COSC 243

Introduction to Logic
And
Combinatorial Logic

Overview

- This Lecture
  - Introduction to Digital Logic
    - Gates
    - Boolean algebra
  - Combinatorial Logic
  - Source: Chapter 20 (8th ed. online) or Appendix B (7th ed.)
  - Source: J.R. Gregg, *Ones and Zeros*
- Next Lecture
  - Sequential Logic
  - Source: Chapter 20 (8th ed. online) or Appendix B (7th ed.)
  - Source: Lecture notes

Internal Assessment Announcement

- Data representation test
- During tutorial time on 15-16 March
- 9.5% of final mark

A Bit of History

- Aristotle (384-322 B.C.)
- George Boole (1815-1864)
- Sought to characterise all of human intelligence in precise symbolic form
- Symbolic logic

Introduction to Digital Logic

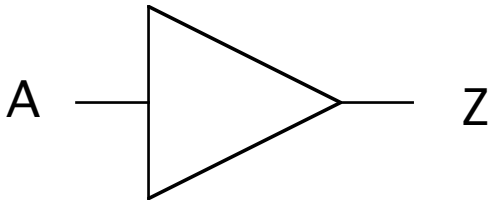
- What do we mean by digital?
- What are the advantages?
- TRUE = HIGH = ON = 1
- FALSE = LOW = OFF = 0

Basic Logic Gates

- Buffer
- Inverter or NOT
- AND
- NAND
- OR
- NOR
- EOR or XOR

Buffer

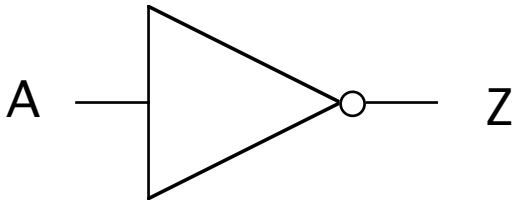
- Normally used to drive several gates or devices requiring higher drive requirements



A	Z
0	
1	

Inverter or NOT

$$Z = \overline{A}$$

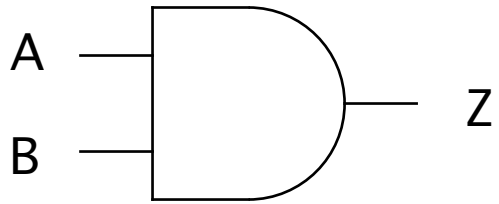


A	Z
0	
1	

AND

$Z = A \cdot B$

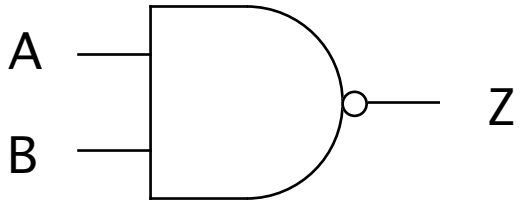
Can be extended: $Z = A \cdot B \cdot C \cdot \ldots$



A	B	Z
0	0	
0	1	
1	0	
1	1	

NAND

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

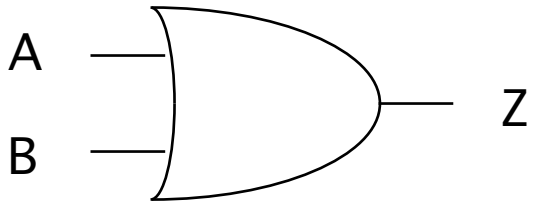


A	B	Z
0	0	
0	1	
1	0	
1	1	

OR

Z = A + B

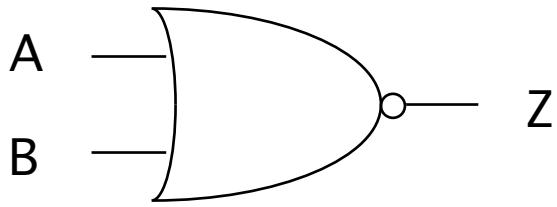
Can be extended Z = A + B + C + . . .



A	B	Z
0	0	
0	1	
1	0	
1	1	

NOR

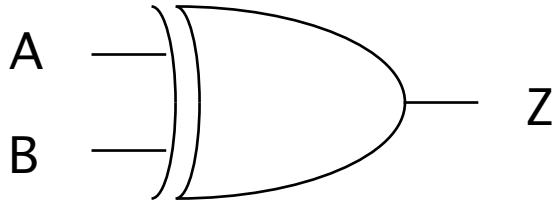
$$Z = \overline{A + B}$$



A	B	Z
0	0	
0	1	
1	0	
1	1	

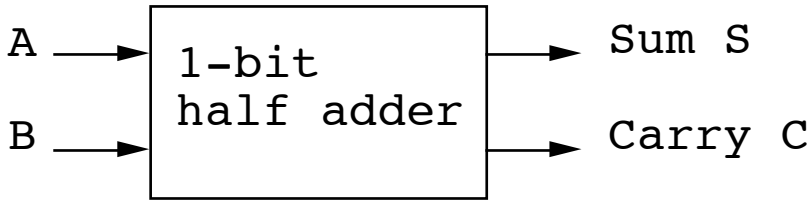
EOR or XOR

$$Z = A \oplus B$$



A	B	Z
0	0	
0	1	
1	0	
1	1	

1-Bit Half Adder



1-Bit Half Adder (cont)

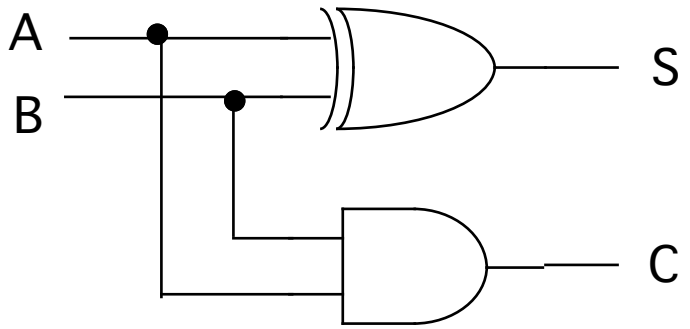
A	B	S	C
0	0		
0	1		
1	0		
1	1		

1-Bit Half Adder (cont)

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$S = \overline{A} \bullet B + A \bullet \overline{B}$$
$$= A \text{ xor } B$$

$$C = A \bullet B$$



Boolean Algebra

Identity Proposition

(Special Properties of 0 and 1)

$$A + 0 = A$$

$$A + 1 = 1$$

$$A \cdot 1 = A$$

$$A \cdot 0 = 0$$

Inverse Proposition

$$A + \overline{A} = 1$$

$$A \cdot \overline{A} = 0$$

Boolean Algebra (cont)

Commutative Proposition

$$A \cdot B = B \cdot A$$

$$A + B = B + A$$

Distributive Proposition

$$A \cdot (B + C) = A \cdot B + A \cdot C$$

$$A + B \cdot C = (A + B) \cdot (A + C)$$

Law of Involution

$$\overline{\overline{A}} = A$$

Boolean Algebra (cont)

Simplification Theorem

$$A + A \cdot B = A$$

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

De Morgan's Theorem

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

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

Proof by Truth Table

Use first version of De Morgan's theorem

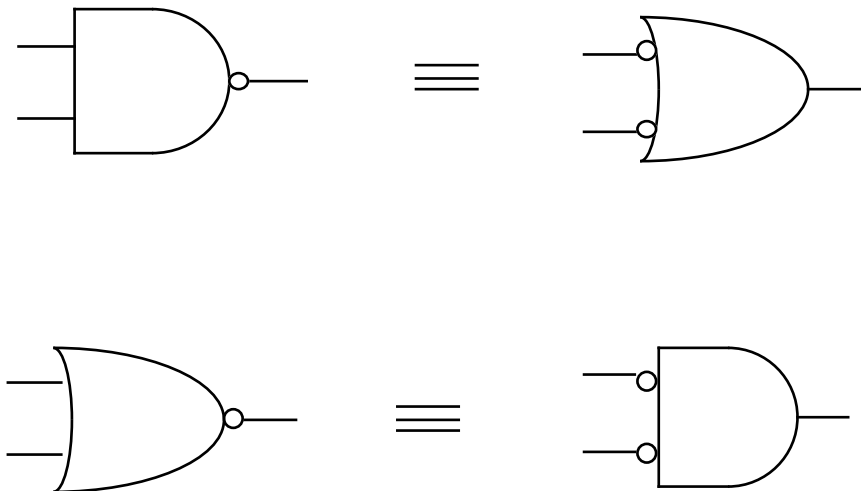
$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

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

NAND/NOR Logic

- Equations of 1-bit half adder requires 2 AND gates, 1 OR gate, and 2 NOT gates
- Inefficient since IC's contain groups of a single type of gate
- De Morgan's theorem states it is possible to convert a NOR into a NAND and vice versa

Equivalent Ways of Drawing NAND/NOR Gates



Introduction to Combinatorial Logic

- Combinatorial logic circuit - one whose outputs are dependent only on the inputs
- Assume the outputs respond immediately.
- In real circuits, propagation delays must be considered

Introduction to Combinatorial Logic (cont)

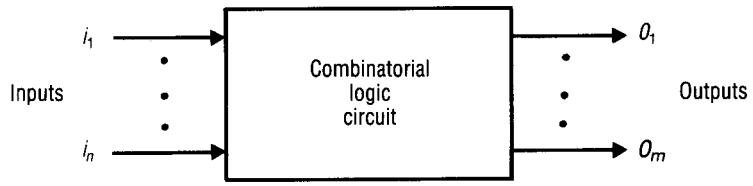


Fig. 3.1 Typical combinatorial logic circuit.

Defining a Combinatorial Circuit

- Truth table
  - For each of the 2^n possible combinations of input signals, the binary value of each of the *m* outputs is listed
- Boolean equations
  - Each output signal is expressed as a Boolean function of its input signals
- Graphical signals
  - Interconnection of gates used to implement the circuit

3-Bit Parity Generator Example

- Add an extra bit to data such that the number of ones in the data is always odd.
- Output - a single output (P)
- Inputs - three inputs labelled A, B, C

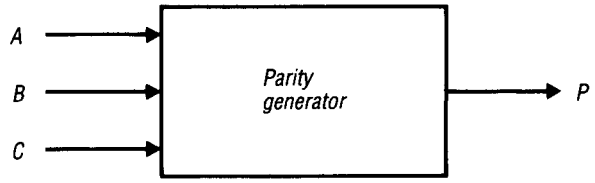


Fig. 3.2 Block diagram of parity generator circuit.

3-Bit Parity Generator Example (cont)

	A	B	C	P
0	0	0	0	
1	0	0	1	
2	0	1	0	
3	0	1	1	
4	1	0	0	
5	1	0	1	
6	1	1	0	
7	1	1	1	

3-Bit Parity Generator Example (cont)

- Boolean Equation
  - In ‘sum of products form’

$$P = \overline{A} \bullet \overline{B} \bullet \overline{C} + \overline{A} \bullet B \bullet C + A \bullet \overline{B} \bullet C + A \bullet B \bullet \overline{C}$$

3-Bit Parity Generator Example (cont)

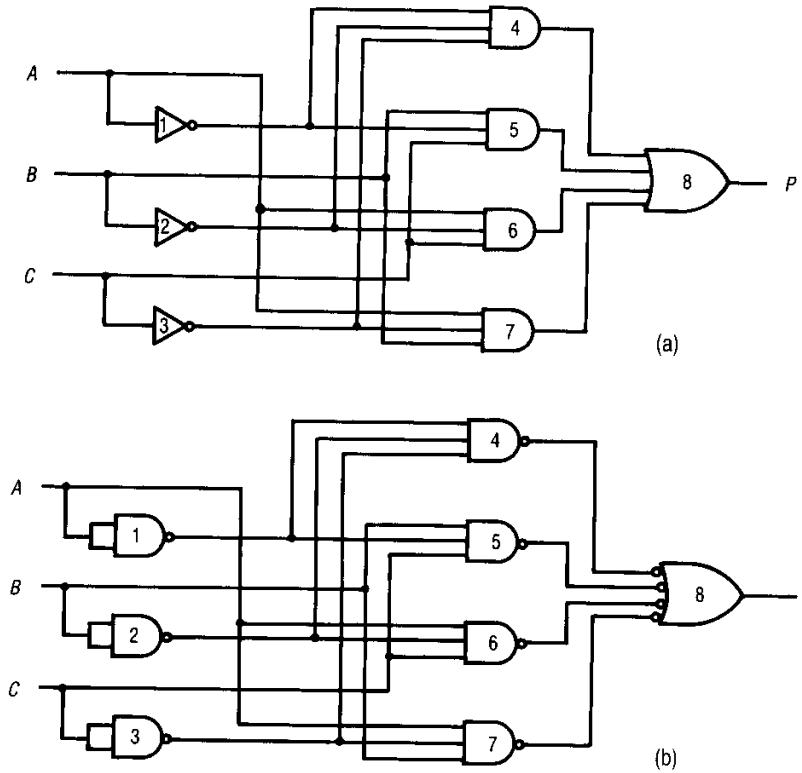
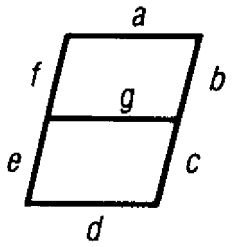


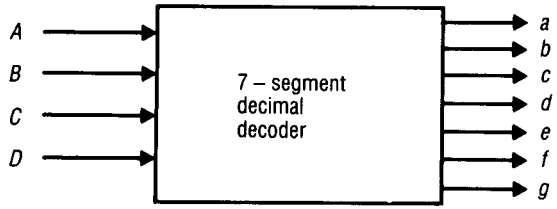
Fig. 3.3 Implementation of parity generator circuit using (a) basic gates; (b) only NAND gates.

How to Specify

- Method A
  - List the outputs required
  - List the inputs available
  - Write the Boolean functions required to obtain the outputs from the inputs
- Method B
  - Create a truth table
  - Label “don’t cares” with an X
  - Account for “can’t normally occur”
  - Can write the Boolean functions from the truth table

7-Segment Decimal Decoder Example



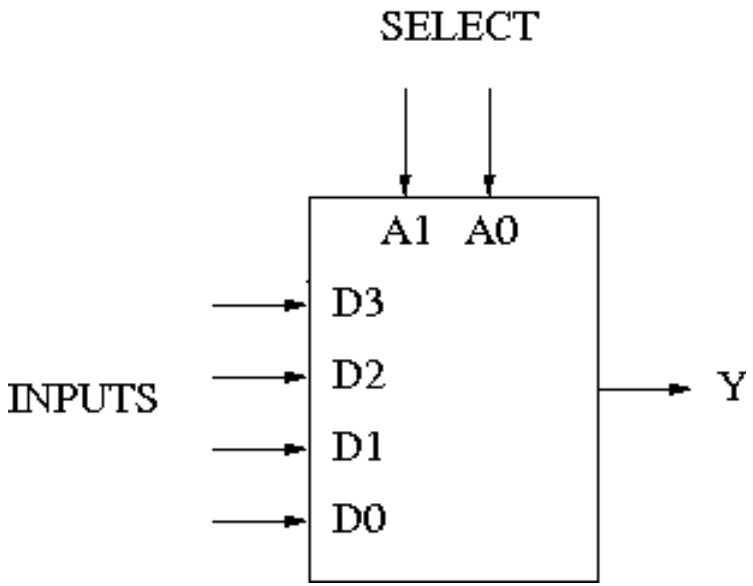


Decimal number	*Binary representation*				*Segments*						
	A	*B*	*C*	*D*	*a*	*b*	*c*	*d*	*e*	*f*	*g*
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	0	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1

Some Things We've Skipped

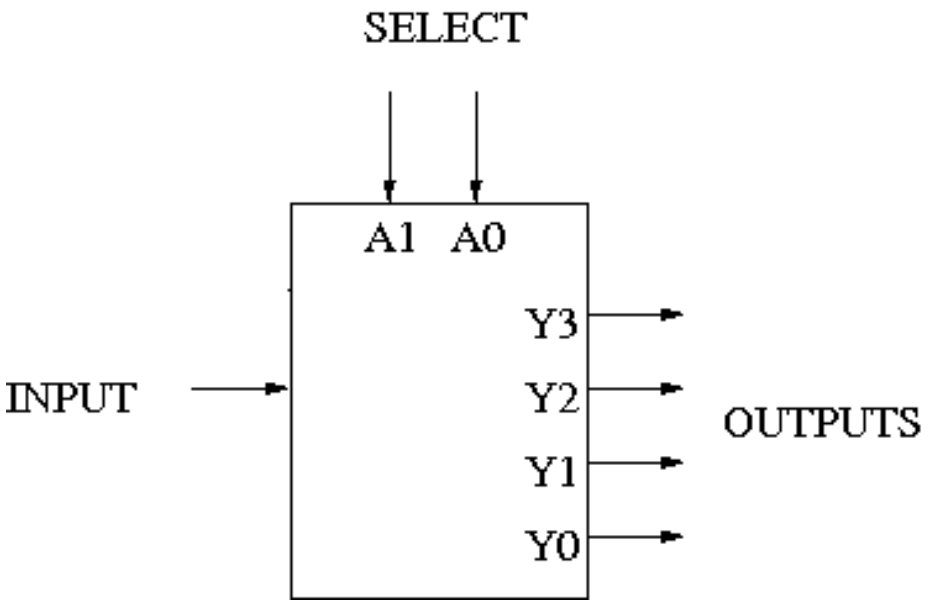
- Minimisation
  - Using Boolean algebra properties and propositions
  - Karnaugh maps
  - Digital techniques

Multiplexer



A1	A0	Y
0	0	D0
0	1	D1
1	0	D2
1	1	D3

Demultiplexer



A1	A0	INPUT
0	0	Y0
0	1	Y1
1	0	Y2
1	1	Y3

Summary

- Learn gate
  - symbols
  - truth tables
- Learn Boolean algebra
  - Rules
- Equations from truth tables
- Combinatorial logic