with buses):

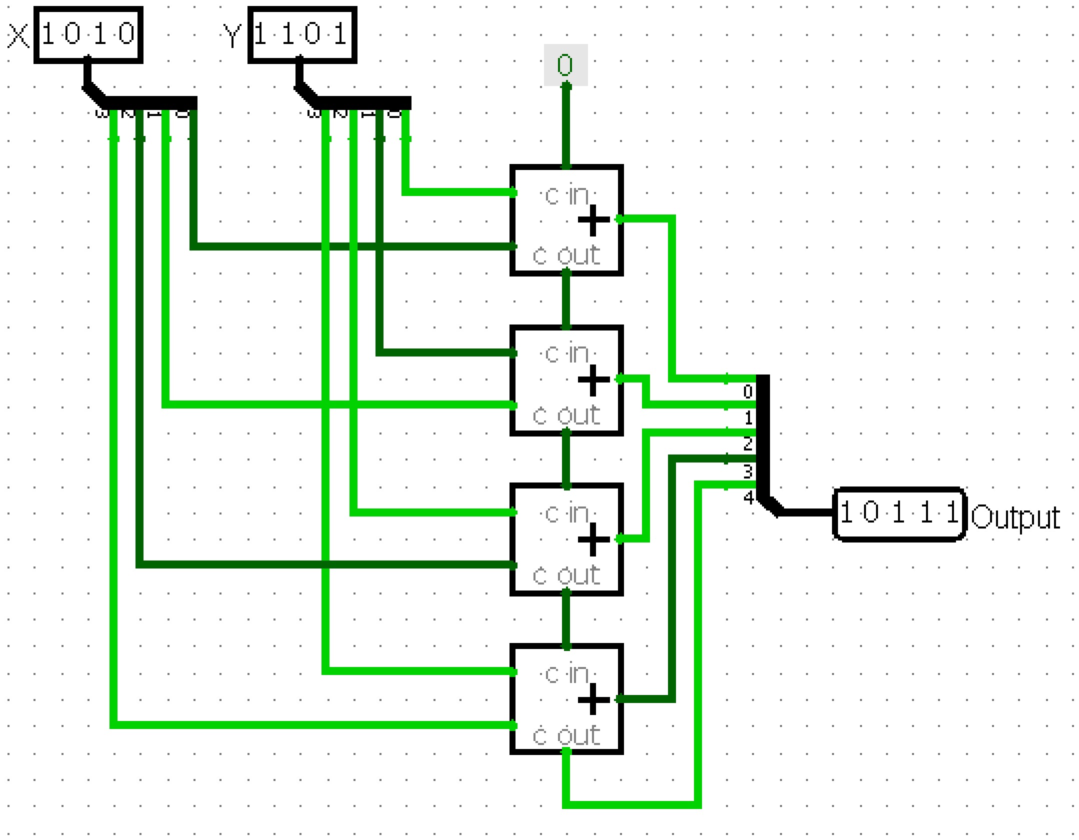
$$
\begin{array}{r}
0\ 1\ 1\ 0 \\
+\quad 1\ 1\ 0\ 0 \\
\hline
1\ 0\ 0\ 1\ 0
\end{array}
$$



Full Adders

- Another example

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



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).

0	1	0	1	X_0
- 0	- 0	- 1	- 1	X_1
00	01	11	00	

$B\ (Borrow)$ $D\ (Difference)$

Negative 1 in two's complement

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

$$\begin{array}{rrrrl} 0 & 1 & 0 & 1 & X_0 \\ -\ 0 & -\ 0 & -\ 1 & -\ 1 & X_1 \\ \hline 00 & 01 & 11 & 00 & \end{array}$$

$B\ (Borrow)$ $\quad$ $D\ (Difference)$

Half Subtractors

SOP Expressions:

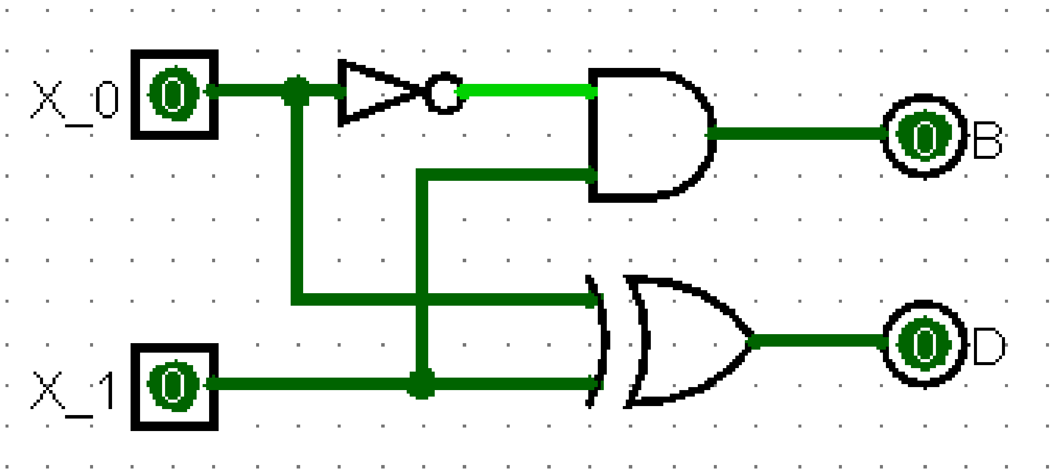
$$B = X_1\overline{X_0}$$

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

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

Half Subtractor

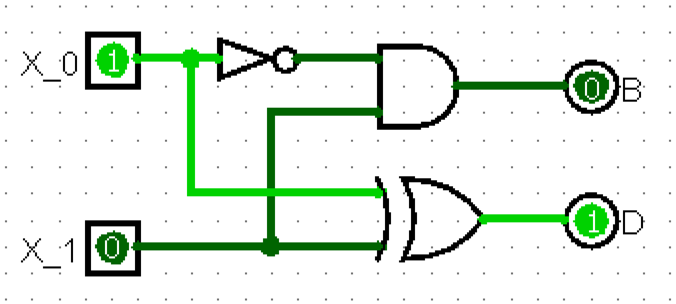
Half Subtractor Logic Circuit:

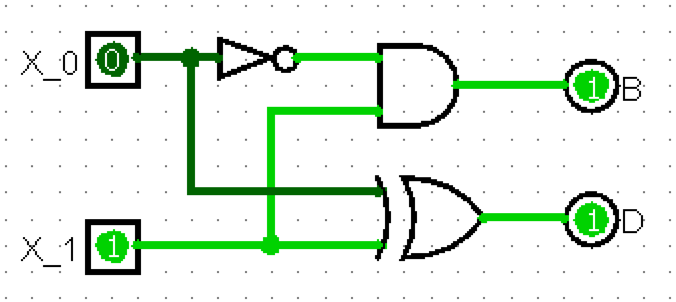


Half Subtractor

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 different and the borrow).

X_1 X_0 $B_{OUT}(Borrow\ Out)$

$B_{IN}(Borrow\ In)$ D $(Difference)$

0 - 0 - 0 = 0 0

0 - 0 - 1 = 1 1 -1

0 - 1 - 0 = 1 1

0 - 1 - 1 = 1 0 -2

1 - 0 - 0 = 0 1

1 - 0 - 1 = 0 0

1 - 1 - 0 = 0 0

1 - 1 - 1 = 1 1

Full Subtractors

SOP Expressions:

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

$$D = \overline{B_{IN}}\,\overline{X_1}X_0 + \overline{B_{IN}}X_1\overline{X_0} + B_{IN}\overline{X_1}\,\overline{X_0} + B_{IN}X_1X_0$$

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:

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

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

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

Full Subtractors

Simplifying:

$$D = \overline{B_{IN}}\,\overline{X_1}X_0 + \overline{B_{IN}}X_1\overline{X_0} + B_{IN}\overline{X_1}\,\overline{X_0} + B_{IN}X_1X_0$$

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

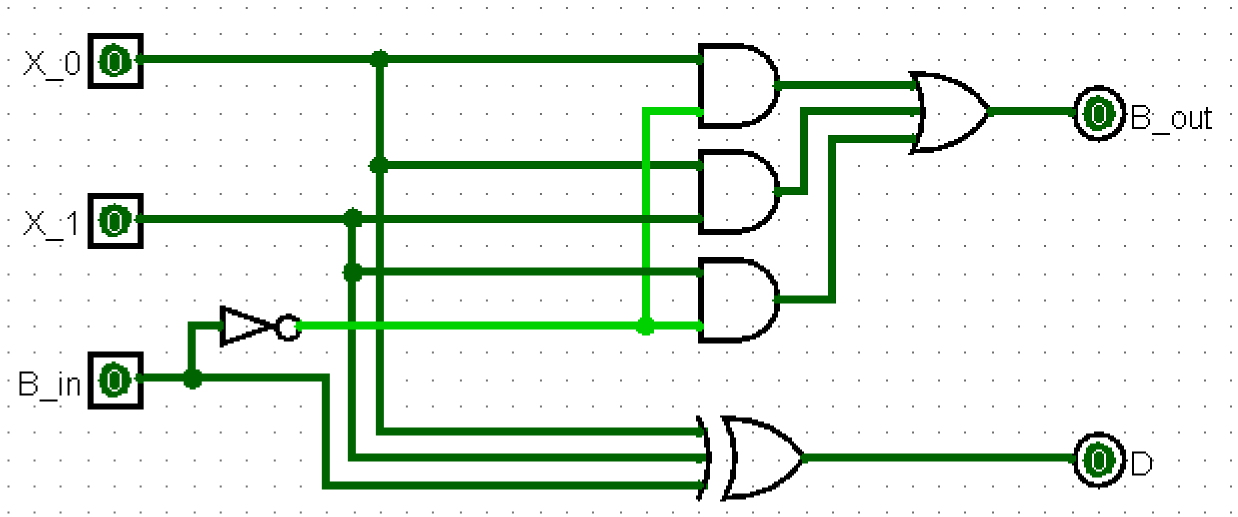
$$D = \overline{B_{IN}}(X_0 \oplus X_1) + B_{IN}(\overline{X_0 \oplus X_1})$$

$$\bar{X}Y + X\bar{Y} = X \oplus Y$$

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

Full Subtractor

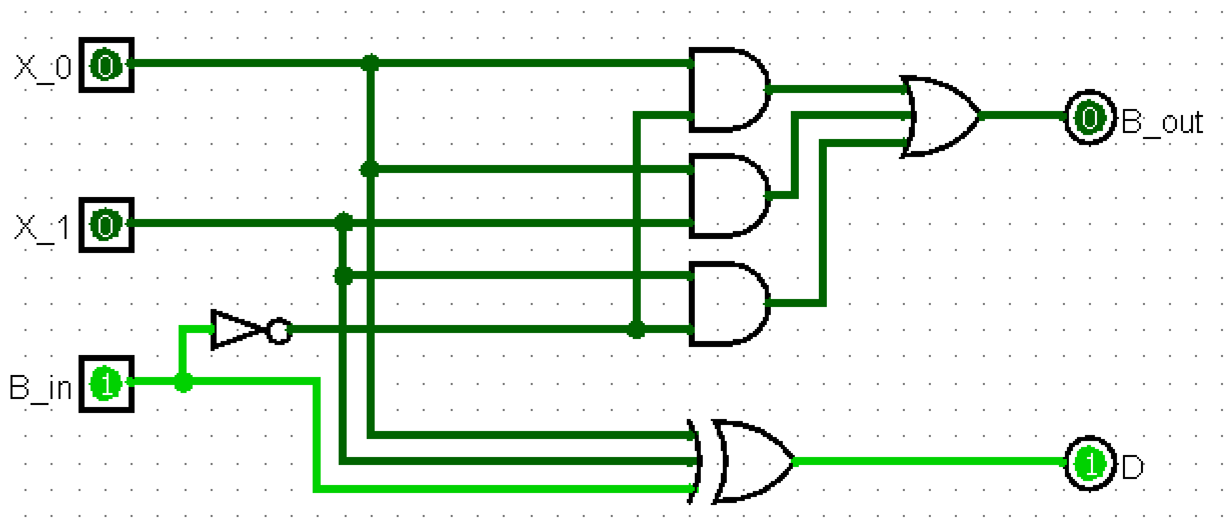
Full Subtractor Logic Circuit:



Full Subtractor

Full Subtractor Logic Circuit:

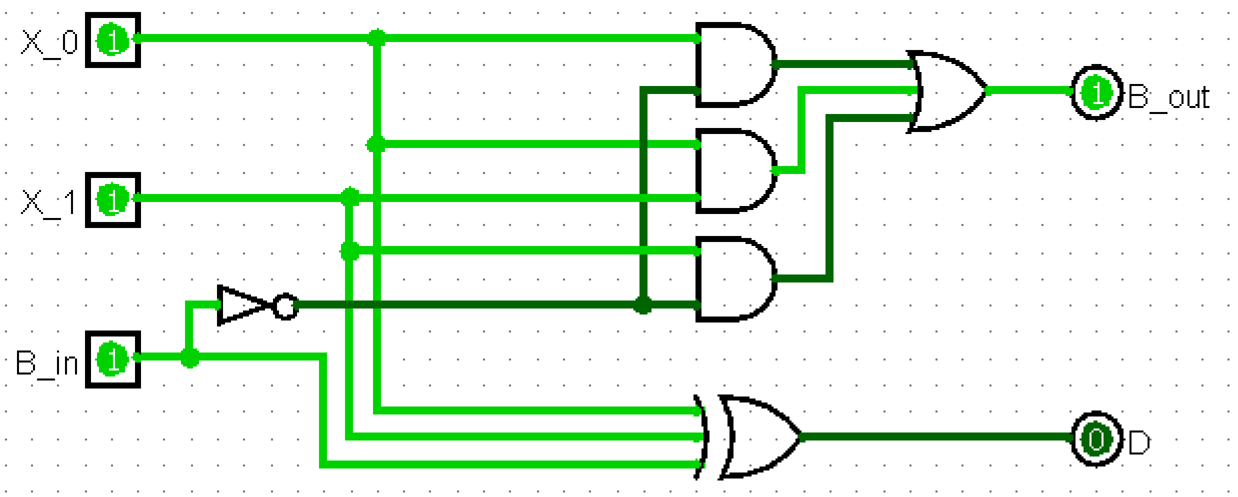
B_{IN}	X_1	X_0	B_{OUT}	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	**0**	**0**	**0**	**1**
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



Full Subtractor

Full Subtractor Logic Circuit:

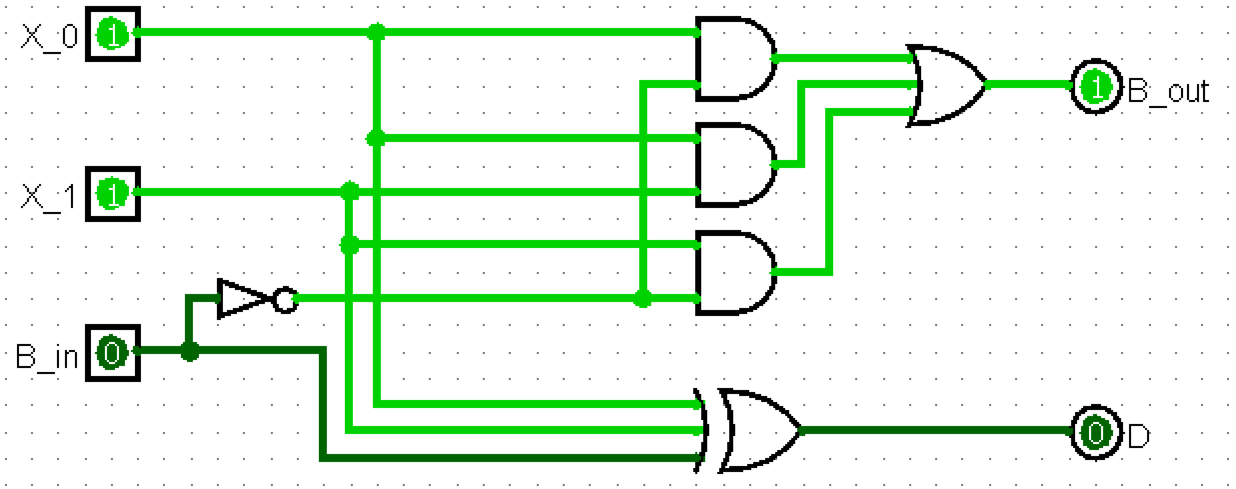
B_{IN}	X_1	X_0	B_{OUT}	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	**1**	**1**	**1**	**1**



Full Subtractor

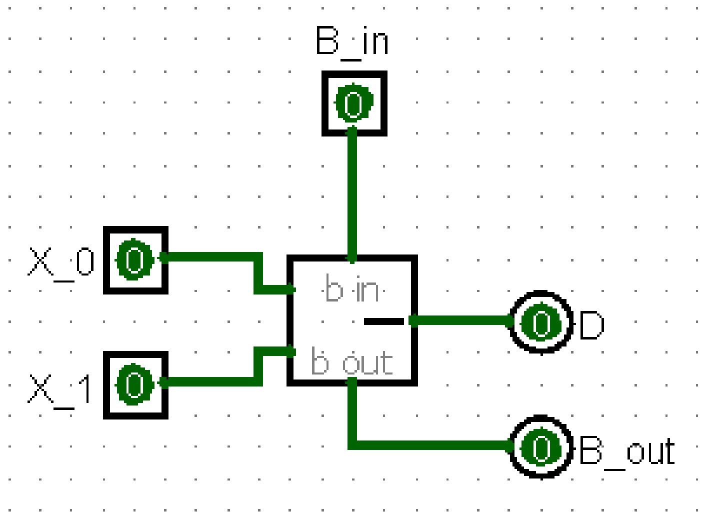
Full Subtractor Logic Circuit:

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

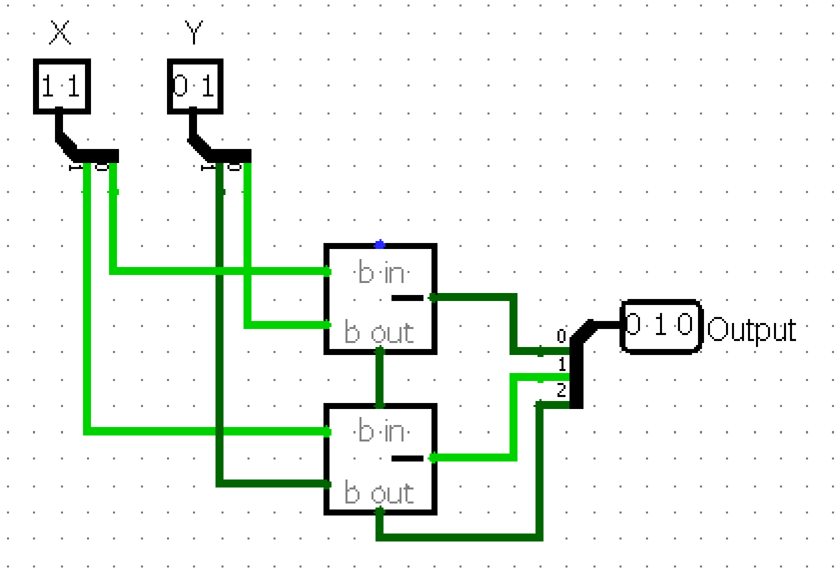
- Abstracted Full Subtractor:



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

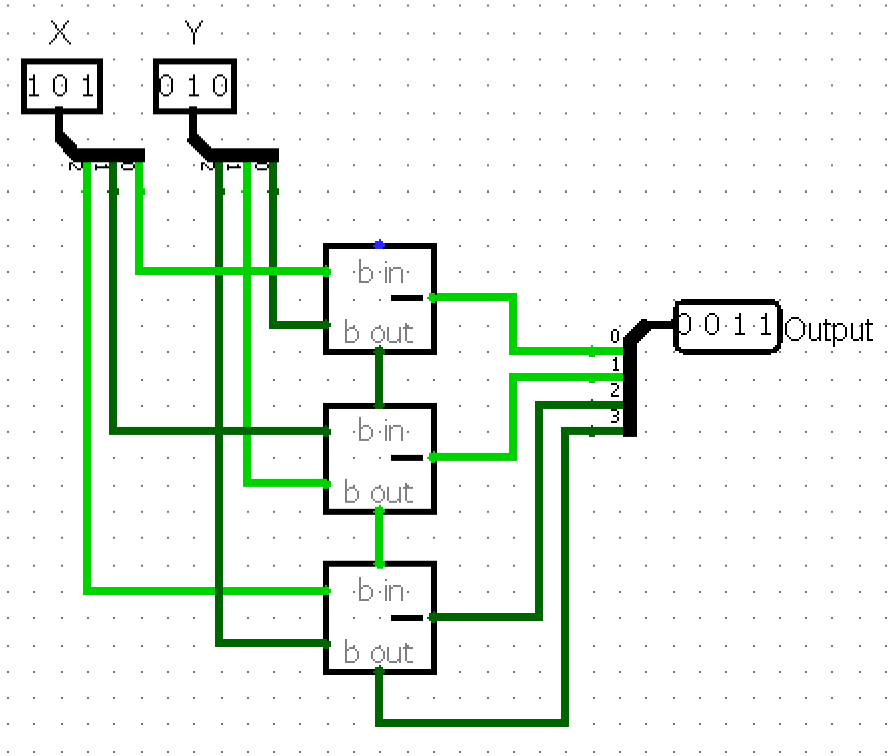
$$
\begin{array}{r}
1\ 1 \\
-\quad 0\ 1 \\
\hline
0\ 1\ 0
\end{array}
$$



Full Subtractors

- A 3-bit Subtractor:

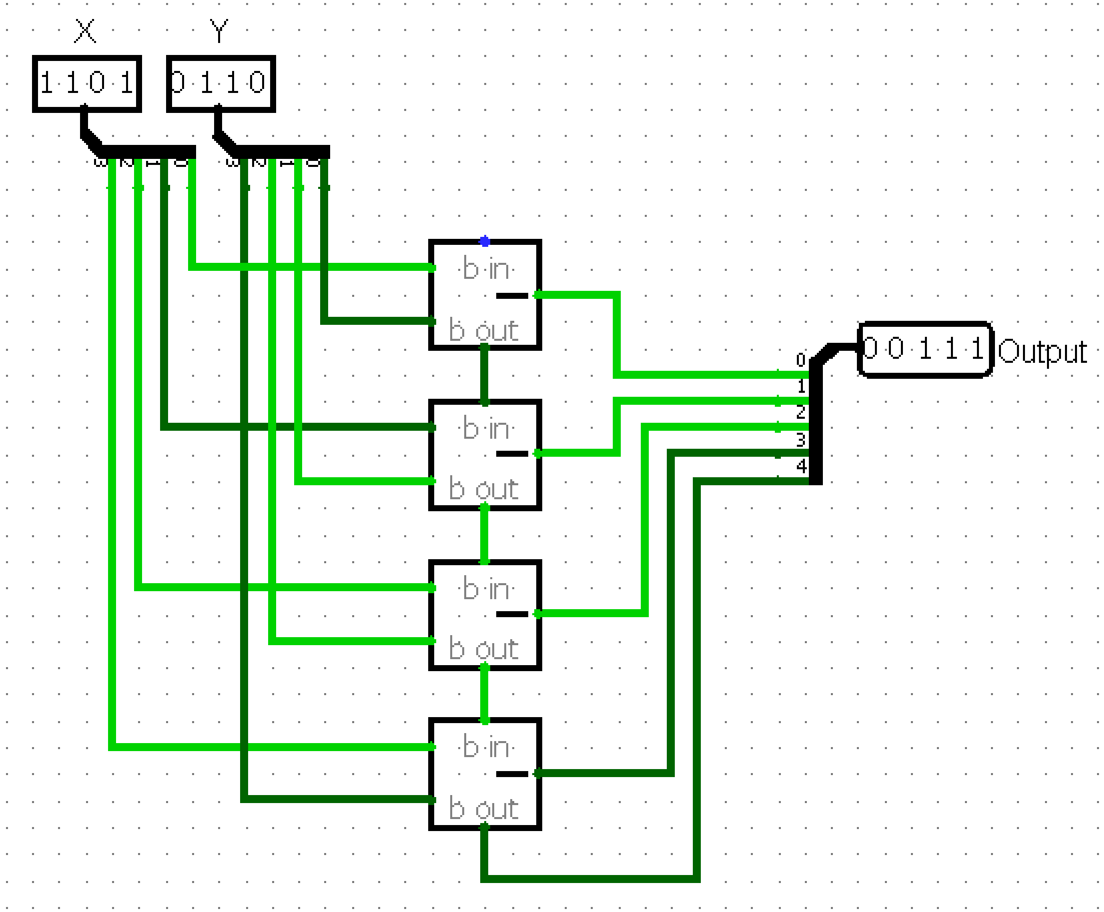
$$
\begin{array}{r}
1\ 0\ 1 \\
-\ \ 0\ 1\ 0 \\
\hline
0\ 0\ 1\ 1
\end{array}
$$



Full Subtractors

- A 4-bit Subtractor:

```
   1 1 0 1
-  0 1 1 0
----------
 0 0 1 1 1
```



Multipliers

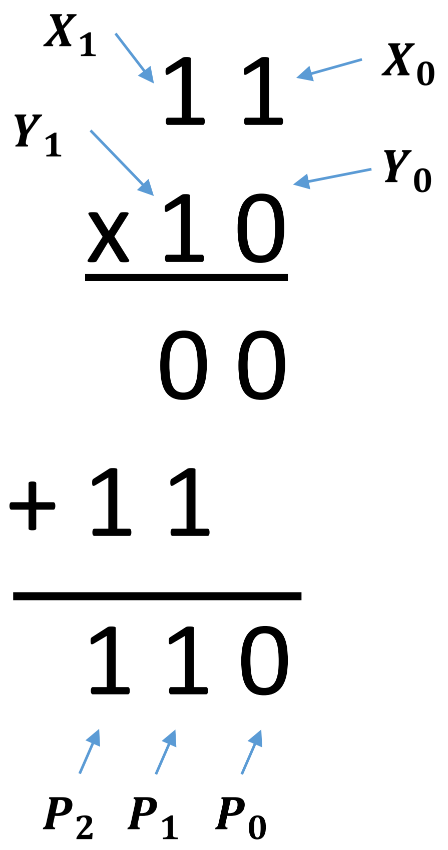
- **Multipliers** (not to be confused with multiplexers) are combinational logic circuits capable of performing multiplication
- Note that the multiplication of two 1-bit numbers is a simple *and* operation

0	1	0	1	X_0
x 0	x 0	x 1	x 1	Y_0
0	0	0	1	

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

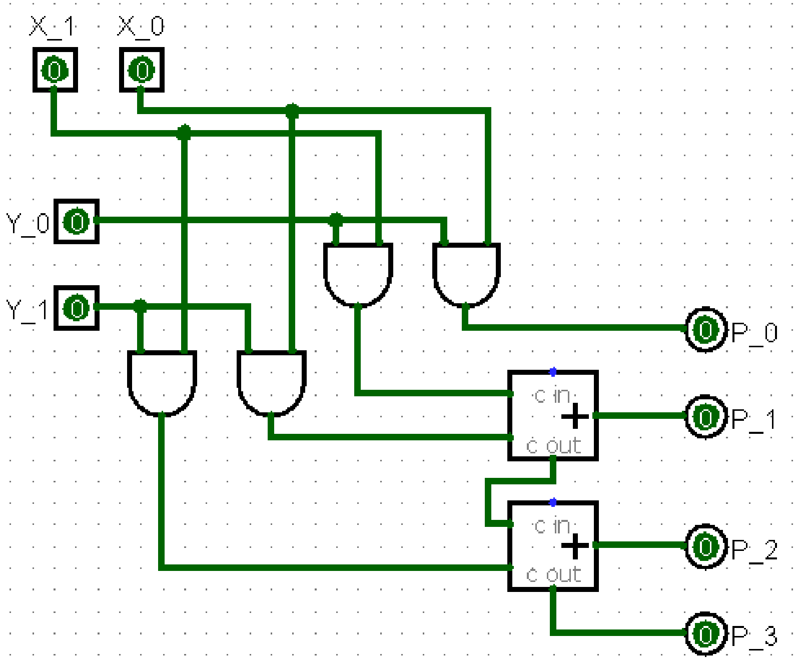
- However, addition will be required when multiplying numbers that are two or more bits.
- We will see how to construct multipliers using full and half adders.
- The largest product of multiplying two, 2-bit numbers is 9:
  - 11 x 11 = 1001 (3 x 3 = 9)
  - Thus, our circuit must have 4 outputs
    -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



Multipliers

2-bit Multiplier Logic Circuit:

- (Uses 2 half adders)



Multipliers

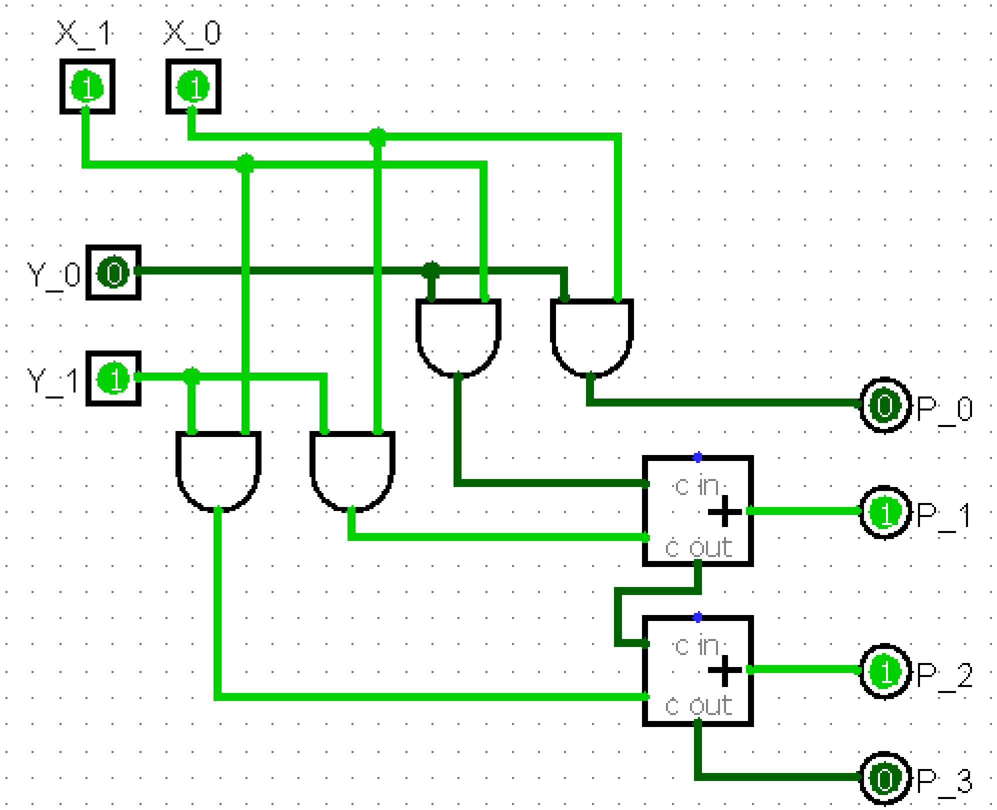
2-bit Multiplier Logic Circuit:

X_1 X_0

Y_1 Y_0

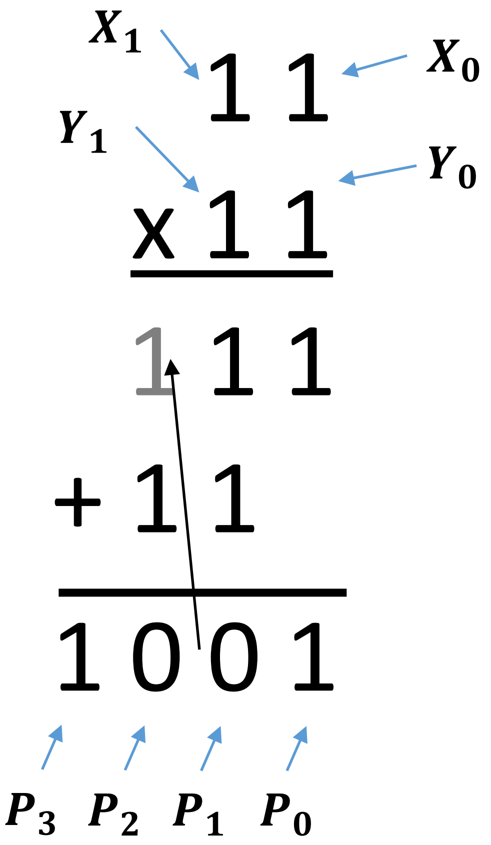
$$\begin{array}{r} 1\,1 \\ \underline{\times\,1\,0} \\ 0\,0 \\ +\,1\,1 \\ \hline 1\,1\,0 \end{array}$$

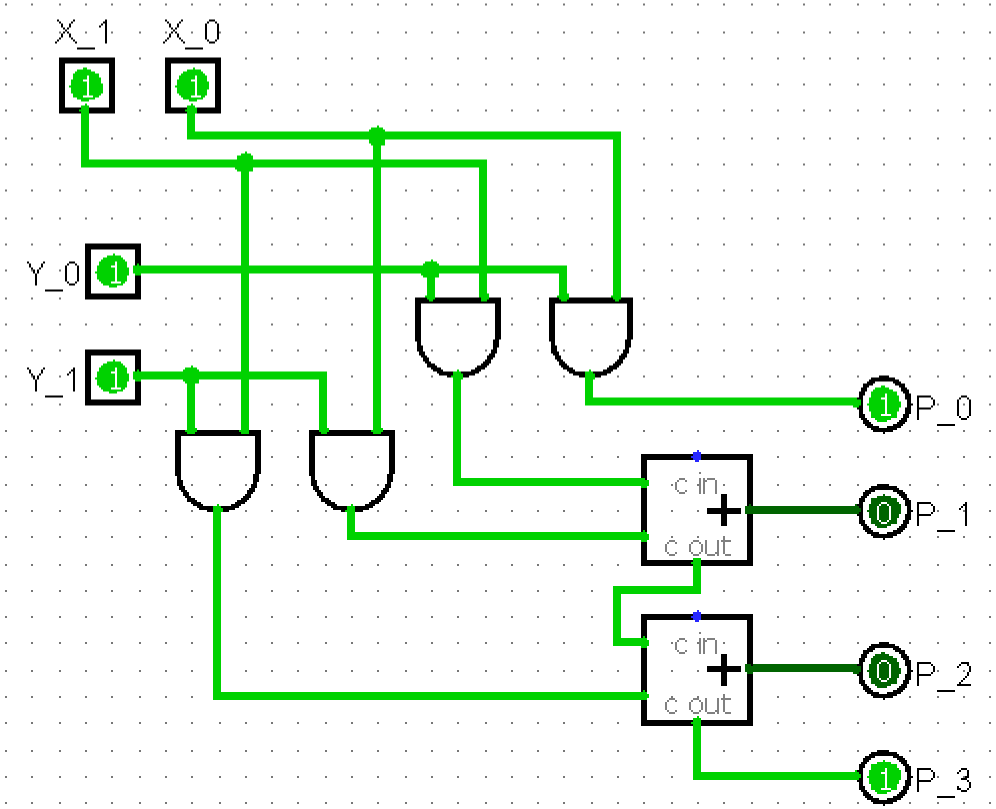
P_2 P_1 P_0



Multipliers

2-bit Multiplier Logic Circuit:

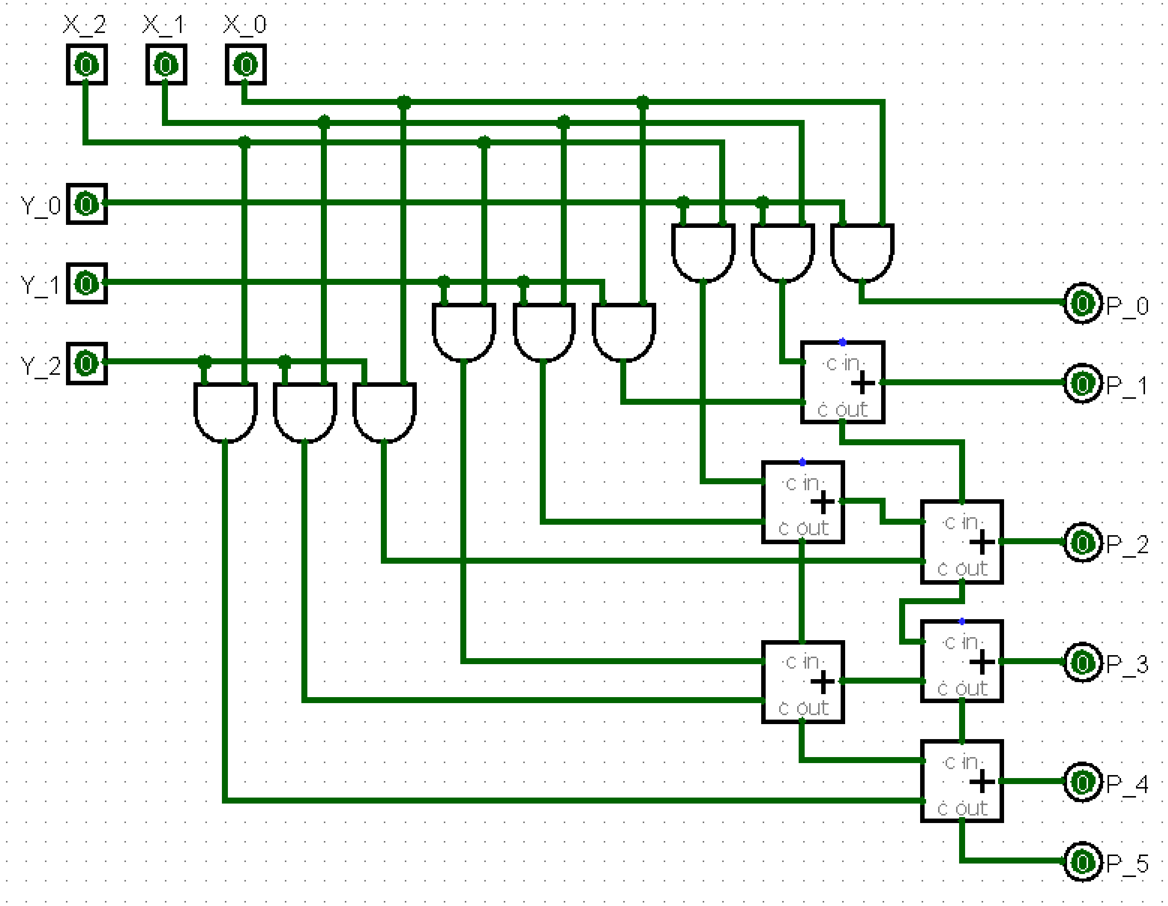




Multipliers

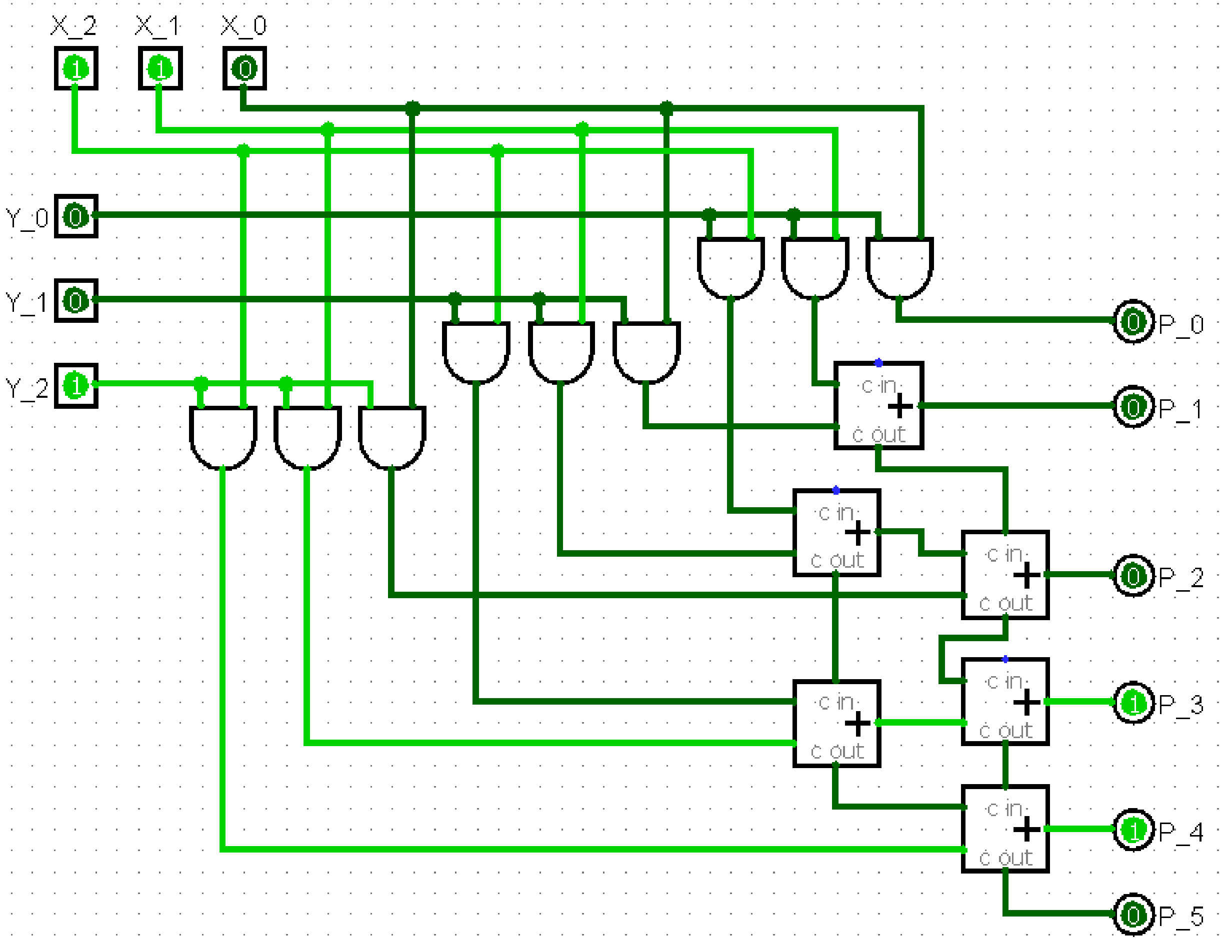
3-bit Multiplier Logic Circuit:

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



Multipliers

3-bit Multiplier Logic Circuit:




```
      1 1 0
    x 1 0 0
    -------
      0 0 0
    0 0 0
+ 1 1 0
-----------
  0 1 1 0 0 0
```

Multipliers

3-bit Multiplier Logic Circuit:

```
      1 1 1
    x 1 1 1
      1 1 1
    1 1 1
+ 1 1 1
 1 1 0 0 0 1
```