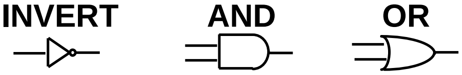
Synthesis of Combinational Logic

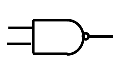
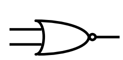
1

*Recap: Combinational Devices**

- Basic combinational devices are
  INVERT AND OR



- Any acyclic interconnection of any number of combinational devices is a combinational device

- Can use  or  to implement any combinational device

* Called logic or Boolean functions after George Boole (circa 1847)

2

How do we specify what we want?

Truth Tables

k inputs — output

2^k rows

A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

input combination

A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	1	0	1

Assume unspecified input combinations produce 0

Number of Distinct truth tables with k inputs is 2^{2^k}

3

Truth Tables and Logic Equations

A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	1	0	1

F = 1 if A=0 AND B=0 AND C=0 OR
A=0 AND B=0 AND C=1 OR
A=0 AND B=1 AND C=0 OR
A=0 AND B=1 AND C=1 OR
A=1 AND B=1 AND C=0
F = 0 otherwise

Write as a logic equation:

$$F = \overline{A}\cdot\overline{B}\cdot\overline{C} + \overline{A}\cdot\overline{B}\cdot C + \overline{A}\cdot B\cdot\overline{C} + \overline{A}\cdot B\cdot C + A\cdot B\cdot\overline{C}$$

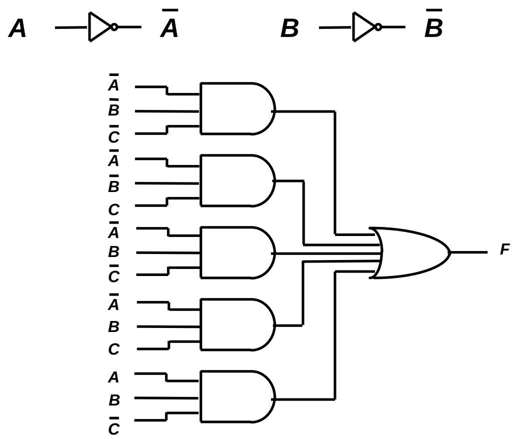
or

$$F = \overline{A}\ \overline{B}\ \overline{C} + \overline{A}\ \overline{B}\ C + \overline{A}\ B\ \overline{C} + \overline{A}\ B\ C + A\ B\ \overline{C}$$

4

Logic Equations and AND-OR Circuits

$$F = \bar{A}\,\bar{B}\,\bar{C} + \bar{A}\,\bar{B}\,C + \bar{A}\,B\,\bar{C} + \bar{A}\,B\,C + A\,B\,\bar{C}$$

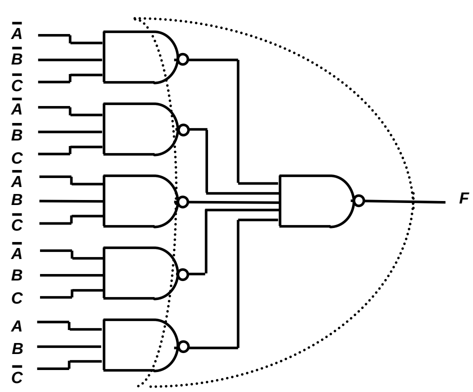


Called a two-level circuit (ignore input inverters)

5

NAND-NAND Circuit

$$F = \bar{A}\,\bar{B}\,\bar{C} + \bar{A}\,\bar{B}\,C + \bar{A}\,B\,\bar{C} + \bar{A}\,B\,C + A\,B\,\bar{C}$$



DeMorgan's law:

$$f_1 + f_2 + f_3 = \overline{\bar{f_1} \cdot \bar{f_2} \cdot \bar{f_3}}$$

Number of transistors in above circuit: 40 (ignoring input inverters)

6

Identities for Logic Functions

Given logic functions f, g, h

$$f \cdot f = f \qquad f + f = f$$
$$f \cdot 1 = f \qquad f + 1 = 1$$
$$f \cdot 0 = 0 \qquad f + 0 = f$$
$$f \cdot \overline{f} = 0 \qquad f + \overline{f} = 1$$

Distributive laws

$$f \cdot (g + h) = f \cdot g + f \cdot h$$
$$(f + g) \cdot (f + h) = f + g \cdot h \quad \textit{WHY?}$$

7

Simplifying Logic Equations

Apply identities to simplify equation for F

$$F = \underbrace{\overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,C}_{} + \underbrace{\overline{A}\,B\,\overline{C} + \overline{A}\,B\,C}_{} + A\,B\,\overline{C}$$

$$= \underline{\qquad\quad} + \underline{\qquad\quad} + A\,B\,\overline{C}$$

$$= \underline{\qquad\quad} + A\,B\,\overline{C}$$

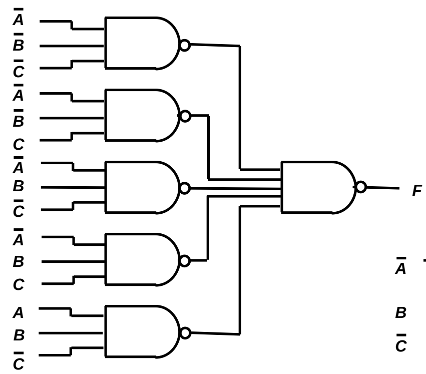
$$= \underline{\qquad\qquad\qquad}$$

8

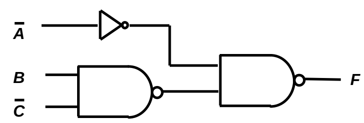
6004 Spring 1998: L05

Comparison

$F = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,C + \overline{A}\,B\,\overline{C} + \overline{A}\,B\,C + A\,B\,\overline{C}$



$F = \overline{A} + B\,\overline{C}$



40 Transistors

8 Transistors

9

Synthesis Flow

Truth Table

Logic Equation

Combinational Circuit

ONE

F = AB + BC ..

F = B (A + C) ..

...

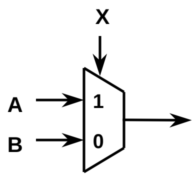
AND-OR

NAND-NAND

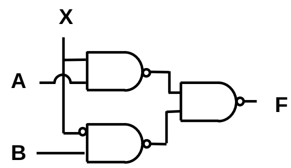
NOR-NOR

10

Multiplexors and Decoders

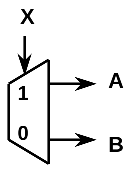


2-1 MULTIPLEXOR (MUX)



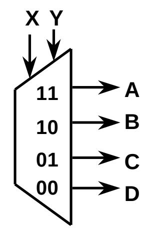
F = A if X = 1, else F = B

1-BIT DECODER



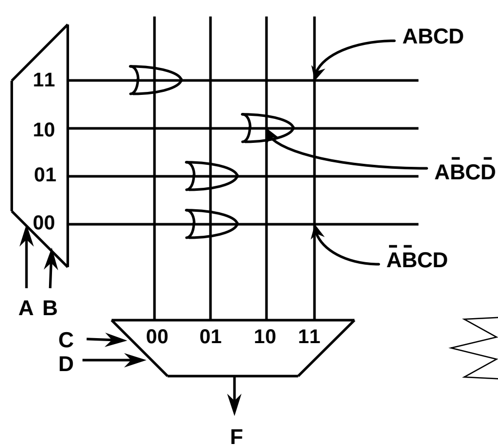
A = 1 if X = 1, else 0
B = 1 if X = 0, else 0

2-BIT DECODER



11

Read-Only Memory (ROM)



ROM size dependent only on number of inputs to the function.

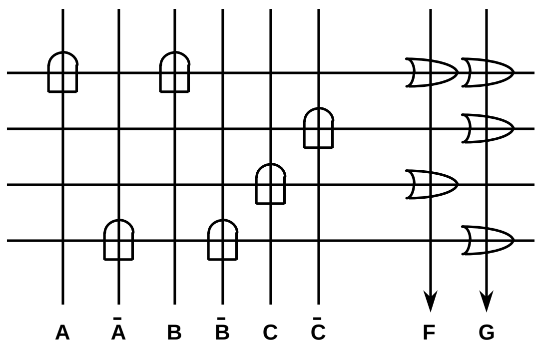
$F = AB\overline{C}\overline{D} + \overline{A}B\overline{C}D + \overline{A}\overline{B}\overline{C}D + A\overline{B}C\overline{D}$

12

6004 Spring 1998: L05

Programmable Logic Array (PLA)

PLA exploits structure of expression.



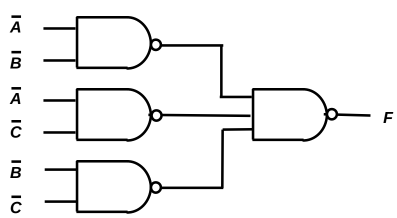
$F = AB + C$

$G = AB + \bar{C} + \bar{A}\bar{B}$

13

Simplistic Viewpoint

- **All we need are two-level circuits**



- **Can implement all logic functions**
- **Two-level circuits are fast because transitions only propagate through two levels of logic (ignoring input inverters)**

14

Parity (XOR) Functions

Count the number of 1's in the inputs and produce a 1 if odd

A	*B*	*P_2*
0	*1*	*1*
1	*0*	*1*

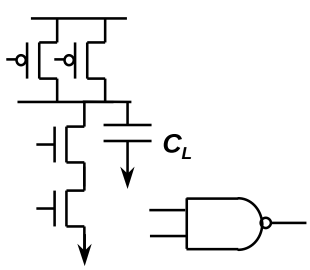
A	*B*	*C*	*P_3*
0	*0*	*1*	*1*
0	*1*	*0*	*1*
1	*0*	*0*	*1*
1	*1*	*1*	*1*

Smallest AND-OR circuit for P_n will contain 2^{n-1} AND gates

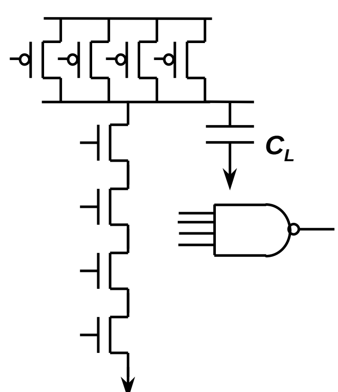
Similar increase for adders, subtractors, multipliers, and most arithmetic functions

15

Circuit Delay: Is Fanin Free?



Time constant for C_L discharge = 2 R_{ON} C_L

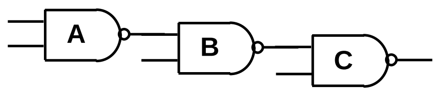


Time constant for C_L discharge = 4 R_{ON} C_L

16

Circuit Delay: Synopsis

- Propagation delay accumulates with each level

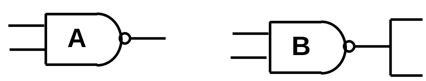


$t_{pd\ A} + t_{pd\ B} + t_{pd\ C}$

- Increased fanin results in increased delay

$t_{pd\ B} > t_{pd\ A}$

- Increased fanout results in increased load capacitance and therefore increased delay



$t_{pd\ B} > t_{pd\ A}$

17

Arithmetic Functions

- Truth tables don't work, need an alternative strategy

HIERARCHICAL BIT-SLICED DESIGN

Given a parameterizable arithmetic function F_n

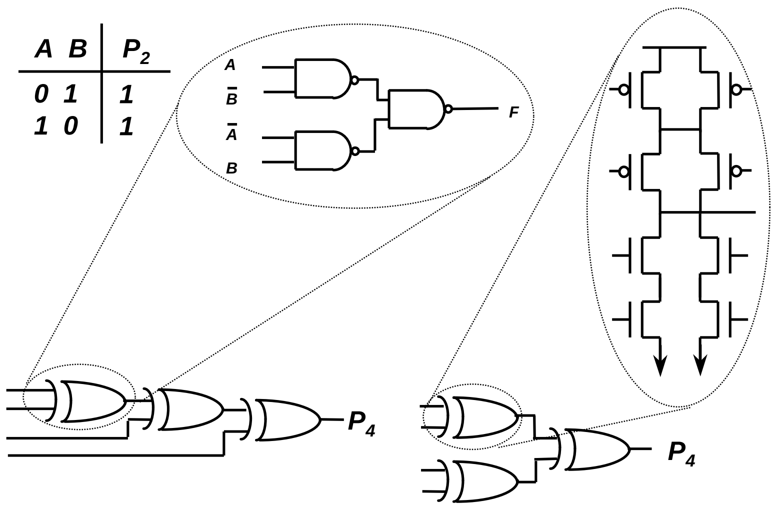
Design F_i , $1 \leq i \leq 4$ using truth-table-based method

Compose individual F_i's to obtain F_n , $n >> i$

18

Revisiting Parity

A	*B*	P_2
0	*1*	*1*
1	*0*	*1*



Binary Addition

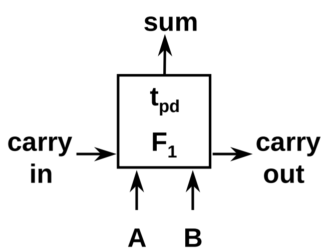
- **Adding two binary numbers**

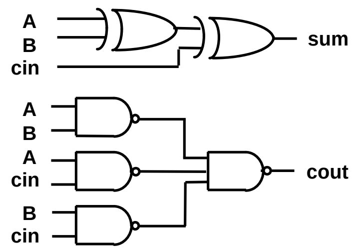
$$\begin{array}{r} 1111 \\ 0101 \\ \hline 10100 \end{array}$$

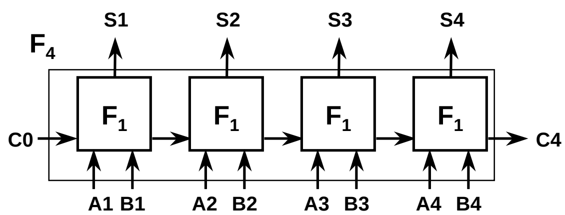
- **We may need to add three bits**
  - **2 bits in same position for the two numbers**
  - **Carry bit from previous position**

A	B	cin	cout	sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Ripple Carry Adder Implementation







Delay at C4 = 4 t_{pd}

21