0 Admin
1 Recap
2 Boolean algebra
3 Algebra, TT, Logic design
4 Timing

1 Recap

$Y = A \cdot B$

$Y = A + B$

$Y = \overline{A}$

$Y = \overline{AB}$

2. Boolean Algebra Rules

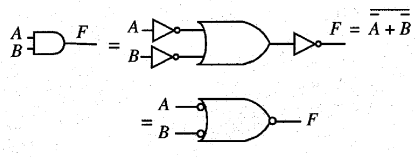
2.1 DeMorgan's Theorem

If $F = AB, \overline{F} = \overline{AB} = \overline{A} + \overline{B} = \overline{\overline{F}} = \overline{\overline{A} + \overline{B}} = \overline{\overline{F}} = \overline{\overline{A} + \overline{B}} = F$

Show by truth table

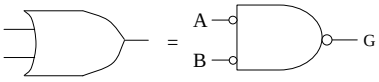
A	B	F	$\overline{A}$	$\overline{B}$	$\overline{A} + \overline{B}$	$\overline{F}$	$\overline{\overline{A} + \overline{B}}$
0	0	0	1	1	1	1	0
0	1	0	1	0	1	1	0
1	0	0	0	1	1	1	0
1	1	1	0	0	0	0	1

2 (cont.): Application of DeMorgan



Corollary: $G = A + B, \overline{G} = \overline{\overline{A} + \overline{B}} = \overline{A} \cdot \overline{B}$

$G = \overline{\overline{A} \cdot \overline{B}}$



2 Boolean Algebra (cont.)

Why useful? "Bubble matching:

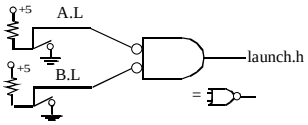
Some signals **asserted** or **active** low

Some signals **asserted** or **active** high

Example: launch rocket if persons A and B both press switch at same time

input: switch A and B asserted *low*

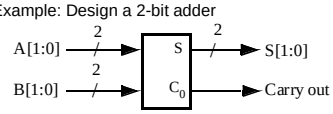
output: launch.H asserted high



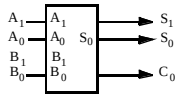
A.L	B.L	Launch.H	
L	L	H	1
L	H	L	0
H	H	L	0
H	H	L	0

3 Algebra, Truth Table, Logic Design

Example: Design a 2-bit adder



4 inputs, 3 outputs



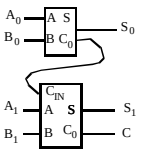
A[1:0]	2 bit #
00	
01	
10	
11	

3 (cont.): Approach 1: Truth Table with 4 Inputs

A_1	A_0	B_1	B_0	S_1	S_0	C

How many lines?
16 (!!!) … too many gates
divide and conquer
partition

Instead, use full adder block



Parallel/serial tradeoff
"ripple carry adder"
each full adder block:
3 inputs, 2 outputs
C_i = Carry In

☞ convention: inputs left, outputs right

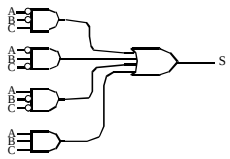
3 Algebra, TT, and Logic Design(cont.)

A	B	C_{IN}	S	C_0	A ⊕ B
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Algebra:

$$S = \overline{A}\,\overline{B}\,C_I + \overline{A}B\cdot\overline{C}_I + A\cdot\overline{B}\cdot\overline{C}_I + A\cdot B\cdot C_I$$

"sum of products form"



3 Algebra, TT, and Logic Design (cont.)

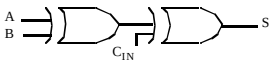
Useful gate: XOR Y = A ⊕ B

A	B	Y
0	0	
0	1	
1	0	
1	1	

$$Y = A\cdot\overline{B} + \overline{A}\cdot B$$

"odd # of 1s" or "if B = 1, invert A"

$$S = \overline{C}_I\cdot[\overline{A}B + A\overline{B}] + C_I[\overline{A}\,\overline{B} + AB]$$
$$= \overline{C}_I\cdot(A\oplus B) + C_I\cdot(\overline{A\oplus B}) =$$
$$\overline{C}_I\cdot X \qquad + C\cdot\overline{X}_I = C_I\oplus(A\oplus B)$$



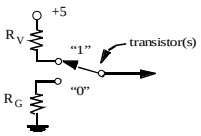
$$C_0 = \overline{A}\cdot B\cdot C_{IN} + A\cdot(B + C_{IN})$$
$$= \overline{A}\cdot B\cdot C_{IN} + A\cdot B + A\cdot C_{IN}$$

Redundant – formal minimization later

4 Timing

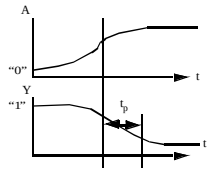
- Two key concepts: what {0,1} and when (timing)
  - How fast can I move or process information?
  - Fundamental limitations: speed of light 3×10^8 m/sec vacuum
    $\sim 2\times10^8$ m/sec conduction
    switching time: capacitance & inductance

4.1 Zeroth order model

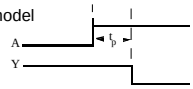


4.2 Voltage level model (more physical)

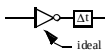
(as seen with oscilloscope)



4.3 Delay model



or



4 Timing (cont.)

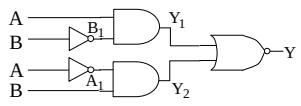
Why big deal?

Dynamic output ≠ static output ("glitches")

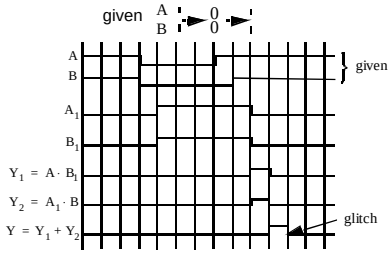
Example: XOR function implemented from AND, OR, INV

idealization: unit delay in each symbol

$$Y = A\cdot\overline{B} + \overline{A}\cdot B$$



4.4 Timing Diagram



$1\oplus1 = 0, 0\oplus0 = 0$

Static behavior fine $1\oplus1 = 0, 0\oplus0 = 0$V

Dynamic behavior glitch → fundamental feature of switching