Logic:Expression

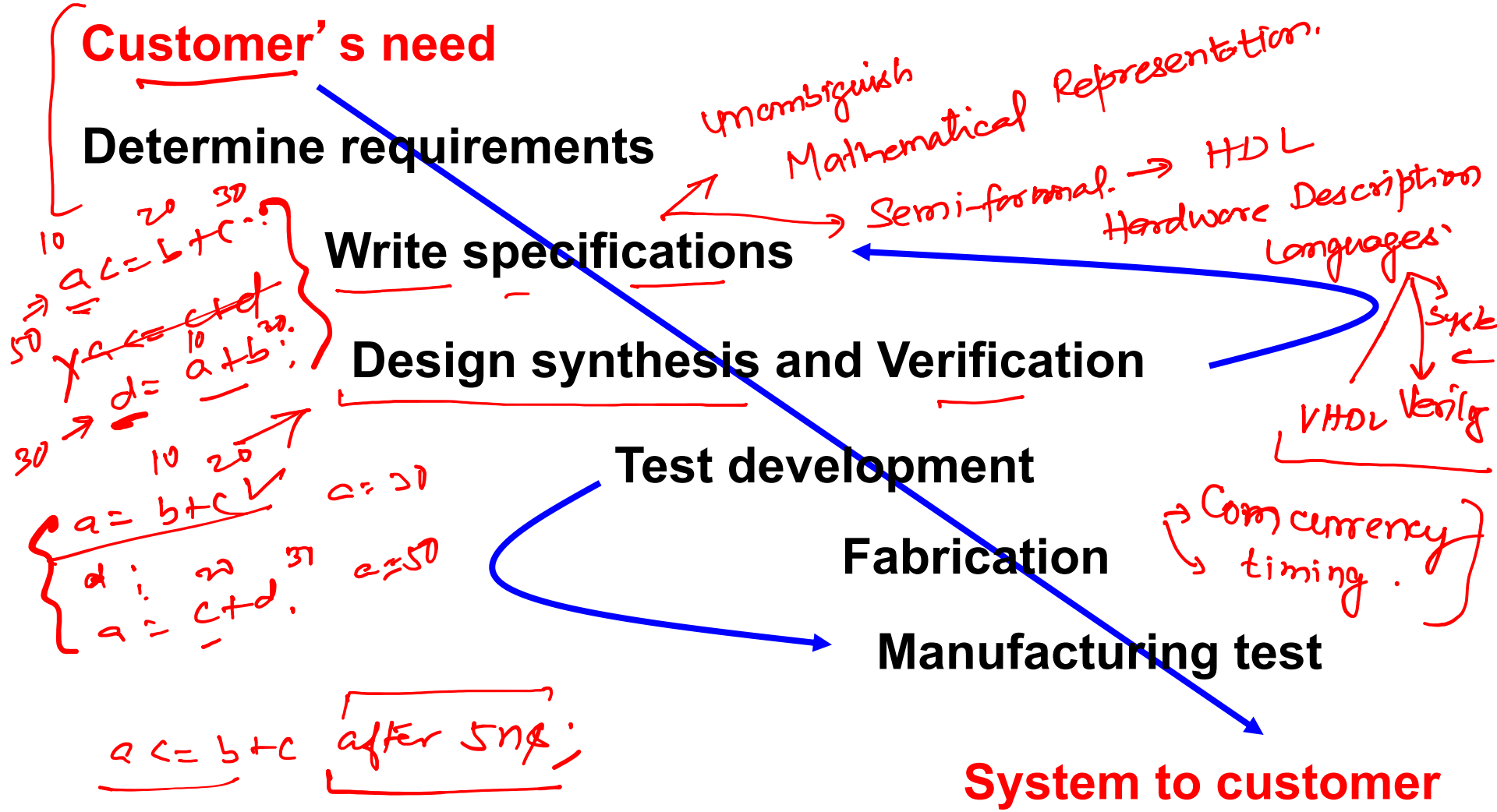
Virendra Singh
Professor
Computer Architecture and Dependable Systems Lab
Department of Computer Science & Engineering, and
Department of Electrical Engineering
Indian Institute of Technology Bombay
http://www.ee.iitb.ac.in/~viren/
E-mail: viren@{cse, ee}.iitb.ac.in

CS-230: Digital Logic Design & Computer Architecture

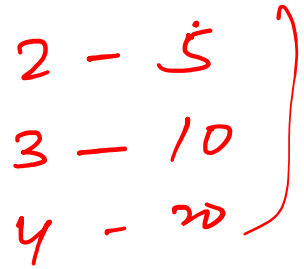
Lecture 2 (06 January 2022)

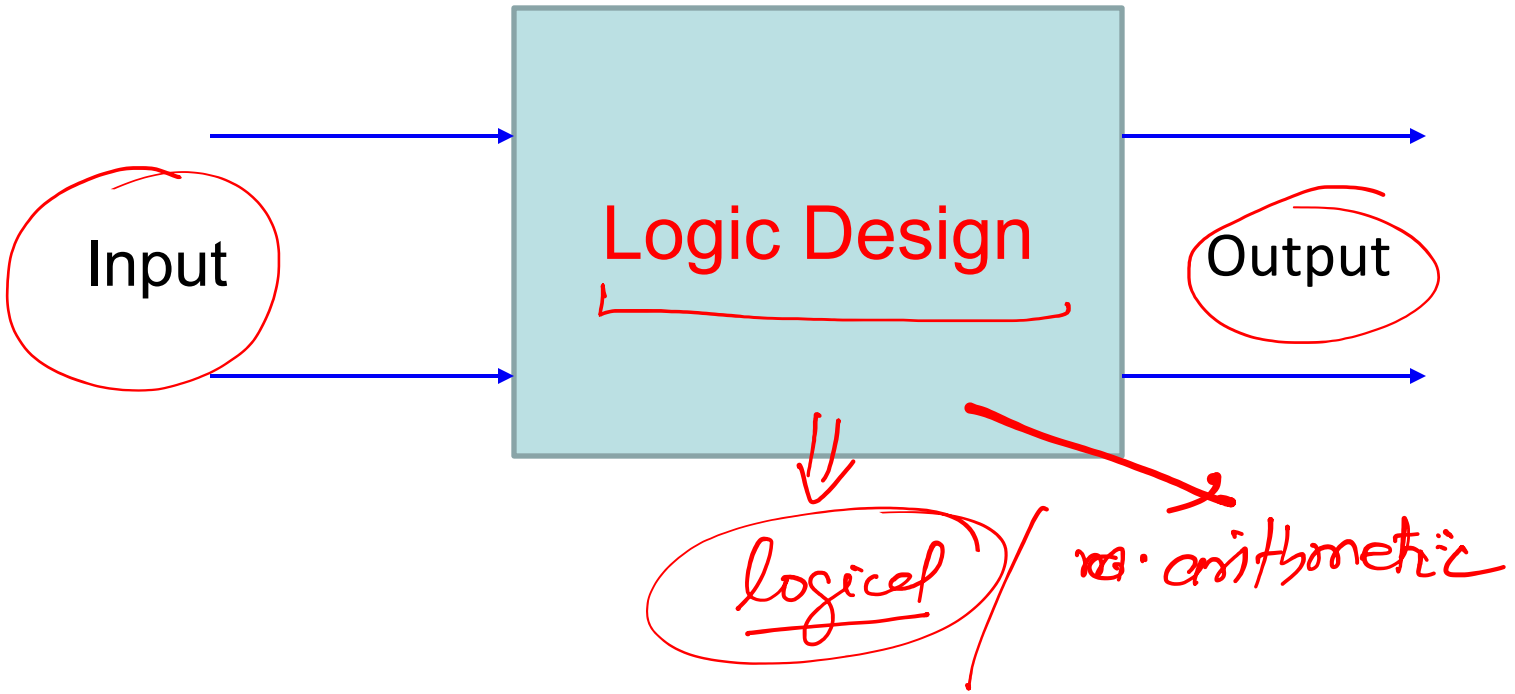
CADSL

System Realization Process



Digital System





LOGIC

What is logic?

- Logic is the study of valid reasoning.

- That is, logic tries to establish criteria to decide whether some piece of reasoning is valid or invalid.

Logic

- Crucial for mathematical reasoning
- Used for designing electronic circuitry
- Logic is a system based on propositions.
- A proposition is a statement that is either true or false (not both).

Introduction: PL

- In Propositional Logic (a.k.a Propositional Calculus or Sentential Logic), the objects are called propositions
- **Definition**: A proposition is a statement that is either true or false, but not both
- We usually denote a proposition by a letter: *p*, *q*, *r*, *s*, …

Introduction: Proposition

- **Definition**: The value of a proposition is called its truth value; denoted by
  - *T* or 1 if it is true or
  - *F* or 0 if it is false
- Opinions, interrogative, and imperative are not propositions
- **Truth table**

p
0
1

Propositions: Examples

- The following are propositions
  - Today is Thursday *M*
  - The floor is wet *W*
  - It is raining *R*
- The following are not propositions
  - Python is the best language *Opinion*
  - When will be the next class? *Interrogative*
  - Do your homework *Imperative*

Logical Operators

- Operators/Connectives are used to create a compound proposition from two or more propositions
  - Negation (denote $\neg$ or ! Or ~)
  - And or logical conjunction (denoted $\wedge$ or .) – logical AND
  - Or or logical disjunction (denoted $\vee$ or +) – logical OR
  - XOR or exclusive or (denoted $\oplus$)
  - Implication (denoted $\Rightarrow$ or $\rightarrow$)
  - Biconditional (denoted $\Leftrightarrow$ or $\leftrightarrow$)

  We define the meaning (semantics) of the logical operators/connectives using truth tables

Logical Operator: Negation

- $\neg p$, the negation of a proposition p, is also a proposition
- Examples:
  - Today is not Monday.
- **Truth table**

p	$\neg p$
0	1
1	0

Logical Operator: Logical And

- The logical operator And is true only when both of the propositions are true. It is also called a conjunction
- Examples
  - It is raining and it is cold
  - (2+3=5) and (1<2)
- Truth table

p	q	$p \wedge q$
0	0	0
0	1	0
1	0	0
1	1	1

Logical Operator: Logical Or

- The logical disjunction, or logical Or, is true if one or both of the propositions are true.
- Examples
  - It is raining or it is the second lecture
  - (2+2=5) $\vee$ (1<2)
  - You may have cake or ice cream
- **Truth table**

p	q	$p \wedge q$	$p \vee q$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Logical Operator: Exclusive Or

- The exclusive Or, or XOR, of two propositions is true when exactly one of the propositions is true and the other one is false
- Example
  - The circuit is either ON or OFF but not both
  - Let $ab<0$, then either $a<0$ or $b<0$ but not both
  - You may have cake or ice cream, but not both
- Truth table

p	q	$p \wedge q$	$p \vee q$	$p \oplus q$
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Logical Operator: Implication

- **Definition:** Let p and q be two propositions. The implication $p \rightarrow q$ is the proposition that is false when p is true and q is false and true otherwise
  - p is called the hypothesis, antecedent, premise
  - q is called the conclusion, consequence
- **Truth table**

p	q	$p \wedge q$	$p \vee q$	$p \oplus q$	$p \Rightarrow q$
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	1

Logical Operator: Implication

- The implication of $p \rightarrow q$ can be also read as
  - If p then q
  - p implies q
  - If p, q
  - p only if q
  - q if p
  - q when p
  - q whenever p
  - q follows from p
  - p is a sufficient condition for q (p is sufficient for q)
  - q is a necessary condition for p (q is necessary for p)

Logical Operator: Implication

- Examples
  - If you buy you air ticket in advance, it is cheaper.
  - If x is an integer, then $x^2 \geq 0$.
  - If it rains, the grass gets wet.
  - If the sprinklers operate, the grass gets wet.

Logical Operator: Equivalence

- **Definition:** The equivalence/biconditional $p \leftrightarrow q$ is the proposition that is true when p and q have the same truth values. It is false otherwise.
- Note that it is equivalent to $(p \rightarrow q) \wedge (q \rightarrow p)$
- **Truth table**

p	q	$p \wedge q$	$p \vee q$	$p \oplus q$	$p \Rightarrow q$	$p \Leftrightarrow q$
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	0	0
1	1	1	1	0	1	1

Logical Operator: Equivalence

- The biconditional $p \leftrightarrow q$ can be equivalently read as
  - p if and only if q
  - p is a necessary and sufficient condition for q
  - if p then q, and conversely
  - p iff q
- Examples
  - $x>0$ if and only if x^2 is positive
  - You may have pudding iff you eat your meal

Truth Tables

- Truth tables are used to show/define the relationships between the truth values of
  - the individual propositions and
  - the compound propositions based on them

p	q	$p \wedge q$	$p \vee q$	$p \oplus q$	$p \Rightarrow q$	$p \Leftrightarrow q$
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	0	0
1	1	1	1	0	1	1

Constructing Truth Tables

- Construct the truth table for the following compound proposition

$$((p \wedge q) \vee \neg q)$$

LOGICAL

p	q	$p \wedge q$	$\neg q$	$((p \wedge q) \vee \neg q)$
0	0	0	1	1
0	1	0	0	0
1	0	0	1	1
1	1	1	0	1

How to Specify Arithmetic Operations?

Number System

Why Binary Arithmetic?

0 — 9

3 + 5

= 8

0011 + 0101

= 1000

Number Systems – Representation

- Positive radix, positional number systems
- A number with *radix* ***r*** is represented by a string of digits:

  $$A_{n-1}A_{n-2} \ldots A_1A_0 \cdot A_{-1}A_{-2} \ldots A_{-m+1}A_{-m}$$

  in which $\mathbf{0} \leq \boldsymbol{A_i} < \boldsymbol{r}$ and **.** is the *radix point*.
- The string of digits represents the power series:

$$(\text{Number})_r = \left(\sum_{i=0}^{i=n-1} A_i \cdot r^i\right) + \left(\sum_{j=-m}^{j=-1} A_j \cdot r^j\right)$$

(Integer Portion) + (Fraction Portion)

Number Systems – Examples

		General	Decimal	Binary
Radix (Base)		r	10	2
Digits		0 => r - 1	0 => 9	0 => 1
Powers of Radix	0	r^0	1	1
	1	r^1	10	2
	2	r^2	100	4
	3	r^3	1000	8
	4	r^4	10,000	16
	5	r^5	100,000	32
	-1	r^{-1}	0.1	0.5
	-2	r^{-2}	0.01	0.25
	-3	r^{-3}	0.001	0.125
	-4	r^{-4}	0.0001	0.0625
	-5	r^{-5}	0.00001	0.03125

Positive Powers of 2

- Useful for Base Conversion

Exponent	Value
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024

Exponent	Value
11	2,048
12	4,096
13	8,192
14	16,384
15	32,768
16	65,536
17	131,072
18	262,144
19	524,288
20	1,048,576
21	2,097,152

Converting Binary to Decimal

- To convert to decimal, use decimal arithmetic to form S (digit × respective power of 2).
- Example: Convert 11010_2 to N_{10}:

Powers of 2:	43210
	11010
	$1 \times 2^4 = 16$
	$1 \times 2^3 = 8$
	$0 \times 2^2 = 0$
	$1 \times 2^1 = 2$
	$0 \times 2^0 = 0$
Sum	$\Rightarrow = 26_{10}$

Converting Decimal to Binary

- Method 1
  - Subtract the largest power of 2 that gives a positive remainder and record the power.
  - Repeat, subtracting from the prior remainder and recording the power, until the remainder is zero.
  - Place 1's in the positions in the binary result corresponding to the powers recorded; in all other positions place 0's.
- Example: Convert 625_{10} to N_2

Converting Decimal to Binary

- Subtract the largest power of 2 (see slide 14) that gives a positive remainder and record the power.
- Repeat, subtracting from the prior remainder and recording the power, until the remainder is zero.
- Place 1's in the positions in the binary result corresponding to the powers recorded; in all other positions place 0's.
- Example: Convert 625_{10} to N_2

9 8 7 6 5 4 3 2 1 0

1 0 0 1 1 1 0 0 0 1

625 - 512	= 113	⇒9
113 - 64	= 49	⇒6
49 - 32	= 17	⇒5
17 - 16	= 1	⇒4
1 - 1	= 0	⇒0
Placing 1's in the result for the positions recorded and 0's elsewhere:		
9876543210		
1001110001		

Commonly Occurring Bases

Name	Radix	Digits
Binary	2	0,1
Octal	8	0,1,2,3,4,5,6,7
Decimal	10	0,1,2,3,4,5,6,7,8,9
Hexadecimal	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

The six letters (in addition to the 10 integers) in hexadecimal represent:

10, 11, 12, 13, 14, 15

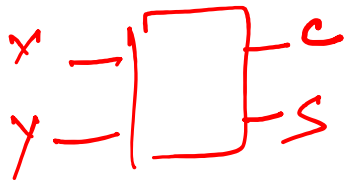
Binary Arithmetic

Single Bit Binary Addition

Given two binary digits (X,Y), we get the following sum (S) and carry (C):

X	**0**	**0**	**1**	**1**
+ Y	**+ 0**	**+ 1**	**+ 0**	**+ 1**
C S	**0 0**	**0 1**	**0 1**	**1 0**

Truth Table: Two Bit Adder



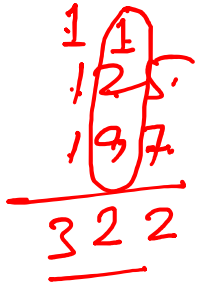
X	Y	Binary Sum (C)(S)
0	0	0 0
0	1	0 1
1	0	0 1
1	1	1 0

CARRY

SUM

Truth Tables of Logical Operations

- Truth tables are used to show/define the relationships between the truth values of
  - the individual propositions and
  - the compound propositions based on them



p	q	$p \wedge q$	$p \vee q$	$p \oplus q$	$p \Rightarrow q$	$p \Leftrightarrow q$
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	0	0
1	1	1	1	0	1	1

Single Bit Binary Addition with Carry

Given two binary digits (X,Y), a carry in (Z) we get the following sum (S) and carry (C):

Carry in (Z) of 0:

Z	0	0	0	0
X	0	0	1	1
+ Y	+ 0	+ 1	+ 0	+ 1
C S	0 0	0 1	0 1	1 0

Carry in (Z) of 1:

Z	1	1	1	1
X	0	0	1	1
+ Y	+ 0	+ 1	+ 0	+ 1
C S	0 1	1 0	1 0	1 1

Full Adder: Include Carry Input

Z	Y	X	S = X + Y + Z	
			Decimal value	Binary value
0	0	0	0	0 0
0	0	1	1	0 1
0	1	0	1	0 1
0	1	1	2	1 0
1	0	0	1	0 1
1	0	1	2	1 0
1	1	0	2	1 0
1	1	1	3	1 1

CARRY

SUM

Truth Table: Full Adder

Z	Y	X	Binary value (C)(S)
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

CARRY

SUM

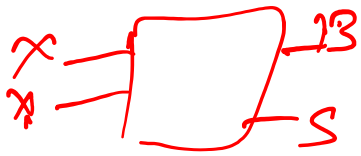
Multiple Bit Binary Addition

- **Extending this to a multiple bit examples:**

Carries	00000
Augend	01100
Addend	10001
Sum	11101

Single Bit Binary Subtraction

- **Given two binary digits (X,Y), we get the following difference (S) and borrow (B)**



X	0	0	1	1
- Y	- 0	-1	-0	-1
BS	0 0	1 1	0 1	0 0

Borrow

Truth Table: Two Bit Subtractor

X	Y	Binary Difference (B)(D)
0	0	0 0
0	1	1 1
1	0	0 1
1	1	0 0

BORROW

DIFFERENCE

Single Bit Binary Subtraction with Borrow

- **Given two binary digits (X,Y), a borrow in (Z) we get the following difference (D) and borrow (B):**
- **Borrow in (Z) of 0:**

Z	0	0	0	0
X	0	0	1	1
- Y	-0	-1	-0	-1
BD	0 0	1 1	0 1	0 0

- **Borrow in (Z) of 1:**

Z	1	1	1	1
X	0	0	1	1
- Y	-0	-1	-0	-1
BD	11	1 0	0 0	1 1

Truth Table: Full Subtractor

Z	Y	X	Binary value (C)(D)
0	0	0	0 0
0	0	1	1 1
0	1	0	0 1
0	1	1	0 0
1	0	0	1 1
1	0	1	1 0
1	1	0	0 0
1	1	1	1 1

CARRY

DIFFERENCE

Multiple Bit Binary Subtraction

- **Extending this to a multiple bit example:**
- **Notes:**
  - **The 0 is a Borrow-In to the least significant bit.**
  - **If the Subtrahend > the Minuend, interchange and append a – to the result.**

Borrows	00000
Minuend	10110
Subtrahend	10010
Difference	00100

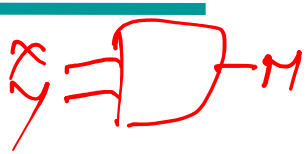
Multiple Bit Binary Subtraction

- Extending this to a multiple bit examples:
- Notes: The 0 is a Borrow-In to the least significant bit. If the Subtrahend > the Minuend, interchange and append a – to the result.

Borrows	00110
Minuend	10110
Subtrahend	10011
Difference	00011

Binary Multiplication

The binary multiplication table is simple:



0 * 0 = 0 | 1 * 0 = 0 | 0 * 1 = 0 | 1 * 1 = 1

Extending multiplication to multiple digits:

Multiplicand	**1011**
Multiplier	**x 101**
Partial Products	**1011**
	0000 -
	1011 - -
Product	**110111**

Unsigned numbers

Signed Magnitude?

- Use fixed length binary representation
- Use left-most bit (called *most significant bit* or MSB) for sign:

  0 for positive

  1 for negative

- Example: $+18_{ten} = 00010010_{two}$

  $-18_{ten} = 10010010_{two}$

Difficulties with Signed Magnitude

- Sign and magnitude bits should be differently treated in arithmetic operations.
- Addition and subtraction require different logic circuits.
- Overflow is difficult to detect.
- "Zero" has two representations:

$$+ 0_{ten} = 00000000_{two}$$

$$- 0_{ten} = 10000000_{two}$$

- *Signed-integers are not used in modern computers.*

Thank You