Logic

Logical Operators

Symbol	Meaning
~	Not
&	Binary And
&&	Logical And
|	Binary Or
||	Logical Or
«	Left Shift (x2^n)
»	Right Shift (floor division: $\frac{num}{2^n}$)

Logical operators act on statements / prepositions, **Bit-wise** operators act on bit strings.

Karnaugh / K - Maps

Quick way to develop a logical statement that encapsulates a truth table.

1. **Change one bit at a time**
2. only fill cells where output is **1 / T**
3. **group** into powers of 2
4. **remove *unstable* variables** (change w/in the group)
5. **result**
    1. **sum of products (SOP)** like $F = BC + AB + AC$
    2. **product of sums (POS)**
        1. uses 0s instead of 1s
        2. result like $F = (A + B) * (B + C)$

Adders

Half-Adder (HA)

Boolean expressions represent a *sum* (s) and *carry* (cout).

Half-Adders use **XOR** and **AND** gates to "add" two bits.

Full-Adder (FA)

To add more than two bits, we need a full-adder, made from two half-adders.

Inputs: `a`, `b`, and `cin`
Outputs: `sum (s)` and `carry out (cout)`

Symbols for logic gates

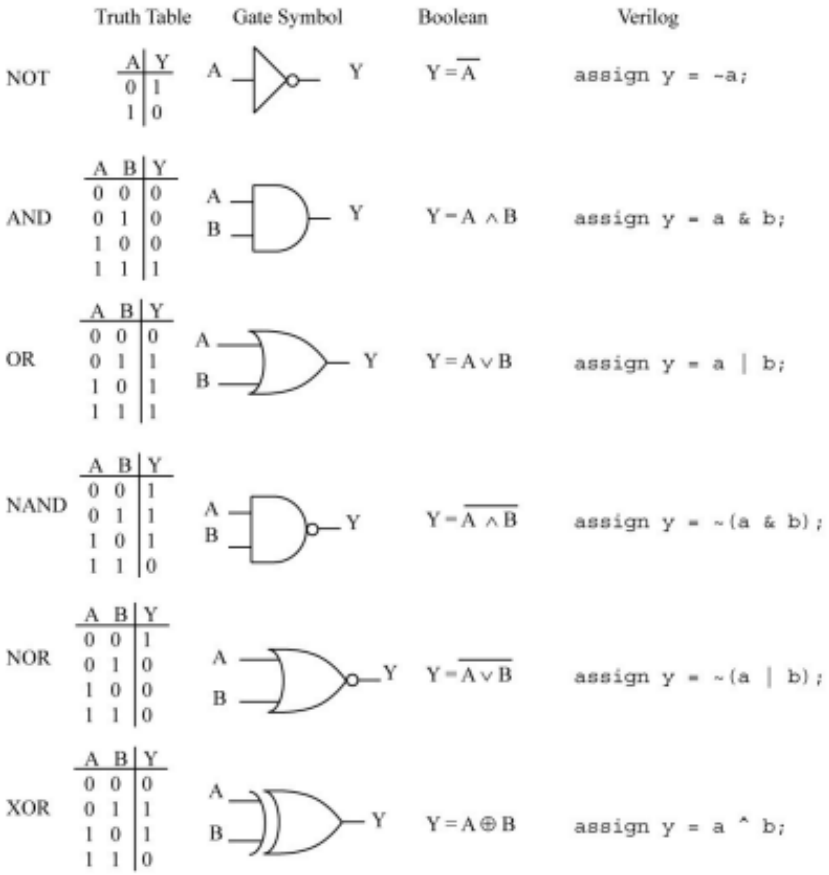


Figure 1: image

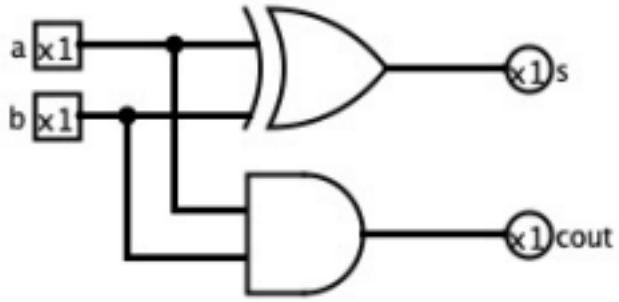


Figure 2: image

Inputs/Ouputs

inputs			outputs	
a	b	cin	s	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

K-Maps

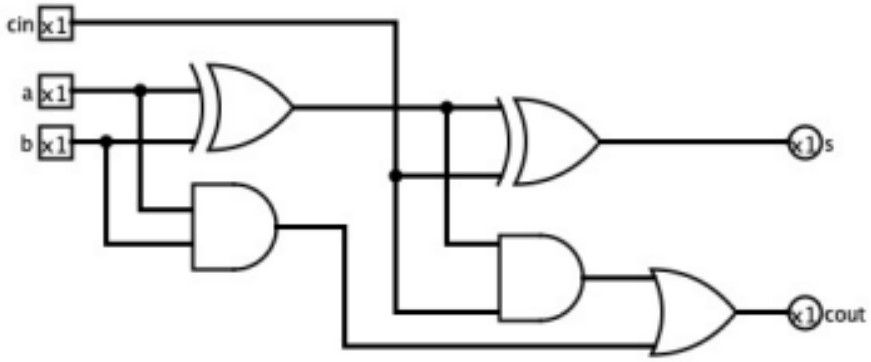
ab \ cin	0	1
00		1
01	1	
11		1
10	1	

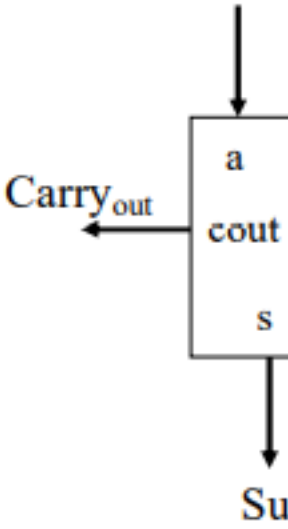
sum (s) k-map

ab \ cin	
00	
01	
11	
10	

co

Full Adder





$s = (\neg a \wedge \neg b \wedge cin) \vee (\neg a \wedge b \wedge \neg cin) \vee (a \wedge \neg b \wedge \neg\, cin) \vee (a \wedge b \wedge cin)$

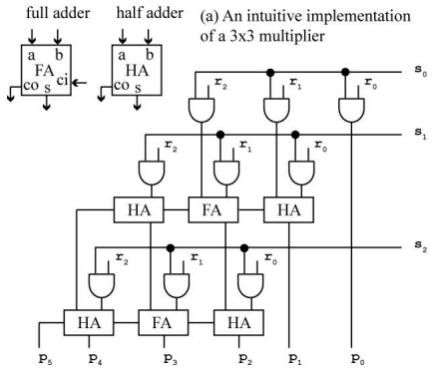
$s = a \oplus b \oplus cin$ (simplified)

$cout = (a \wedge b) \vee (a \wedge cin) \vee (b \wedge cin)$

Multiplication

Binary	Decimal
1 1 1	7
x 1 0 1	x 5
1 1 1	35
0 0 0	
+ 1 1 1	
1 0 0 0 1 1 = 35	

Simple **AND** gate used to copy the top if the bottom is a **1**. Then, we use a combination of HAs and FAs to sum the multiplication.



Notice: we do not need FAs everywhere!