EE-224: Digital Design

Expression

Virendra Singh

Professor

Computer Architecture and Dependable Systems Lab

Department of Electrical Engineering

Indian Institute of Technology Bombay

http://www.ee.iitb.ac.in/~viren/

E-mail: viren@ee.iitb.ac.in

System Realization Process

Customer's need

Determine requirements

Write specifications

Design synthesis and Verification

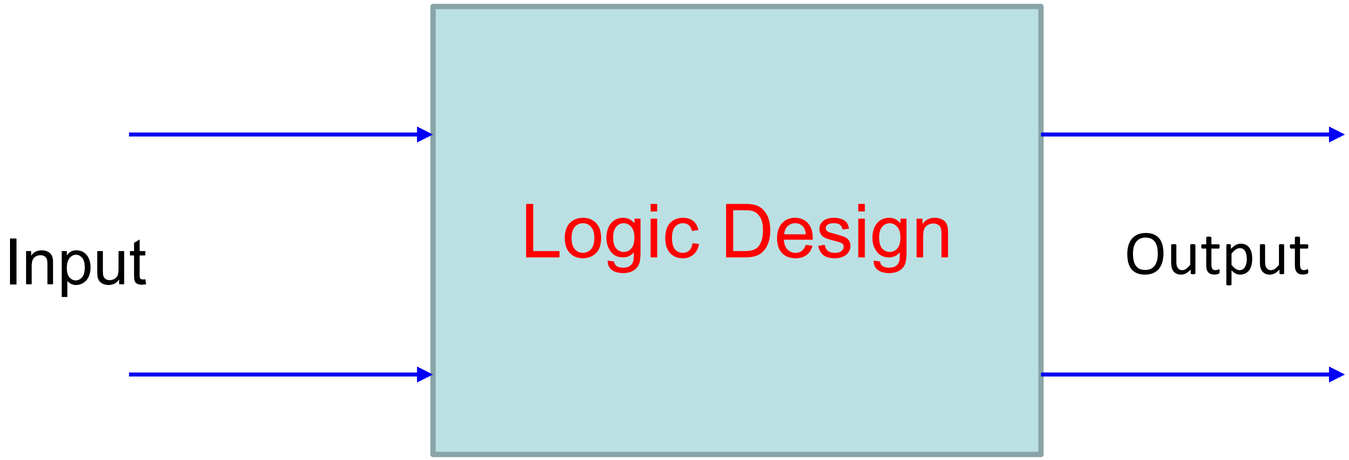
Test development

Fabrication

Manufacturing test

System to customer

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.

Valid Reasoning

- While in every piece of reasoning certain statements are *claimed to* follow from others, this may in fact not be the case.
- Example: "If I win the lottery, then I'm happy. However, I did not win the lottery. Therefore, I am not happy."
- A piece of reasoning is *valid* if the statements that are claimed to follow from previous ones do *indeed* follow from those. Otherwise, the reasoning is said to be *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).

The Statement/Proposition

- "Elephants are bigger than mice."

Is this a statement? yes

Is this a proposition? yes

What is the truth value of the proposition? true

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*

Are these propositions?

- 2+2=7
- Every integer is divisible by 10
- Google is an excellent company

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)
  - Ramesh's cow is dead and Ramesh's is not dead.
- Truth table

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

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	
0	1	0	1	1	
1	0	0	1	1	
1	1	1	1	0	

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.

Exercise: Which of the following implications is true?

- If -1 is a positive number, then 2+2=5

  True. The premise is obviously false, thus no matter what the conclusion is, the implication holds.

- If -1 is a positive number, then 2+2=4

  True. Same as above.

- If *sin* $x = 0$, then $x = 0$

  False. x can be a multiple of π. If we let $x=2\pi$, then *sin* $x=0$ but $x\neq 0$. The implication "if *sin* $x = 0$, then $x = k\pi$, for some k" is true.

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	
0	1	0	1	1	1	
1	0	0	1	1	0	
1	1	1	1	0	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

Which of the following equivalence/biconditionals is true?

- $x^2 + y^2 = 0$ if and only if $x=0$ and $y=0$

  True. Both implications hold

- $2 + 2 = 4$ if and only if $\sqrt{2}<2$

  True. Both implications hold.

- $x^2 \geq 0$ if and only if $x \geq 0$

  False. The implication "if $x \geq 0$ then $x^2 \geq 0$" holds.
  However, the implication "if $x^2 \geq 0$ then $x \geq 0$" is false.
  Consider $x=-1$.
  The hypothesis $(-1)^2=1 \geq 0$ but the conclusion fails.

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

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

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?

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:

$$A_{n-1}A_{n-2}\ \dots\ A_1A_0\ .\ A_{-1}\ A_{-2}\ \dots\ A_{-m+1}\ A_{-m}$$

in which $\mathbf{0 \leq A_i < r}$ 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

Special Powers of 2

2^{10} (1024) is Kilo, denoted "K"

2^{20} (1,048,576) is Mega, denoted "M"

2^{30} (1,073, 741,824)is Giga, denoted "G"

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 $\times$ respective power of 2).
- Example:Convert 11010_2 to N_{10}:

```
Powers of 2:   43210
               11010
             ⇨ 1 x 2^4 = 16
             ⇨ 1 x 2^3 =  8
             ⇨ 0 x 2^2 =  0
             ⇨ 1 x 2^1 =  2
             ⇨ 0 x 2^0 =  0
Sum                  ⇨ = 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

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

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