Revised: 5/9/2024

Digital Logic IV

Michael C. Hackett
Assistant Professor, Computer Science

Lecture Topics

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

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).

$$
\begin{array}{cccccccc}
 & 0 & & 1 & & 0 & & 1 \\
- & \underline{0} & - & \underline{0} & - & \underline{1} & - & \underline{1} \\
 & 00 & & 01 & & 11 & & 00
\end{array}
$$

X_0

X_1

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

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

B (Borrow) D (Difference)

Half Subtractors

SOP Expressions:

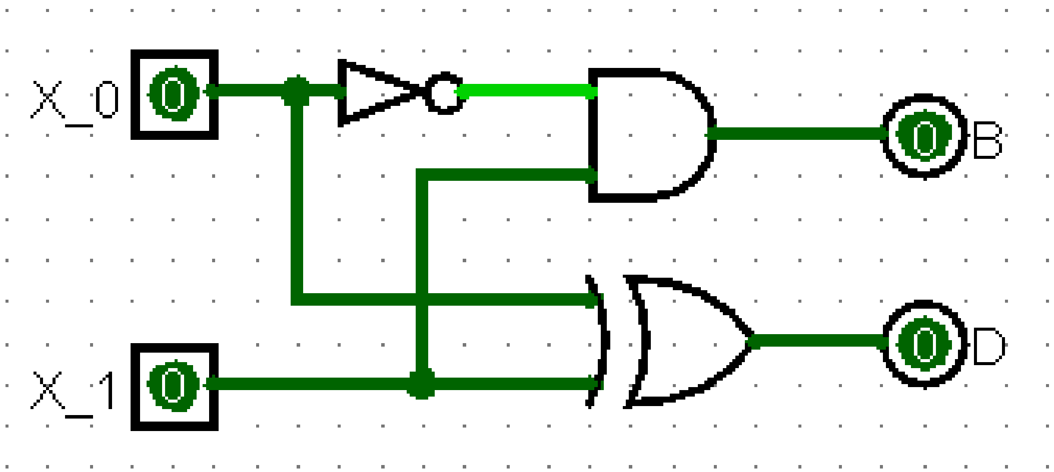
$$B = X_1\overline{X_0}$$

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

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

Half Subtractor

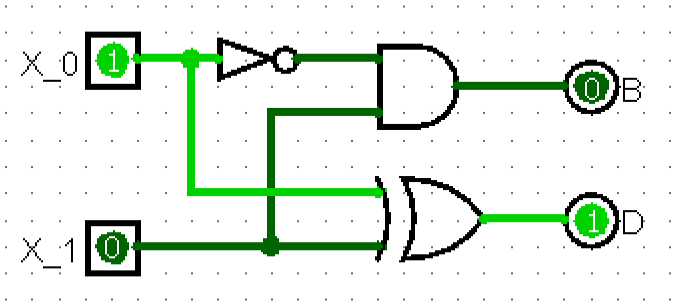
Half Subtractor Logic Circuit:

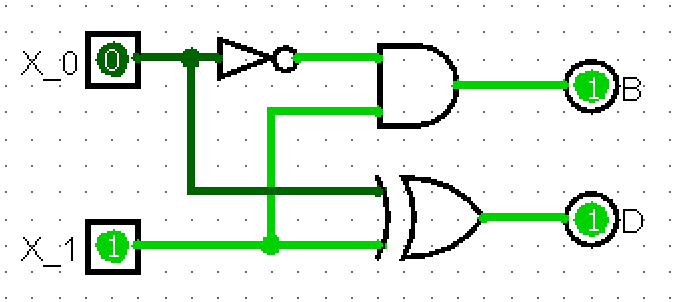


Half Subtractor

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).

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

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

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

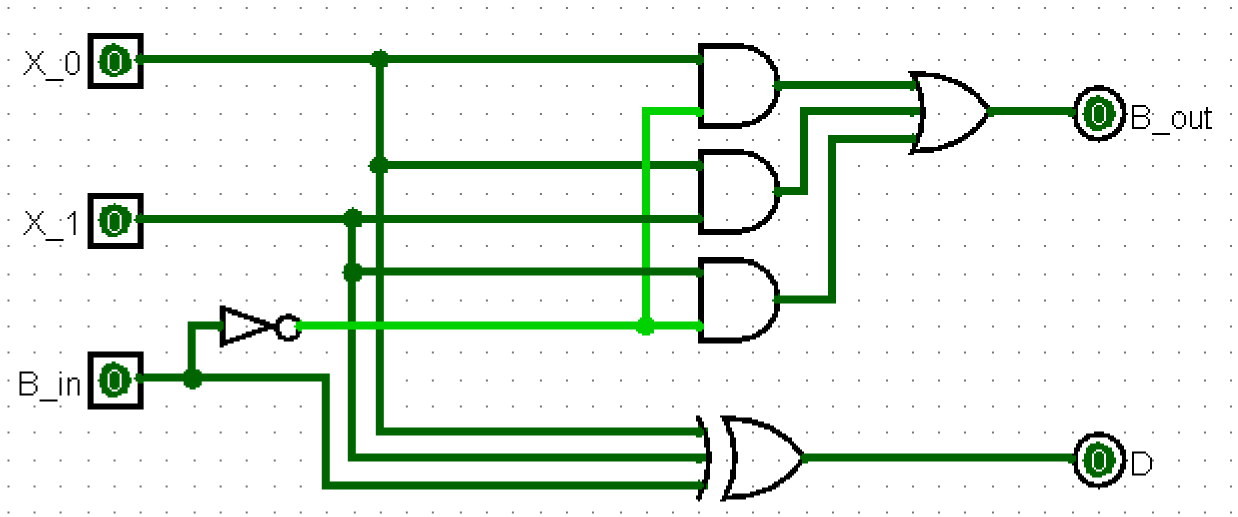
$$D = \overline{B_{IN}}(X_0 \oplus X_1) + B_{IN}(\overline{X_0 \oplus X_1})$$

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

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

Full Subtractor

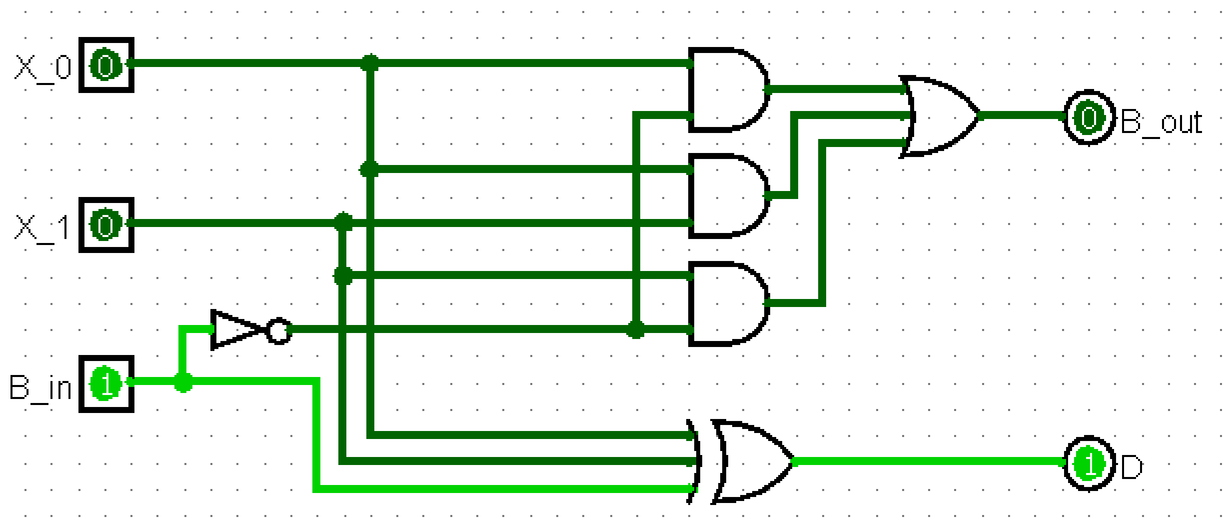
Full Subtractor Logic Circuit:



Full Subtractor

Full Subtractor Logic Circuit:

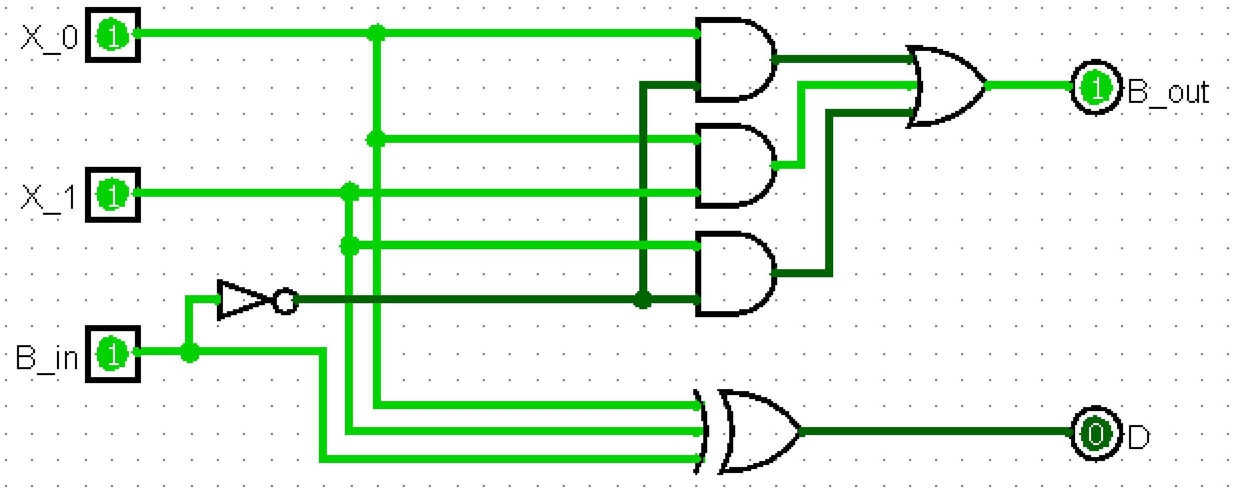
B_{IN}	X_1	X_0	B_{OUT}	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	**0**	**0**	**0**	**1**
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



Full Subtractor

Full Subtractor Logic Circuit:

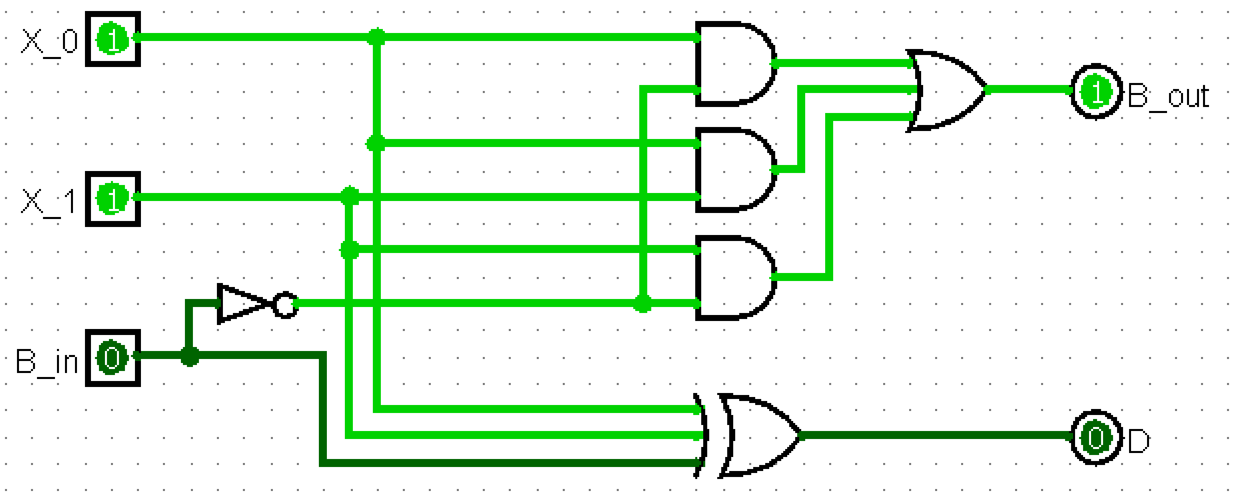
B_{IN}	X_1	X_0	B_{OUT}	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	**1**	**1**	**1**	**1**



Full Subtractor

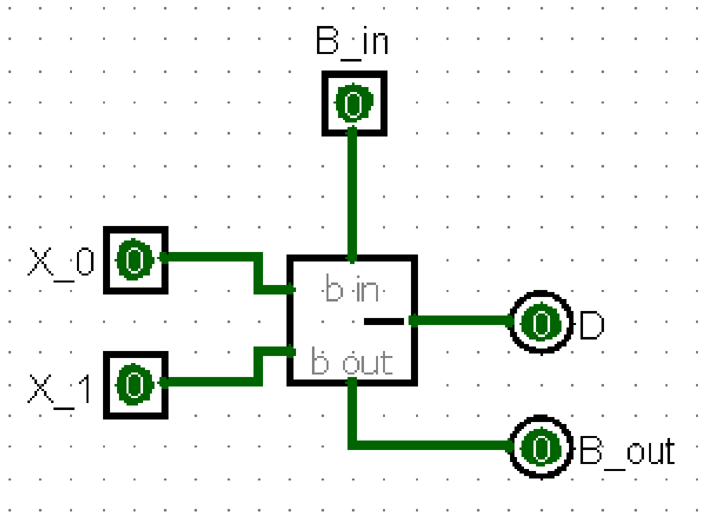
Full Subtractor Logic Circuit:

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

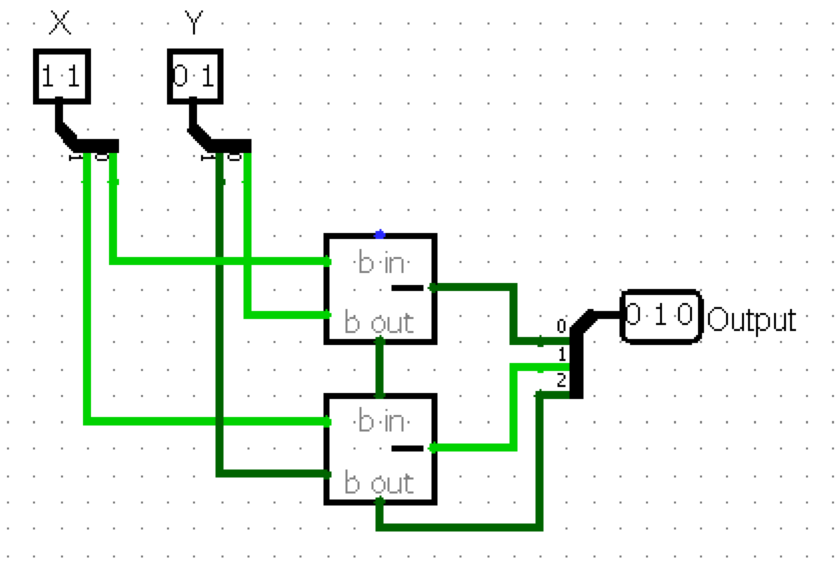
- Abstracted Full Subtractor:



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

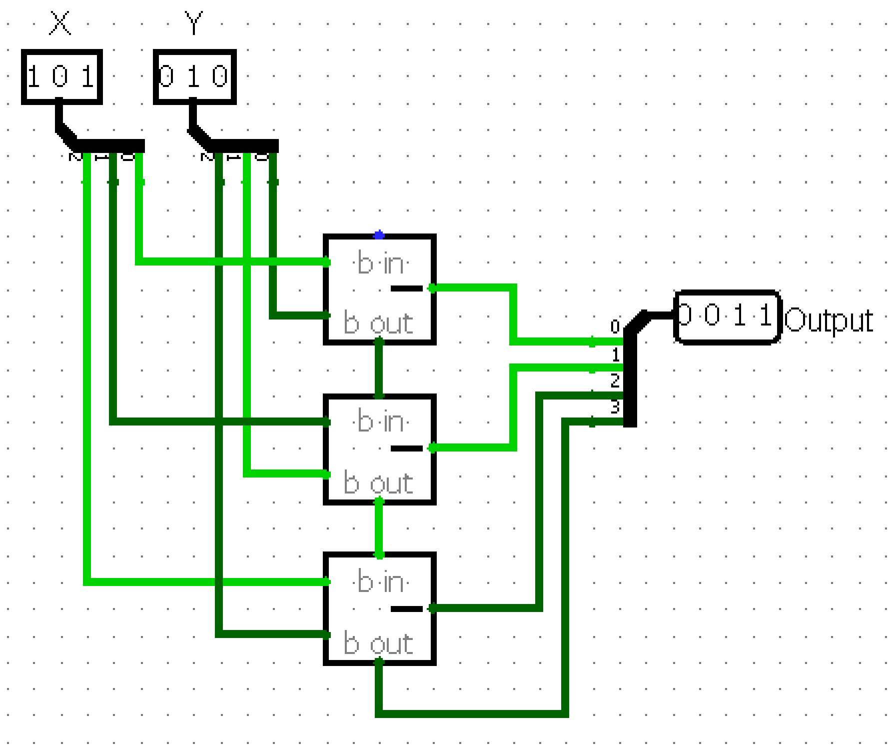
$$\begin{array}{rr} & 1\ 1 \\ - & 0\ 1 \\ \hline & 0\ 1\ 0 \end{array}$$



Full Subtractors

- A 3-bit Subtractor:

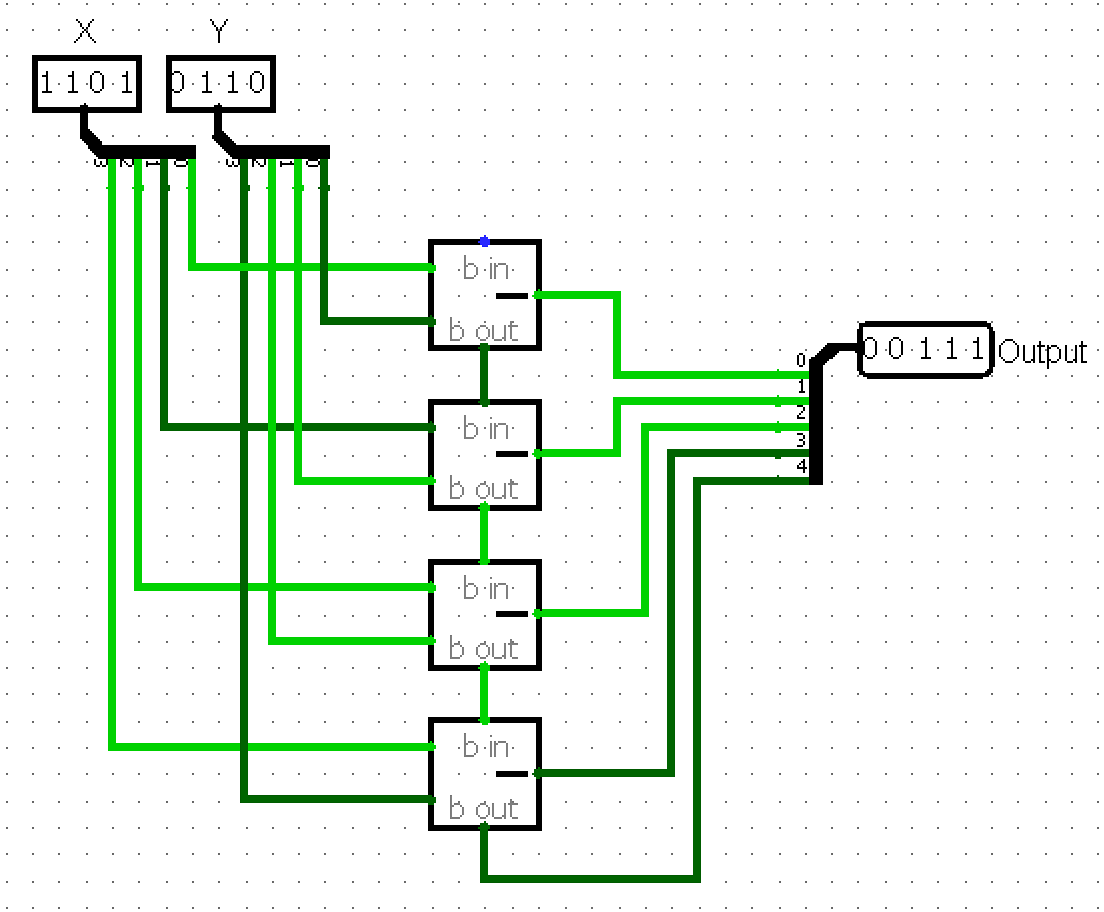
```
    1 0 1
-   0 1 0
---------
  0 0 1 1
```



Full Subtractors

- A 4-bit Subtractor:

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



Multipliers

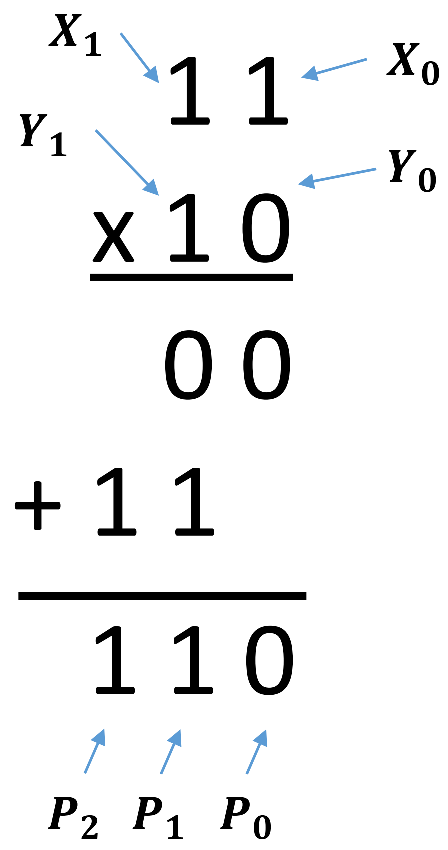
- **Multipliers** (not to be confused with multiplexers) are combinational logic circuits capable of performing multiplication
- Note that the multiplication of two 1-bit numbers is a simple *and* operation

$$
\begin{array}{r} 0 \\ \underline{\times\ 0} \\ 0 \end{array}
\qquad
\begin{array}{r} 1 \\ \underline{\times\ 0} \\ 0 \end{array}
\qquad
\begin{array}{r} 0 \\ \underline{\times\ 1} \\ 0 \end{array}
\qquad
\begin{array}{rl} 1 & \leftarrow X_0 \\ \underline{\times\ 1} & \leftarrow Y_0 \\ 1 & \end{array}
$$

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

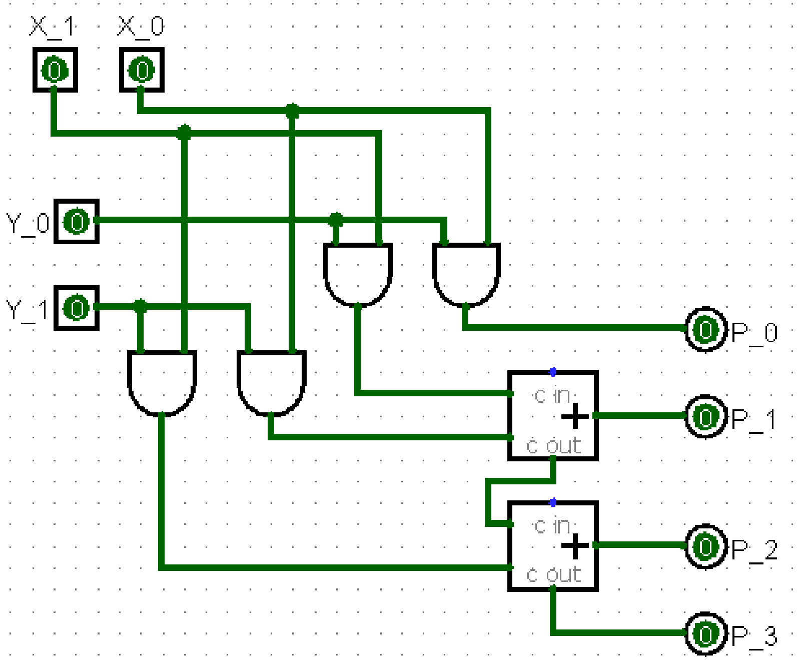
- However, addition will be required when multiplying numbers that are two or more bits.
- We will see how to construct multipliers using full and half adders.
- The largest product of multiplying two, 2-bit numbers is 9:
  - 11 x 11 = 1001 (3 x 3 = 9)
  - Thus, our circuit must have 4 outputs
    -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



Multipliers

2-bit Multiplier Logic Circuit:

- (Uses 2 half adders)



Multipliers

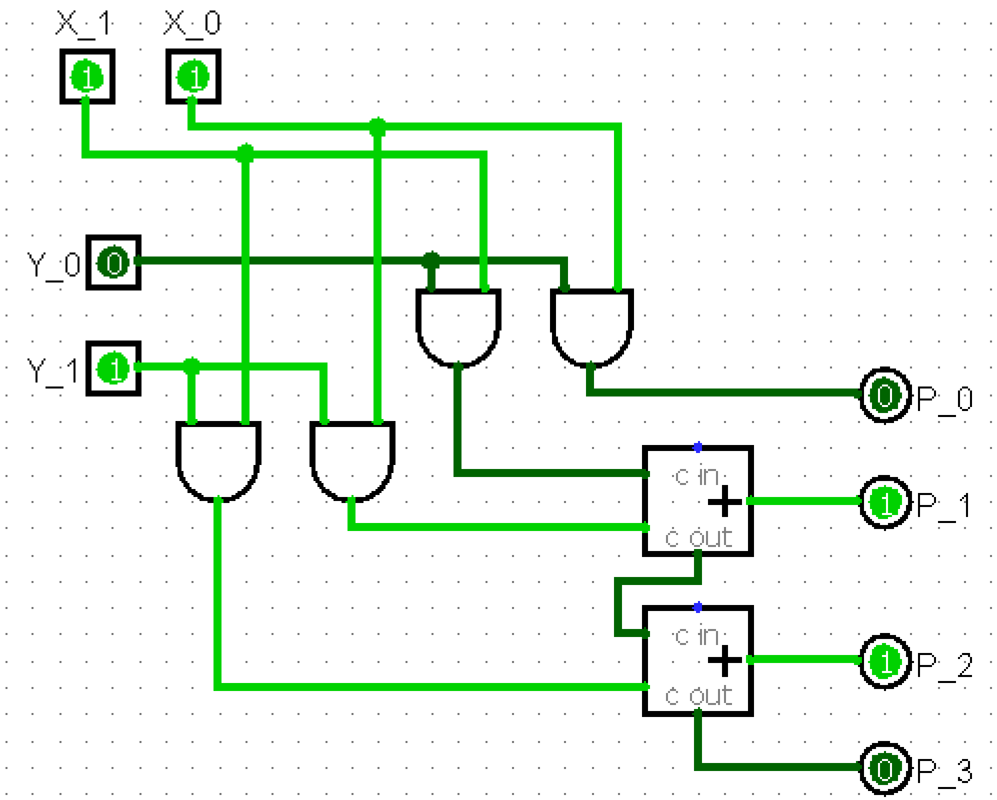
2-bit Multiplier Logic Circuit:

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

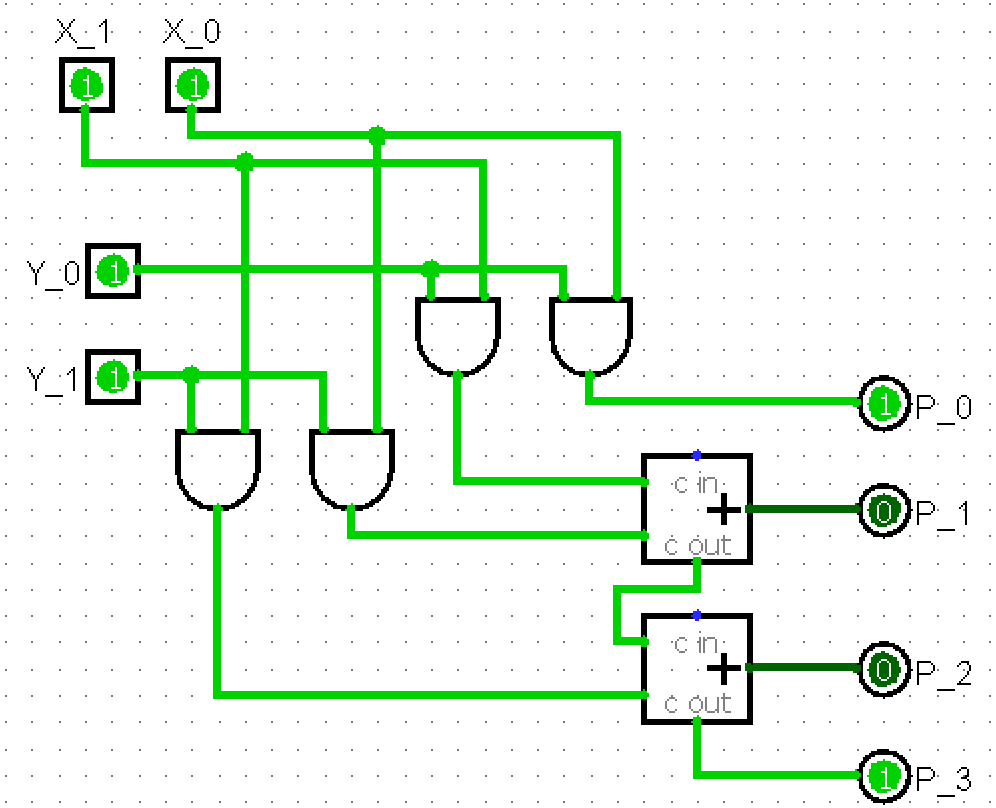
2-bit Multiplier Logic Circuit:

X_1 X_0

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

Y_1 Y_0

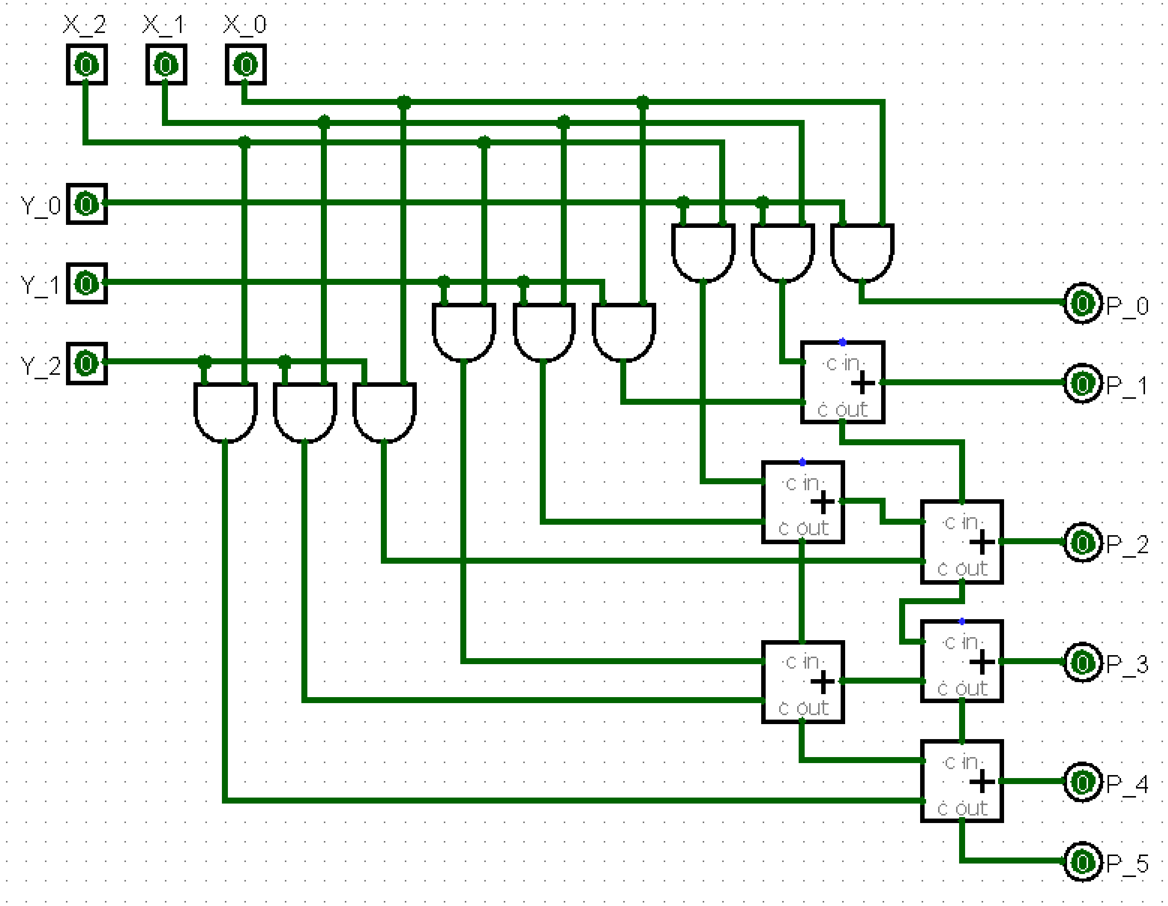
P_3 P_2 P_1 P_0



Multipliers

3-bit Multiplier Logic Circuit:

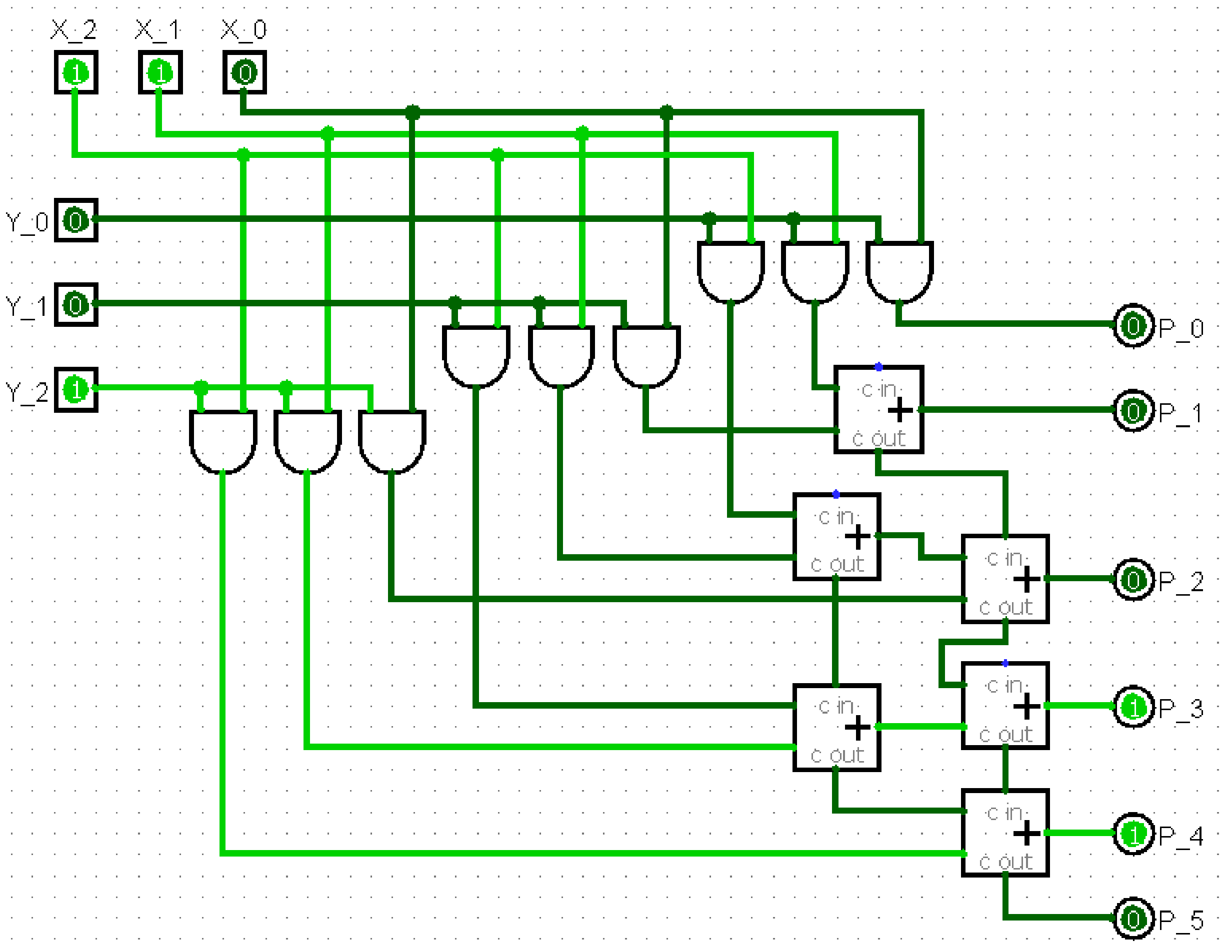
- (Uses 2 half adders)
- (Uses 3 full adders)



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
```