Conversion Between base-2, 8, 16.

↳ easy since all power of 2.

e.g.,

$(4\ 3\ 6\ 5\ 7\ 3\ 4\ 3)_8$

$(100\ 011\ 110\ 101\ 111\ 011\ 100\ 011)_2$

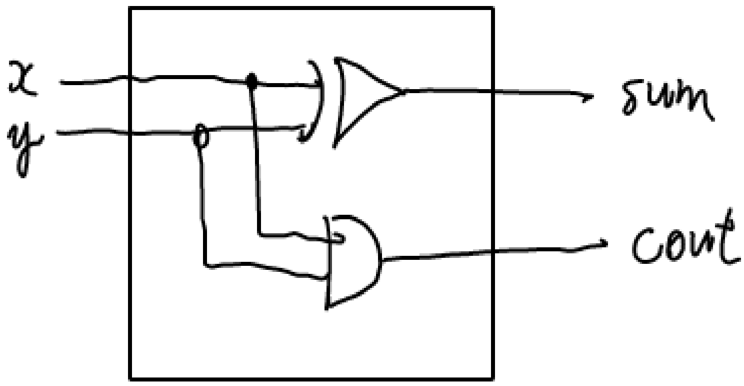
$(\qquad\qquad)_{16}$

Addition

consider adding 2-bits

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

x	y	sum	cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



half adder

Still not enough.

$$\begin{array}{r}01\\+11\\\hline 100\end{array}\qquad\begin{array}{r}1\\+3\\\hline 4\end{array}$$

carry out of one pair of bits is the carry in of the next pair of bits.

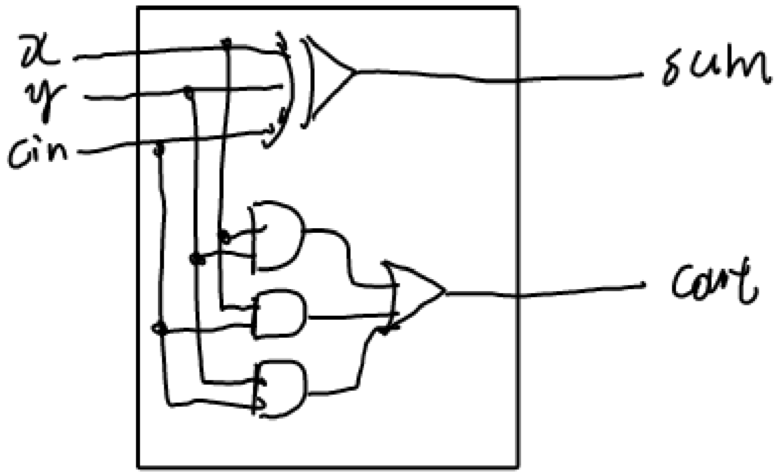
x	y	cin	sum	cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$\text{sum} = x \oplus y \oplus z$

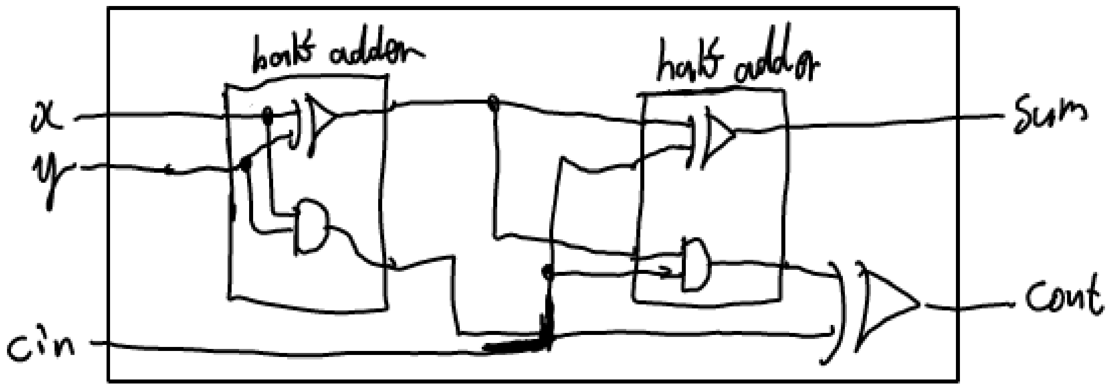
$\text{cout} = y\,\text{cin} + x\,\text{cin} + xy$ ← from kmap for cout.

$= xy + \text{cin}(x+y)$

$= xy + \text{cin}(x \oplus y)$



Full adder



Think about how to add n-bits.
(i.e, we want a circuit that can add unsigned integers represented in n bits.

$$\begin{array}{r} 1010101 \\ +\ 0011001 \\ \hline \underbrace{0}_{\text{cout}}\,\underbrace{1101110}_{\text{answer}} \end{array} \qquad \begin{array}{r} 85 \\ +25 \\ \hline 110 \end{array}$$

$$
\begin{array}{r}
1010101 \\
+\ 1011001 \\
\hline
\underbrace{1}_{\text{cout}}\,\underbrace{0101110}
\end{array}
\qquad
\begin{array}{r}
85 \\
+\ 89 \\
\hline
174
\end{array}
$$

7-bits is too small to represent $(174)_{10}$

1 means OVERFLOW

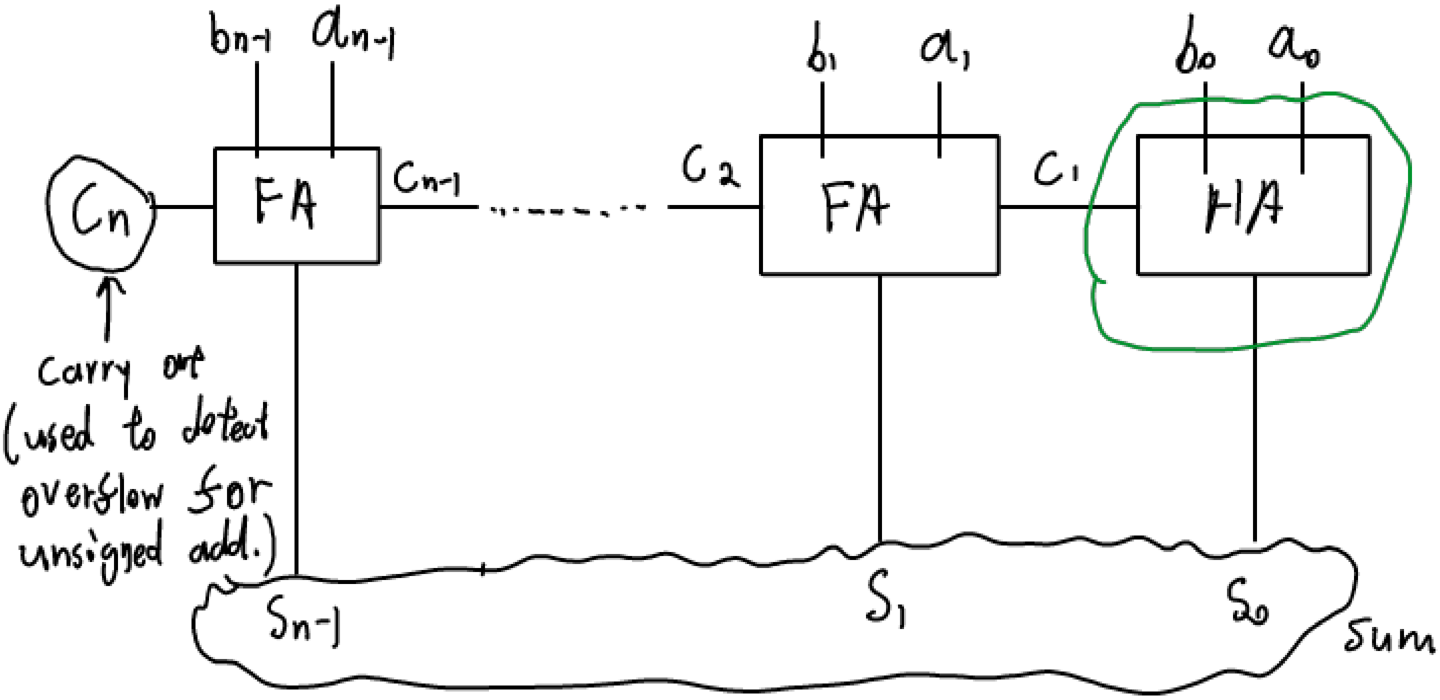
This is the answer in 7-bits.

46 ← wrong

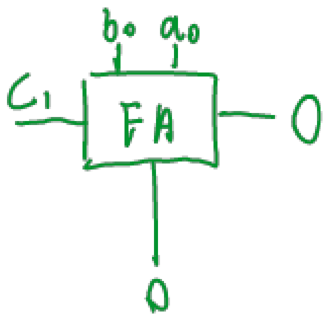
How to build a circuit to add 2 n-bit number?

$$A = (a_{n-1}, a_{n-2}, \ldots, a_2, a_1, a_0)_2$$

$$B = (b_{n-1}, b_{n-2}, \ldots, b_2, b_1, b_0)_2$$



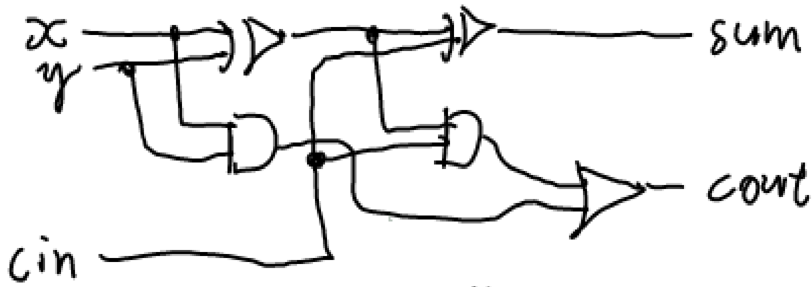
* HA could be replaced as:

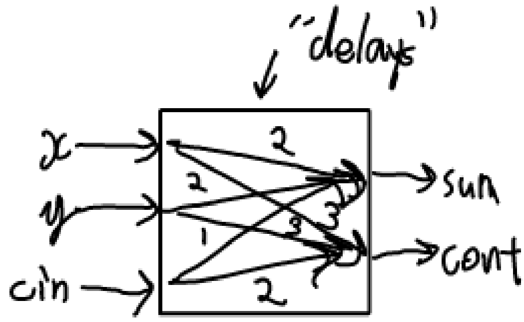


This is called an n-bit ripple adder

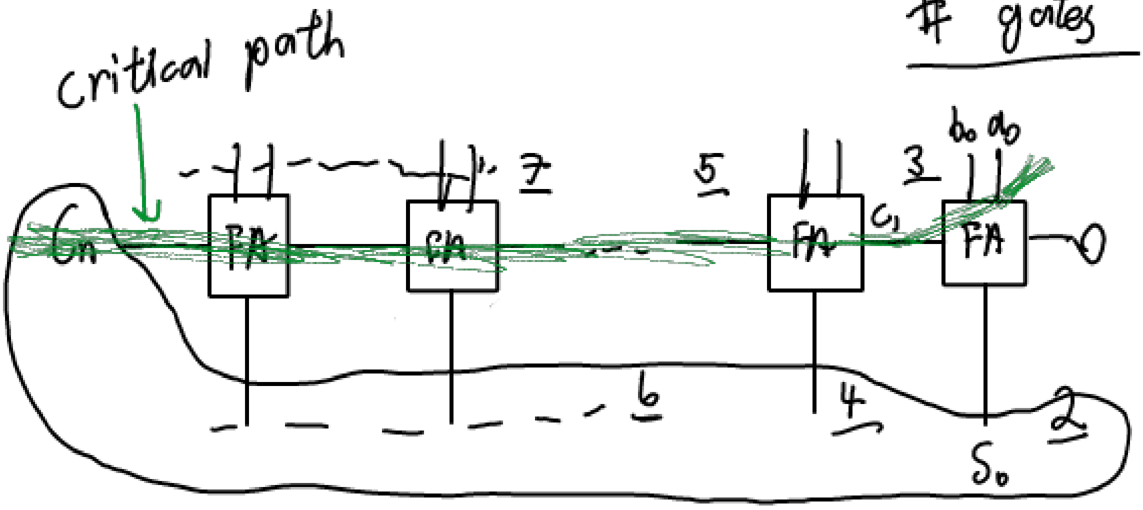
Info. ripples down the carry chain.

performance of a ripple adder?





gates



n-bits

$3+2(n-1)$ units of time before C_n is guaranteed to stop changing.

↑
first adder