Adder

In order to construct a CPU, we need to perform arithmetic and logical operations.
Basic arithmetic operator: addition
What's involved in adding binary numbers?

carry	0	1	1	0		decimal
		0	0	1	1	3
		0	1	1	0	6
result		1	0	0	1	9

In each column:

input: add 2 bits, along with a carry bit from the previous result
output: 1 bit result, 1 bit carry

Half-adder

adds 2 bits (x, y), generates sum (s) and carry (c)
(note that previous carry is being ignored)

Truth table

x	y	s	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

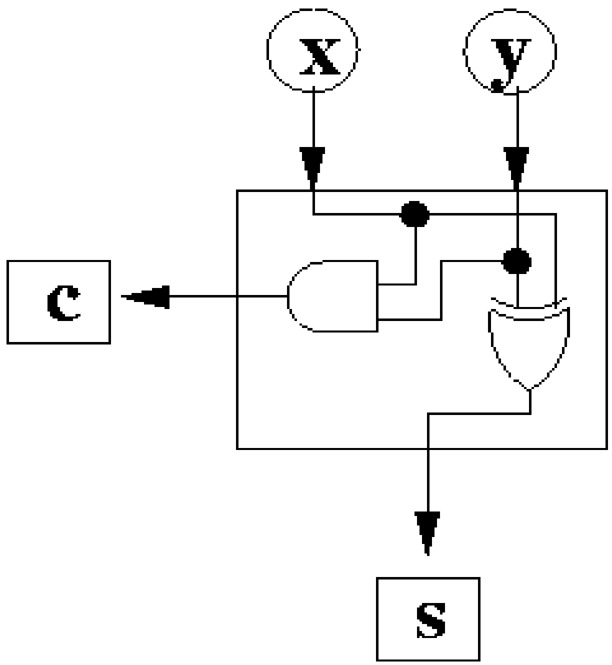
Boolean expressions:

s = \xy + x\y = x XOR y
c = xy

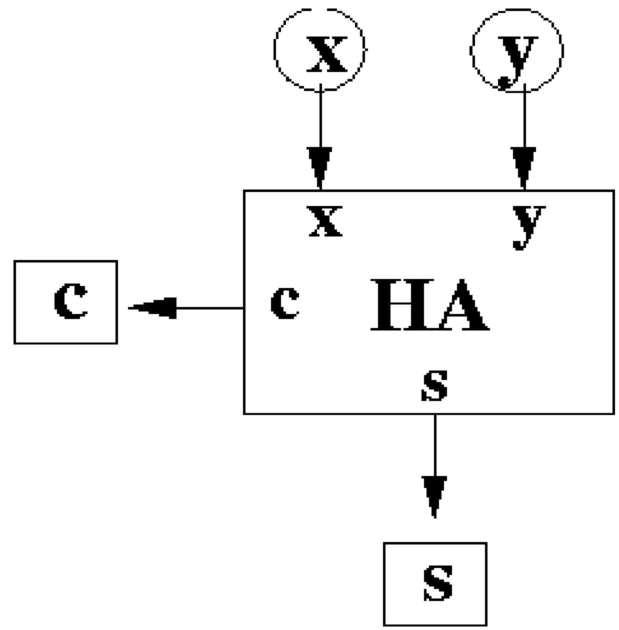
Half adder: circuit

```
s = \xy + x\y = x XOR y
c = xy
```

Circuit:



Black box:



Full adder

In order to perform true addition, we need to use the carry from the previous result

Full adder

Data inputs: x, y, c_{in} (carry in)

Data outputs: s, c_{out} (carry out)

Truth table

x	y	c_{in}	s	c_{out}
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

Boolean expressions

$$s = \backslash x \backslash y c_{in} + \backslash x y \backslash c_{in} + x \backslash y \backslash c_{in} + x y c_{in}$$

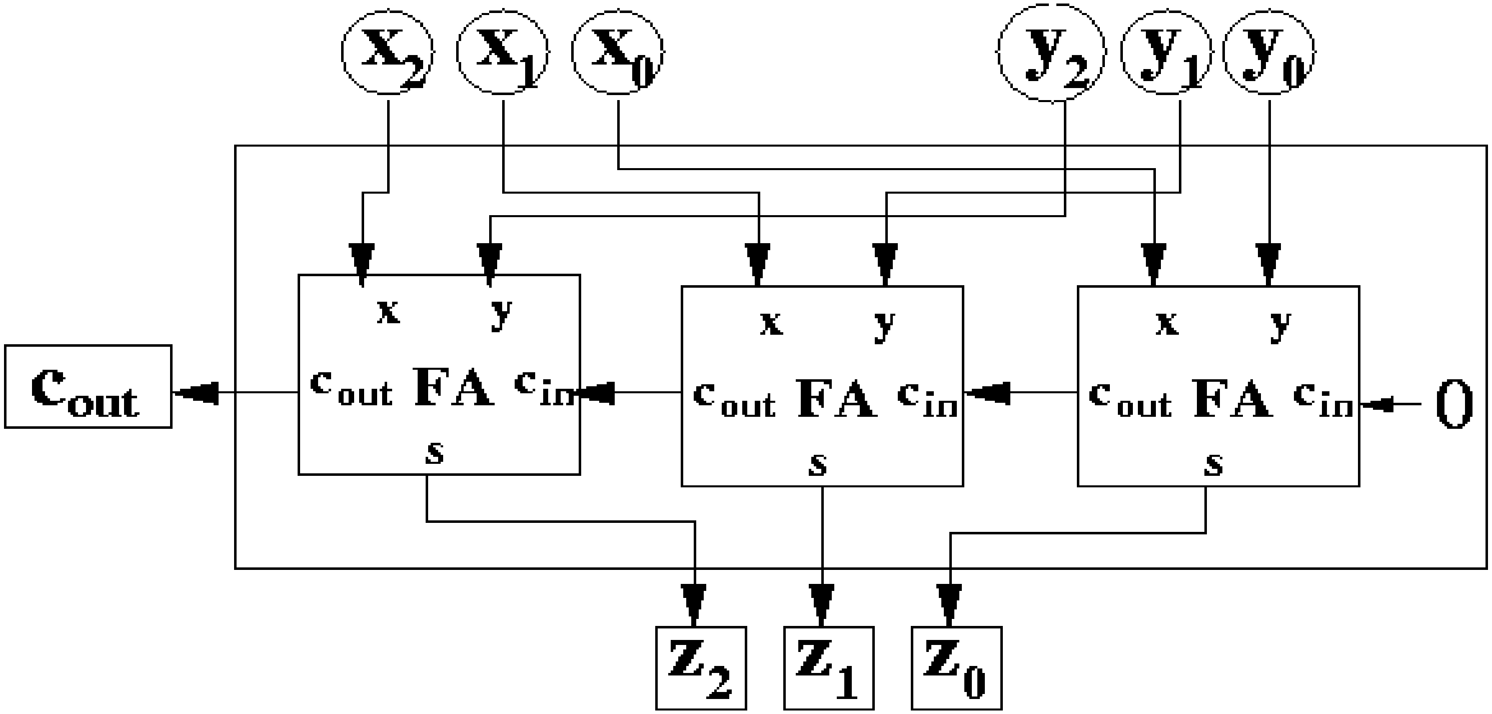
$$c_{out} = \backslash x y c_{in} + x \backslash y c_{in} + x y \backslash c_{in} + x y c_{in}$$

Ripple carry adder

adding k-bit values

ripple-carry adder: combine k 1-bit full adders

3-bit ripple-carry adder



Notice that 0 is hard-wired as carry-in for rightmost full adder (could have used half adder)

Adder delay

Ripple-carry adder is logically correct, but may be slow.

Each circuit requires a finite amount of time to give stable outputs when inputs change.

Circuits are working in parallel, but it takes a finite amount of time before the
carry-in from one circuit is available for the next one to use.

Assume time T for each FA to generate output.
n-bit ripple carry adder has O(n) delay: nT

Speed can be improved by using "carry-lookahead":
compute carries in parallel