THE UNIVERSITY OF TEXAS AT DALLAS

CS4341 Digital Logic & Computer Design

Lecture Notes 3

Omar Hamdy
Assistant Professor
Department of Computer Science

Review: Signed Numbers

- *To carry out the various arithmetic operations, we need numbers as positive or negative.*
- *In sign-magnitude notation, the MSB is reserved to represent the number sign. 0 represents plus sign and 1 represents minus sign*
- *the 1's complement of N is obtained by reversing each bit in N (0 becomes 1, and 1 becomes 0).*
- *If N consists of n-bits then, 1's complement of N =* $\mathbf{(2^n - 1) - N}$

Review: 2’s Complement Representation

- *Used in most computers today*
- *Definition: given a binary number N, the 2’s complement of N =1’s complement + 1*

starting value	$00100100_2 = +36$
step1: reverse the bits (1's complement)	11011011_2
step 2: add 1 to the value from step 1	$+ \quad 1_2$
sum = 2's complement representation	$11011100_2 = -36$

Review: 2’s Complement Representation

- *Another way to obtain the 2's complement: Starting at the least significant 1:*
- Leave all the 0s to its right unchanged
- Complement all the bits to its left

```
Binary Value
= 00100 1 00
2's Complement
= 11011 1 00
```

Ranges for Unsigned and Signed Numbers

- For n-bit unsigned integers: Range is 0 to $(2^n - 1)$
- For n-bit signed integers: Range is -2^{n-1} to $(2^{n-1} - 1)$:
  - Positive range: 0 to $(2^{n-1} - 1)$
  - Negative range: -2^{n-1} to -1

Note: in 2's Complement, there is only one zero: 2's complement of 0 = 0

Unsigned and Signed Values

➢ *Positive numbers:*

- Signed value = Unsigned value

➢ *Negative numbers:*

- To obtain the signed value, we assign a negative weight to MSB and add to the rest of the bit values.

$$N = b_{n-1} \times (-2^{n-1}) + b_{n-2} \times 2^{n-2} + \ldots + b_0 \times 2^0$$

Example

Find the numerical value of signed number $(10110100)_2$

1	0	1	1	0	1	0	0
-128	64	32	16	8	4	2	1

- Value = -128 + 32 + 16 + 4 = -76
- If we store this number in 16-bit memory instead, would the numerical value change?

Example

Find the signed and unsigned ranges in 4-bit binary representation

- Unsigned: 0000_2 to 1111_2 = 0_{10} - 15_{10}
- Signed (using 2's Complement):
  - Positive Range: 0000_2 (0_{10}) to 0111_2 ($+7_{10}$)
  - Negative Range: 1000_2 (-8_{10}) to 1111_2(-1_{10})

Ranges for Unsigned and Signed Numbers

8-bit Binary value	Unsigned value	Signed value
00000000	0	0
00000001	1	+1
00000010	2	+2
. . .	. . .	. . .
01111110	126	+126
01111111	127	+127
10000000	128	-128
10000001	129	-127
. . .	. . .	. . .
11111110	254	-2
11111111	255	-1

Binary Addition

Start with the least significant bit (rightmost bit):

- Add each pair of bits
- Include the carry in the addition, if present
- Discard the carry if it exceeds the available maximum number size (overflow)

carry		**1**	**1**	**1**	**1**				
	0	0	1	1	0	1	1	0	(54)
+	0	0	0	1	1	1	0	1	(29)
	0	1	0	1	0	0	1	1	(83)
bit position:	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	

Binary Subtraction

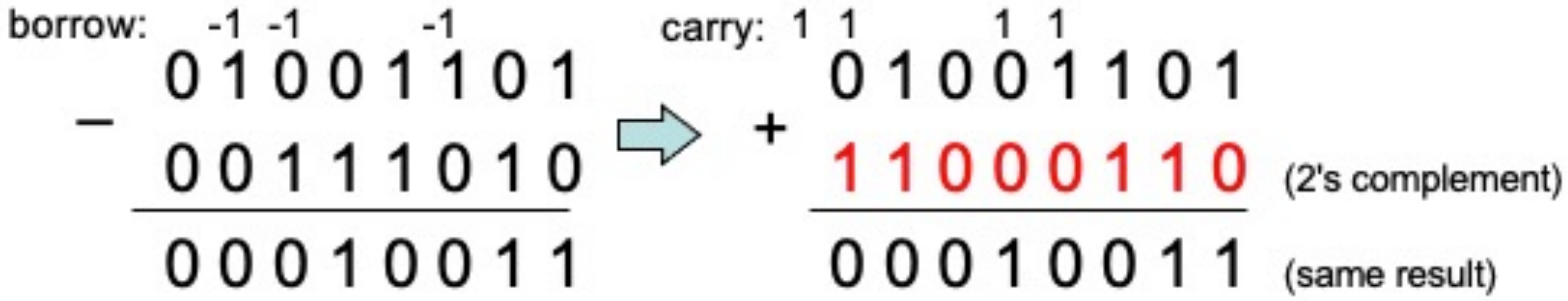
Start with the least significant bit (rightmost bit):

- Subtract each pair of bits
- Include the carry in the subtraction, if present

borrow			-1	-1			-1		
	0	0	1	1	0	1	1	0	(54)
–	0	0	0	1	1	1	0	1	(29)
	0	0	0	1	1	0	0	1	(25)
bit position:	7	6	5	4	3	2	1	0	

Binary Subtraction using 2’s Complement

- When subtracting A – B, convert B to its 2's complement
- Add A to (2’s complement of B)



2's Complement Addition with Overflow

- If the addition result overflows the +ve range within the binary number size, it gives an incorrect answer
- Example: add $4_{10} + 5_{10}$ using 4-bit 2's complement numbers

$$(0100)_2 + (0101)_2 = (1001)_2 = (-7)_{10}$$

- Solution: Sign Extension

- When a two's complement number is extended to more bits, the sign bit must be copied into the most significant bit positions. This process is called sign extension.

Sign Extension

- Step 1: Move the number into the lower-significant bits
- Step 2: Fill the remaining higher bits with the sign bit
- Example: sign-extend 10110011_2 to 16 bits

$$10110011_2 = -77 \quad \Rightarrow \quad \boxed{11111111}\ 10110011 = -77$$

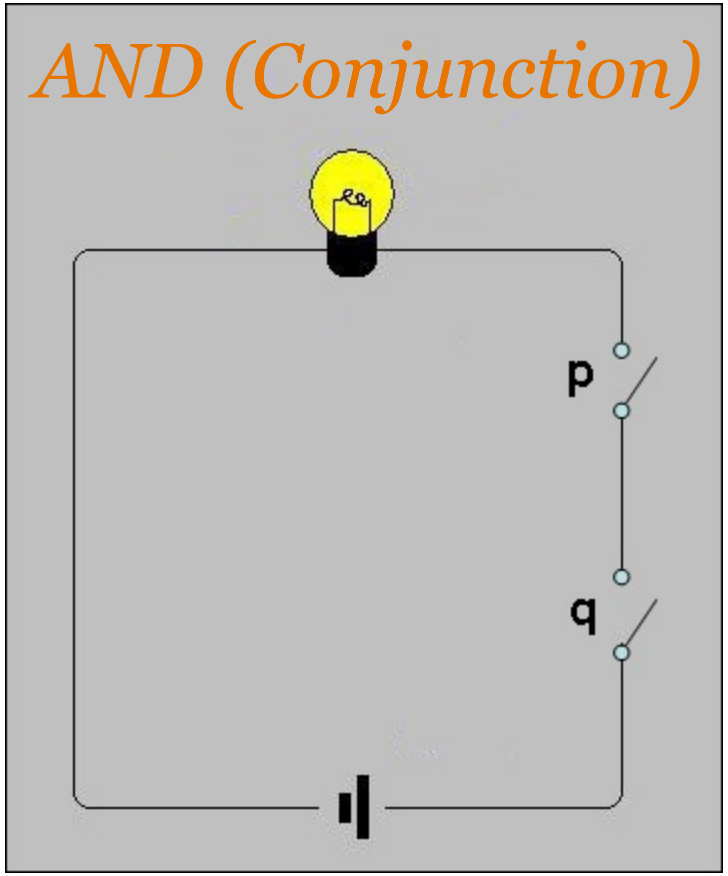
Binary Logic and Gates

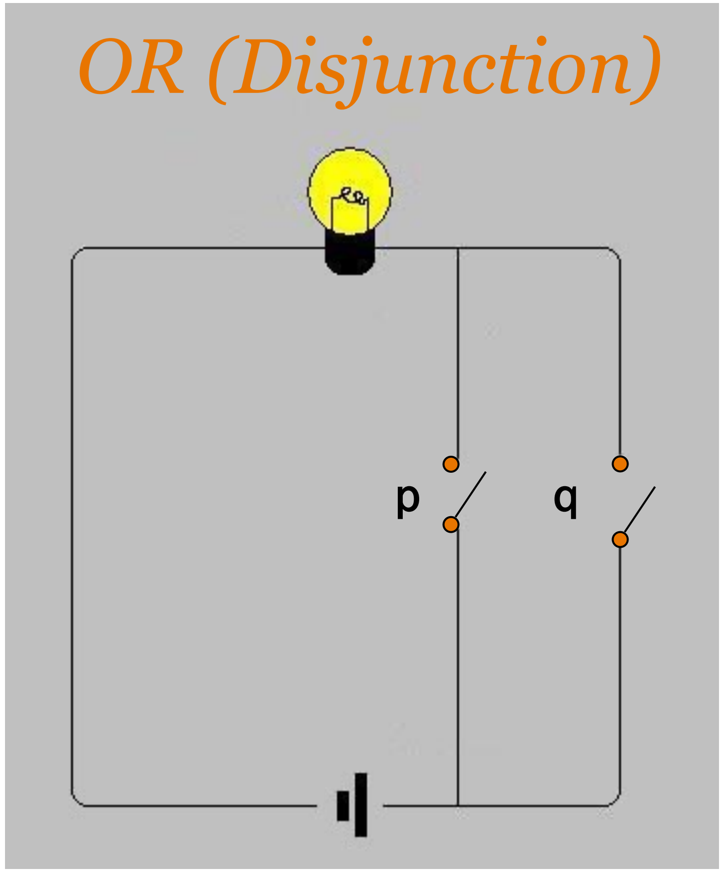
- *Binary variables: take on one of two values (0, 1)*
- *Logical operators: operate on binary values and variables*
- *Basic operators: AND, OR, NOT*
- *Boolean algebra: A useful mathematical system for specifying and transforming logic functions.*
- *Boolean algebra is the foundation of digital design*

Logical Operators

- ***AND: is denoted by:***
  - X AND Y = X.Y = XY
- ***OR: is denoted by:***
  - X OR Y = X+Y
- ***NOT: is denoted by:***
  - NOT X = ~X = X' = $\overline{X}$

Conjunction (AND) vs Disjunction (OR)





Basic Operator Definitions

- *Operations are defined on the values "0" and "1" for each operator:*

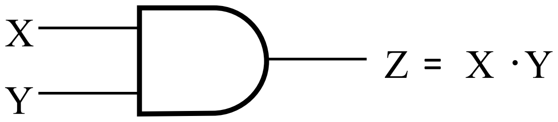
AND		OR		NOT	
$0 \cdot 0$	0	$0 + 0$	0	$\overline{0}$	1
$0 \cdot 1$	0	$0 + 1$	1	$\overline{1}$	0
$1 \cdot 0$	0	$1 + 0$	1		
$1 \cdot 1$	1	$1 + 1$	1		

Truth Table

- *Truth table is a tabular listing of the values of a function for ALL possible combinations of values on its arguments*
- *Truth table is a useful tool to study the behavior of any Boolean function*

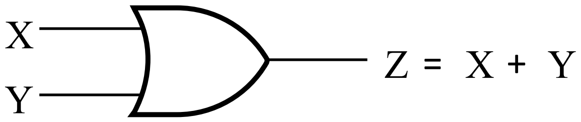
Basic Logic Gates and Truth Table

➢ *Logic gates are simple digital circuit that implements the logical operators*



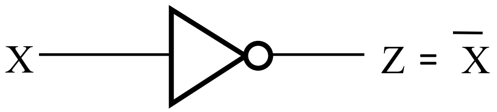
AND gate

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1



OR gate

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

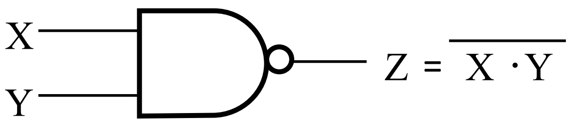


NOT gate or inverter

X	Z
0	1
1	0

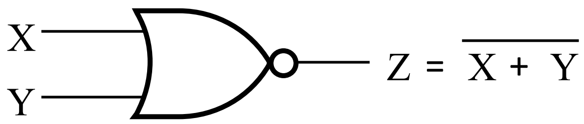
Other Logic Gates

- *The following other logic gates are useful in optimizing complex circuit designs*



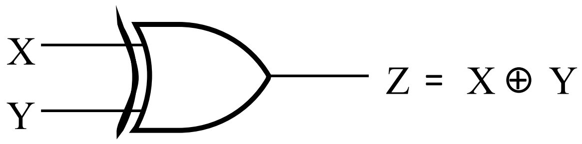
NAND gate

X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0



NOR gate

X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0



XOR gate

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

Multiple Variables & Mixed Gates

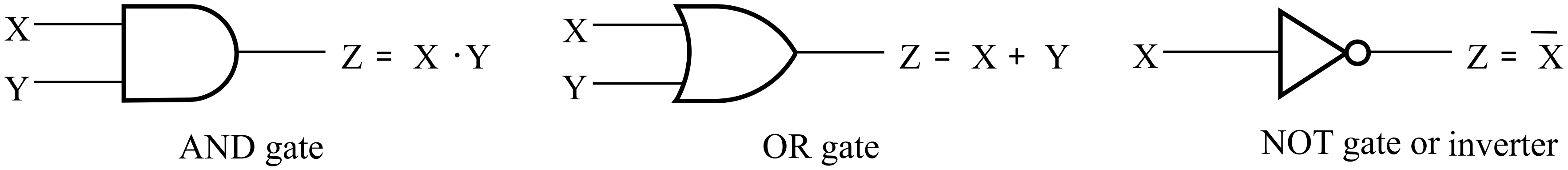
- *Studying the behavior of multiple variables and mixed gates follows different simplification techniques.*
- *This is possible using Boolean algebra postulates and rules.*
- *In general, you need to:*
  - represent all input combinations
  - segment the design (Boolean expression) into simple gates
  - combine the results to find the final circuit behavior

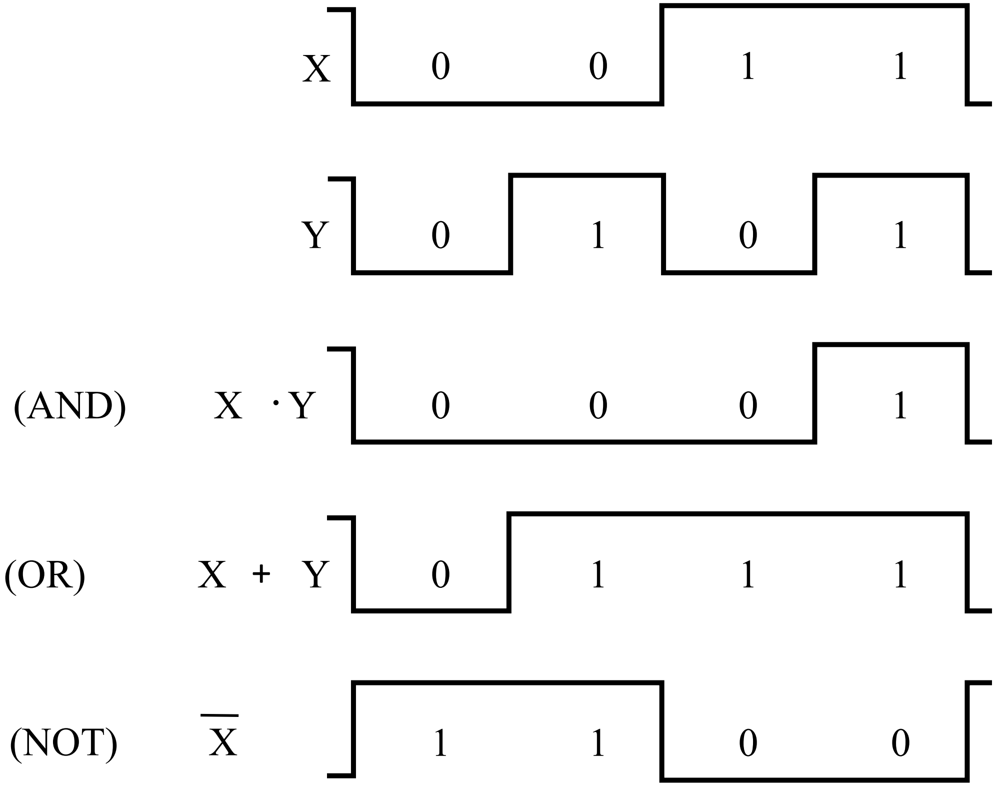
Example

- *Evaluate the following logic function:* $F(X, Y, Z) = X\,Y + \overline{Y}\,Z$

X	Y	Z	XY	$\overline{Y}$	$Z\overline{Y}$	$XY + Z\overline{Y}$
0	0	0	0	1	0	0
0	0	1	0	1	1	1
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	0	1	0	0
1	0	1	0	1	1	1
1	1	0	1	0	0	1
1	1	1	1	0	0	1

Waveform Behavior View





To Do List

- Review lecture notes
- Study 1.1 – 1.5
- Read 1.6
- Solve practice problems from chapter 1