Boolean Algebra & Logic Design

Ref: Appendix B

1

Boolean Algebra

- Developed by George Boole in the 1850s
- Mathematical theory of logic.
- Shannon was the first to use Boolean Algebra to solve problems in electronic circuit design. (1938)

2

Variables & Operations

- All variables have the values 1 or 0
  - sometimes we call the values TRUE / FALSE
- Three operators:
  - OR written as +, as in $A+B$
  - AND written as •, as in $A \cdot B$
  - NOT written as an overline, as in $\overline{A}$

3

Operators: OR

- The result of the OR operator is 1 if either of the operands is a 1.
- The only time the result of an OR is 0 is when both operands are 0s.
- OR is like our old pal *addition*, but operates only on binary values.

4

Operators: AND

- The result of an AND is a 1 only when both operands are 1s.
- If either operand is a 0, the result is 0.

- AND is like our old nemesis *multiplication*, but operates on binary values.

5

Operators: NOT

- NOT is a *unary* operator – it operates on only one operand.
- NOT *negates* it's operand.
- If the operand is a 1, the result of the NOT is a 0.
- If the operand is a 0, the result of the NOT is a 17.678.

  just kidding – it's a 1 (*wake up*)!

6

Equations

Boolean algebra uses equations to express relationships. For example:

$$X = A \cdot (\overline{B} + C)$$

This equation expressed a relationship between the value of ***X*** and the values of ***A***, ***B*** and ***C***.

7

Quiz (already?)

What is the value of each X:

$$X_1 = 1 \cdot (0 + 1)$$

$$X_2 = A + \overline{A}$$

$$X_3 = A \cdot \overline{A}$$

$$X_4 = X_4 + 1$$ ← huh?

8

Laws of Boolean Algebra

Just like in *good old algebra*, Boolean Algebra has postulates and identities.

We can often use these laws to reduce expressions or put expressions in to a more desirable form.

9

Basic Postulates of Boolean Algebra

- Using just the basic postulates – everything else can be derived.

Commutative laws

Distributive laws

Identity

Inverse

10

Identity Laws

$$A + 0 = A$$

$$A \cdot 1 = A$$

11

Inverse Laws

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

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

12

Commutative Laws

$$A + B = B + A$$

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

13

Distributive Laws

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

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

14

Other Identities

Can be derived from the basic postulates.

Laws of Ones and Zeros

Associative Laws

DeMorgan’s Theorems

15

Zero and One Laws

$A + 1 = 1$ Law of Ones

$A \cdot 0 = 0$ Law of Zeros

16

Associative Laws

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

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

17

DeMorgan's Theorems

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

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

18

Other Operators

- Boolean Algebra is defined over the 3 operators AND, OR and NOT.
  - this is a *functionally complete set*.
- There are other useful operators:
  - NOR: is a 0 if either operand is a 1
  - NAND: is a 0 only if both operands are 1
  - XOR: is a 1 if the operands are different.
- NOTE: NOR is (by itself) a functionally complete set!

19

Boolean Functions

- Boolean functions are functions that operate on a number of Boolean variables.
- The result of a Boolean function is itself either a 0 or a 1.
- Example: *f(a,b) = a+b*

20

Question

- How many Boolean functions of 1 variable are there?
- We can answer this by listing them all!

$$f_1(x) = x$$
$$f_2(x) = \overline{x}$$
$$f_3(x) = 0$$
$$f_4(x) = 1$$

21

Tougher Question

- How many Boolean functions of 2 variables are there?
- It's much harder to list them all, but it is still possible…

22

Alternative Representation

- We can define a Boolean function by showing it using algebraic operations.
- We can also define a Boolean function by listing the value of the function for all possible inputs.

23

OR as a Boolean Function $f_{or}(a,b)=a+b$

a	b	$f_{or}(a,b)$
0	0	0
0	1	1
1	0	1
1	1	1

This is called a "truth table"

24

Truth Tables

a	*b*	OR	AND	NOR	NAND	XOR
0	0	0	0	1	1	0
0	1	1	0	0	1	1
1	0	1	0	0	1	1
1	1	1	1	0	0	0

25

Truth Table for (X+Y)·Z

X	Y	Z	(X+Y)·Z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

26

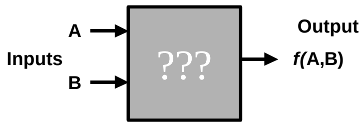
Gates

- Digital logic circuits are electronic circuits that are implementations of some Boolean function(s).
- A circuit is built up of *gates*, each *gate* implements some simple logic function.

- The term *gates* is named for Bill Gates, in much the same way as the term *gore* is named for Al Gore – the inventor of the Internet.

27

A Gate



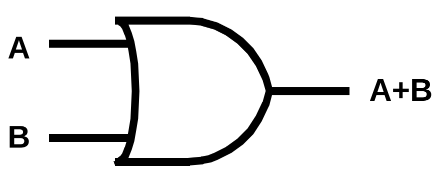
28

Gates compute something!

- The output depends on the inputs.
- If the input changes, the output might change.
- If the inputs don’t change – the output does not change.

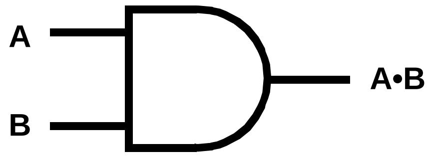
29

An OR gate



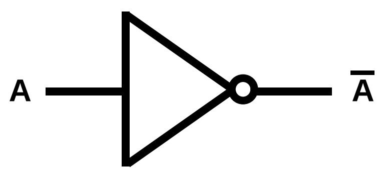
30

An AND gate



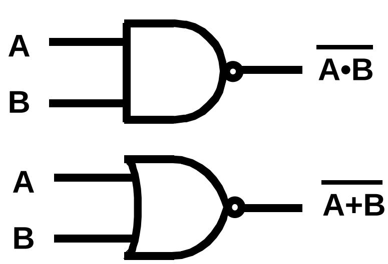
31

A NOT gate



32

NAND and NOR gates



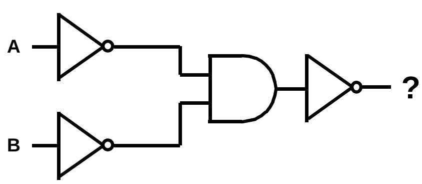
33

Combinational Circuits

- We can put gates together into circuits
  - output from some gates are inputs to others.
- We can design a circuit that represents any Boolean function!

34

A Simple Circuit



35

Truth Table for our circuit

a	b	$\bar{a}$	$\bar{b}$	$\bar{a} \bullet \bar{b}$	$\overline{\bar{a} \bullet \bar{b}}$
0	0	1	1	1	0
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	0	0	1

36

Alternative Representations

- Any of these can express a Boolean function. :

Boolean Equation

Circuit (Logic Diagram)

Truth Table

37

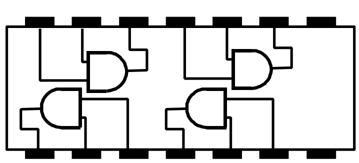
Implementation

- A logic diagram is used to design an *implementation* of a function.
- The implementation is the specific gates and the way they are connected.
- We can buy a bunch of gates, put them together (along with a power source) and build a machine.

38

Integrated Circuits

- You can buy an AND gate *chip*:



39

Function Implementation

- Given a Boolean function expressed as a truth table or Boolean Equation, there are many possible implementations.
- The actual implementation depends on what kind of gates are available.
- In general we want to minimize the number of gates.

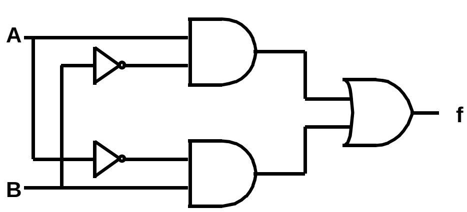
40

Example: $f = A \bullet \overline{B} + \overline{A} \bullet B$

A	B	$A \bullet \overline{B}$	$\overline{A} \bullet B$	f
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	0

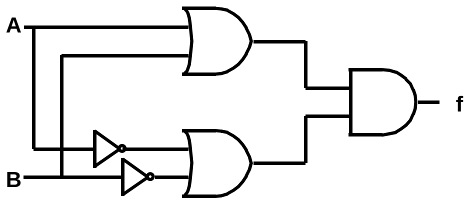
41

One Implementation $f = A \bullet \overline{B} + \overline{A} \bullet B$



42

Another Implementation



$$f = A \bullet \overline{B} + \overline{A} \bullet B = (A + B) \bullet (\overline{A} + \overline{B})$$

43

Proof it's the same function

$A \bullet \overline{B} + \overline{A} \bullet B =$	
$\overline{\overline{\overline{(A \bullet \overline{B})} \bullet \overline{(\overline{A} \bullet B)}}} =$	DeMorgan's Law
$\overline{(\overline{A} + B) \bullet (A + \overline{B})} =$	DeMorgan's Laws
$\overline{((\overline{A} + B) \bullet A) + ((\overline{A} + B) \bullet \overline{B})} =$	Distributive
$\overline{(\overline{A} \bullet A + B \bullet A) + (\overline{A} \bullet \overline{B} + B \bullet \overline{B})} =$	Distributive
$\overline{(B \bullet A) + (\overline{A} \bullet \overline{B})} =$	Inverse, Identity
$\overline{(B \bullet A)} \bullet \overline{(\overline{A} \bullet \overline{B})} =$	DeMorgan's Law
$(\overline{B} + \overline{A}) \bullet (A + B)$	DeMorgan's Laws

44