CS 230 – Introduction to Computers and Computer Systems

Module 1 – Arithmetic, Hardware, Data

Overview

- number representation
- boolean algebra and gate logic
- integer arithmetic
- non-numerical data types
- floating point

Number Representation

- *radix representation*
  - radix also know as *base*
- writing natural numbers using a finite alphabet
- given an n-digit word in base *r*

$$d_{n-1} d_{n-2} d_{n-3} \ldots d_3 d_2 d_1 d_0$$

- integer value is $\sum_{0}^{n-1} d_i * r^i$

Radix Representation

- humans: base-10, decimal
  - why?
- computers: base-2, binary
  - why? electrical simplicity
    - analog/digital conversion (high vs. low voltage)
  - low-level decimal conversion? Only if necessary
    - storage expansion / waste

Examples

135_{dec} $5 \cdot 10^0 + 3 \cdot 10^1 + 1 \cdot 10^2 = 135$

- not too surprising...

1440_{sep} $0 \cdot 7^0 + 4 \cdot 7^1 + 4 \cdot 7^2 + 1 \cdot 7^3 = 567$

$A32_{hex}$ $2 \cdot 16^0 + 3 \cdot 16^1 + 10 \cdot 16^2 = 2610$

- use letters A...F to express digits > 9
- A-10, B-11, C-12, D-13, E-14, F-15

Conversion from Decimal

- repeatedly divide by target base
- remainders generate digits
  - from right to left...
- example: $3219_{\mathbf{dec}}$

$$\begin{aligned} 3219/16 &= 201 \; R\,3 \\ 201/16 &= 12 \quad R\,9 \\ 12/16 &= 0 \quad\; R\,12 \end{aligned}$$

= $C93_{\mathbf{hex}}$

Binary Numbers

- only 0 and 1 as digits – represent low and high voltage
- example

$11101100_{\textbf{bin}}$

$$2^2 + 2^3 + 2^5 + 2^6 + 2^7 = 236$$

- permits simple binary operations

Binary / Hex Conversion

$0000_{\mathbf{bin}} = 0_{\mathbf{hex}}$

$0001_{\mathbf{bin}} = 1_{\mathbf{hex}}$

$0010_{\mathbf{bin}} = 2_{\mathbf{hex}}$

$0011_{\mathbf{bin}} = 3_{\mathbf{hex}}$

…

$1111_{\mathbf{bin}} = F_{\mathbf{hex}}$

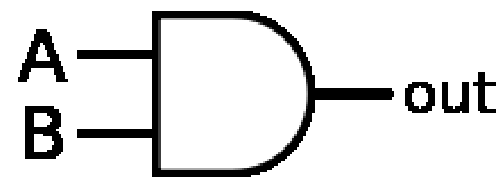
Boolean Algebra

- algebra to express binary logic
- basic operators: **OR, AND, NOT**
- OR operators: ∨
  - A ∨ B
- AND operators: ∧
  - A ∧ B
- NOT operators: ¬ -
  - ¬ A $\overline{A}$

AND

Truth Table

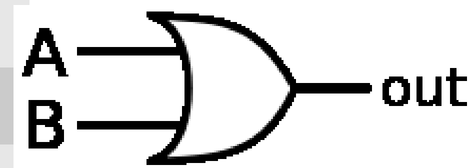
A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

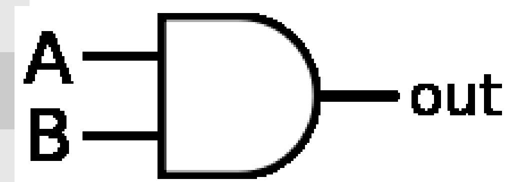


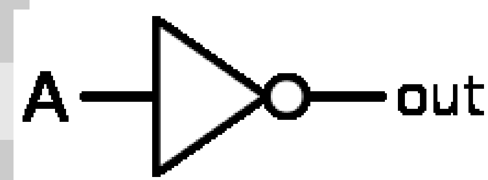
- Logic Gates: Digital systems are constructed using logic gates that implement functions like AND, OR, NOT.
  - Logic gates are implemented using electronic circuits

Truth Tables and Gates

OR	X	Y	Result
X ∨ Y	0	0	0
X + Y	0	1	1
	1	0	1
	1	1	1
AND	X	Y	Result
X ∧ Y	0	0	0
X * Y	0	1	0
	1	0	0
	1	1	1
NOT	X	Result	
¬ X	0	1	
	1	0	







Other Rules

- Identities

$A \vee 0 = A$

$A \wedge 1 = A$

$A \vee A = A$

$A \wedge A = A$

- Involution

$\neg(\neg A) = A$

- Annihilators

$A \vee 1 = 1$

$A \wedge 0 = 0$

- Complements

$A \vee \neg A = 1$

$A \wedge \neg A = 0$

Other Rules - 2

- Commutative Law

$A + B = B + A$ *and* $A * B = B * A$

$A \vee B = B \vee A$ *and* $A \wedge B = B \wedge A$

- Associative Law

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

$A \vee (B \vee C) = (A \vee B) \vee C$ *and*
$A \wedge (B \wedge C) = (A \wedge B) \wedge C$

Other Rules - 3

- Distributive Law

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

$A \wedge (B \vee C) = A \wedge B \vee A \wedge C$

- De Morgan's Law

$\neg (A \vee B) = \neg A \wedge \neg B$

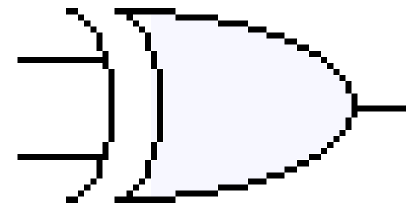
$\neg (A \wedge B) = \neg A \vee \neg B$

EXCLUSIVE OR

- XOR

$X \oplus Y$

X	Y	Result
0	0	0
0	1	1
1	0	1
1	1	0

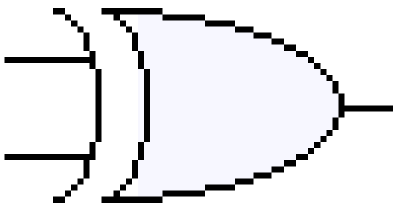


EXCLUSIVE OR

- XOR

$X \oplus Y$

X	Y	Result
0	0	0
0	1	1
1	0	1
1	1	0



$$X \oplus Y = (\neg X \land Y) \lor (X \land \neg Y)$$

$$X \oplus Y = \neg(X \land Y) \land (X \lor Y)$$

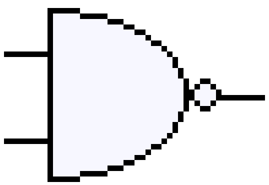
$$X \oplus Y = (\neg X \lor \neg Y) \land (X \lor Y)$$

NOT AND

- NAND

$X \mid Y$

X	Y	Result
0	0	1
0	1	1
1	0	1
1	1	0



- NAND is *functionally complete* (as is NOR)

  $\neg X = X \mid X$

  $X \wedge Y = (X \mid Y) \mid (X \mid Y)$

  $X \vee Y = (X \mid X) \mid (Y \mid Y)$

Bytes

1 byte (B)	8 bit
1 kilobyte (K/Kb)	2^{10} byte = 1024 byte
1 megabyte (M/Mb)	2^{20} byte = 1024 Kb
1 gigabyte (G/Gb)	2^{30} byte = 1024 Mb
1 terabyte (T/Tb)	2^{40} byte = 1024 Gb
1 petabyte (P/Pb)	2^{50} byte = 1024 Tb
1 exabyte (E/Eb)	2^{60} byte = 1024 Pb
1 zettabyte (Z/Zb)	2^{70} byte = 1024 Eb
1 yottabyte (Y/Yb)	2^{80} byte = 1024 Zb

Binary Addition

- textbook procedure
- add digits right to left (least significant bit to most significant bit)
  - include carry-over

2-Bit Addition

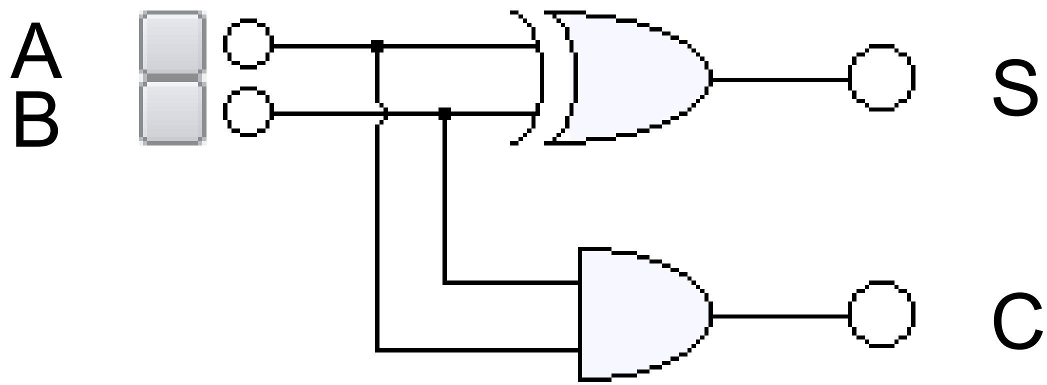
- Truth Table

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- Sum = _____
- Carry = _____

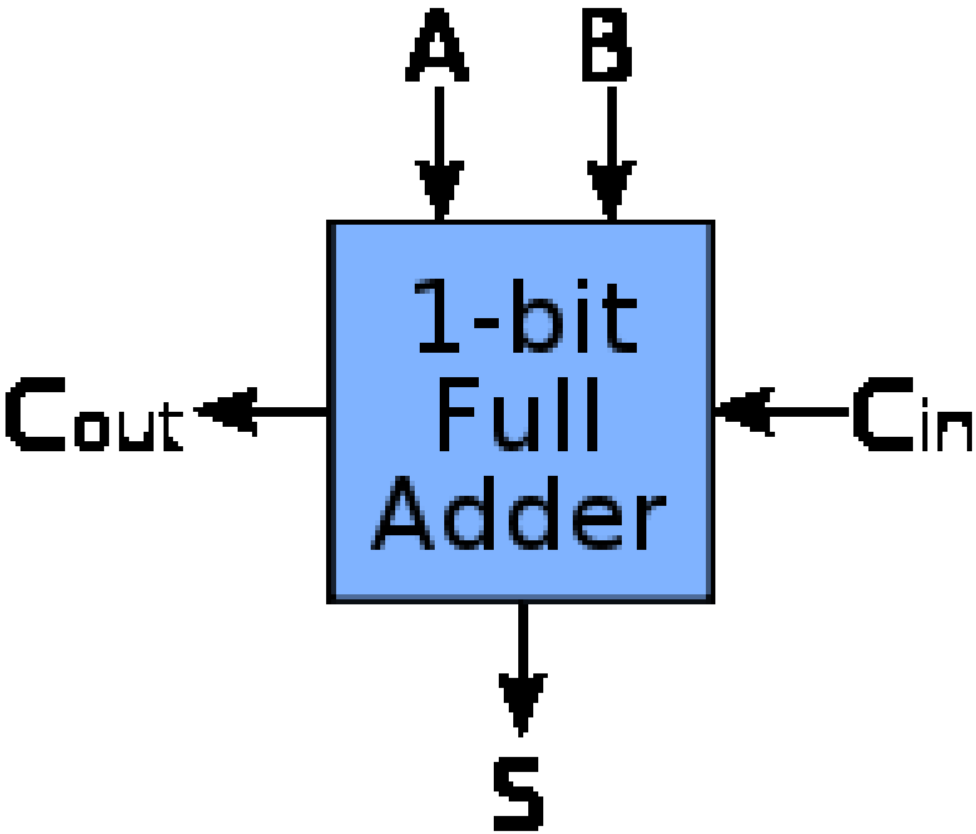
Half Adder

- combine XOR and AND gate



- **S**um and **C**arry
- Carry-in?

Full Adder



2-Bit Addition with Carry

- multiple bits: need to include carry
- Truth Table

A	B	$Carry_{in}$	Sum	$Carry_{o\ out}$
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

2-Bit Addition with Carry

- Sum: 1 or 3 bits set

$A \oplus B \oplus C_{in}$

- C_{out}: 2 or 3 bits set

$(A \land B) \lor (A \land C_{in}) \lor (B \land C_{in})$

$(A \land B) \lor ((A \lor B) \land C_{in})$

$(A \land B) \lor ((A \oplus B) \land C_{in})$

Full Adder

- comprised of two half adders and OR