Digital Design

Ms. Umarah Qaseem

1

Home Tasks

- Are you all added on Teams?
- Have you all downloaded the book on cellphones?
- It will be marked in the next class/lab.

2

2

1

Chapter 1

Digital Systems and Binary Numbers

3

2

Lecture 2

Binary Arithmetic, Compliments, Addition through compliments, Binary Codes, Error Detection, Parity, Registors, Logic Gates

4

Arithmetic - Addition

- Binary addition is similar to decimal arithmetic

No carries

	0	1	1	0	0
+	1	0	0	0	1
	1	1	1	0	1

	1	0	1	1	0	0	Carries
		1	0	1	1	0	
+		1	0	1	1	1	
	1	0	1	1	0	1	

1+1 is 2 (or 10_2), which results in a carry

5

Arithmetic - subtraction

No borrows

	1	0	1	1	0
-	1	0	0	1	0
	0	0	1	0	0

	0	0	1	1	0	Borrows
	1	1	1	1	0	
-	1	0	0	1	1	
	0	1	0	1	1	

0 - 1 results in a borrow

Borrow makes it $(10)_2 = (2)_{10}$

6

Arithmetic -- Multiplication

			1	0	1	1
		x		1	0	1
			1	0	1	1
		0	0	0	0	
	1	0	1	1		
1	1	0	1	1	1	

7

Complements

- Conventional addition (using carry) is easily implemented in digital computers.
- However; subtraction by borrowing is difficult and inefficient for digital computers
- Much more efficient to implement subtraction using ADDITION of the COMPLEMENTS of numbers

8

Two types of Complements of Radix R

- R's Complement
  - In Binary 2's complement
  - In Decimal 10's complement
- (R-1) Complement aka Diminished Radix
  - In Binary 1's complement
  - In Decimal 9's Complement

9

Summary: How to take Compliments

- **9's Compliment**: Subtract each digit from 9
- **1's Compliment**: Subtract each digit from 1
- OR Invert each bit.

- **10's Compliment**: Take 9's Compliment and add 1
- **2's Compliment**: Take 1's Compliment and add 1
- OR Keep the last 1 and invert the rest of bits

10

10

Summary: PROOF With the Formulae

- Given a number *N* in base *r* having *n* digits
- **r's Compliment =**

  $\boxed{(r^n - N)}$

- **R-1's Compliment =**

  $\boxed{(r^n - 1) - N}$

11

11

Diminished Radix (r-1) Complement

$\boxed{(r^n - 1) - N}$

- Example: Take the number N = 546700
- Here n=6 so put value in formula
- $= (10^6 - 1) - 546700$
- $= (1000000 - 1) - 546700$
- = 999999 – 546700
- = 453299 So, 9's complement is 453299
- In simple words subtract each digit from 9

12

Diminished Radix (r-1) Complement - Activity

- 9’s Complement
- Find the 9’s complement of
- 546700 and 12389

The 9’s complement of 546700 is
999999 - 546700= 453299

And the 9’s complement of 12389 is
99999- 12389 = 87610.

	9	9	9	9	9	9
-	5	4	6	7	0	0
	4	5	3	2	9	9

		9	9	9	9	9
-		1	2	3	8	9
		8	7	6	1	0

13

Diminished Radix (r-1) Complement – Base 2

$(r^n - 1) - N$

- Example: Take the number N = 1010
- Here n=4, then put values in formula
- $(2^4 - 1) - 1010$ Note: $(15)_{10} = (1111)_2$
- $= (16 - 1)_{10} - (1010)_2$
- $= (1111)_2 - (1010)_2$
- Note: 1-0 =1 and 1-1 =0 (Bit Changes)
- In simple words just change the bits

14

Diminished Radix (r-1) Complement - Activity

- Find out the 1's Complement of 1011001 & 0001111

	1	1	1	1	1	1	1
−	1	0	1	1	0	0	1
	0	1	0	0	1	1	0

	1	1	1	1	1	1	1
−	0	0	0	1	1	1	1
	1	1	1	0	0	0	0

The complement 1's of 1011001 is 0100110

The 1's complement of 0001111 is 1110000

15

Radix (r) Complement - 10's Complement

(r^n –N)

- Find the 10's complement of 546700 and 12389
- The 10's complement of 546700 is
- 1000000 - 546700= 453300
- The 10's complement of 12389 is
- 100000 - 12389 = 87611.

	1	0	0	0	0	0	0
−		5	4	6	7	0	0
		4	5	3	3	0	0

	1	0	0	0	0	0
−		1	2	3	8	9
		8	7	6	1	1

Notice that it is the same as 9's complement + 1.

17

Radix (r) Complement - 2's Complement

- Find the 2's complement of 1011001 & 0001111

	1	0	0	0	0	0	0	0
–		1	0	1	1	0	0	1
		0	1	0	0	1	1	1

	1	0	0	0	0	0	0	0
–		0	0	0	1	1	1	1
		1	1	1	0	0	0	1

The 2's complement of 1011001 is 0100111

The 2's complement of 0001111 is 1110001

18

Fast Computation of 2's Complement

- **Method 1:**
- The 2's complement of binary number is obtained by adding 1 to the 1's complement value.

Example:
1's complement of 101100 is 010011 (invert the 0's and 1's)
2's complement of 101100 is 010011 + 1 = 010100

19

Fast Computation of 2's Complement

- **Method 2**
- The 2's complement can be formed by leaving all least significant 0's and the first 1 unchanged, and then replacing l's by 0's and 0's by l's in all other higher significant bits.

- Example:
- The 2's complement of 1101100 is
- 0010100
- Leave the two low-order 0's and the first 1 unchanged, and then replacing 1's by 0's and 0's by 1's in the four most significant bits.

20

Subtraction with r-Complement

Subtract N from M : M – N

r's complement of N = r^n – N
Add M to r's complement
Result = M + (r^n – N)

(1) if M ≥ N, simply ignore the carry
(2) if M < N, The answer is negative. Take the r's complement of sum and place negative sign in front of sum.

21

Example 1 M>N

Perform the subtraction 72532 - 13250 = 59282.

M > N : Case 1 Discard carry

The 10's complement of 13250 is 86750. Therefore:

	M =	72532
10's complement of	*N* =	+86750
	Sum =	159282

Discard end carry

Answer = 59282

22

Example 2: M<N --- Case 2

Perform the subtraction 13250 - 72532 = -59282.

	M =	13250
10's complement of	*N* =	+27468
	Sum =	40718

Now Take 10's complement of Sum. That means: 100000 – 40718

= 59282

Place negative sign in front of the number: -59282

23

Exercise

- Subtract (3250 – 72532) using 10's Complement.

  M = 03250

  N = 72532

24

Exercise

- Using 2's complement subtract 1000100 from 1010100

  M = 1010100

  N = 1000100

26

Exercise

- Subtract (1000100 – 1010100) using 2's Complement

  M = 1000100

  N = 1010100

28

Signed Binary Numbers

- It is usual to represent the sign with a bit placed in the leftmost position of the binary number

Sign bit

The Most common notations are:

- **Signed-magnitude system.**
- **Signed-complement system.**

Sign bit 0 positive
Sign bit 1 negative

30

Signed Magnitude System

Example 1

01001 → + 9
11001 → – 9

Example 2

3-bit binary pattern

Bit Pattern	000	001	010	011	100	101	110	111
Signed-Magnitude Decimal Value	+0	+1	+2	+3	-0	-1	-2	-3

31

Signed Complement System

- In this system, a negative number is indicated by its complement.
- The signed-complement system can use either the 1's complement or the 2's complement notations

32

Example

- Assuming the representation of the number 9 in binary with 8-bits, we have the following cases:
- Unsigned 9 or +9 has the same representation in both signed-magnitude and signed-complement systems which is: 00001001
- -9 has the signed-magnitude representation: 10001001
- -9 has the signed-1's complement representation: 11110110
- -9 has the signed-2's complement representation: 11110111

33

Signed Complement System

- The signed-complement conversion table of a 3-bit binary pattern is as follows:

Bit Pattern	000	001	010	011	100	101	110	111
Signed 1's complement decimal value	+0	+1	+2	+3	-3	-2	-1	-0
Signed 2's complement decimal value	+0	+1	+2	+3	-4	-3	-2	-1

34

Signed Complement System

4-bit system

- Positive Values

Value	Sign-and-Magnitude	1s Comp.	2s Comp.
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000

- *Negative values*

Value	Sign-and-Magnitude	1s Comp.	2s Comp.
-0	1000	1111	-
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1110	1001	1010
-7	1111	1000	1001
-8	-	-	1000

35

Complement of Fractions

- We can extend the idea of complement on fractions.
- Examples:
  - Negate 0101.01 in 1s-complement
    Answer: 1010.10
  - Negate 111000.101 in 1s-complement
    Answer: 000111.010
  - Negate 0101.01 in 2s-complement
    Answer: 1010.11

36

Binary Codes

The 8421 BCD

- BCD stands for Binary-Coded Decimal.
- A BCD number is a four-bit binary group that represents one of the ten decimal digits 0 through 9

Decimal Digit	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Note: 1010, 1011, 1100, 1101, 1110, and 1111 are **INVALID CODE!**

BCD: *Example:*

- Convert Decimal to BCD

	4	9	2	6
Decimal number 4926	4	9	2	6
8421 BCD coded number	0100	1001	0010	0110

39

Activity

- Convert the BCD coded number
- 1000 0111 0001 into decimal.

BCD Coded Number	1000	0111	0001
Decimal Number	8	7	1

40

BCD

- People understand decimal system better
- BCD makes it easy to replace a decimal number with an individual binary code
- Decimal 15 is BCD 0001 0101 in Binary it was 1111
- Since most computers store data in eight-bit bytes
  - Ignore 4 extra bits
  - One can store two digits per byte, called "packed" BCD

41

BCD Addition

- BCD is a numerical code and can be used in arithmetic operations. Here is how to add two BCD numbers:

1. Add the two BCD numbers, using the rules for basic binary addition.
2. If a 4-bit sum is equal to or less than 9, it is a valid BCD number.
3. If a 4-bit sum > 9, or if a carry out of the 4-bit group is generated it is an invalid result. Add 6 (0110) to a 4-bit sum in order to skip the six the invalid states and return the code to 8421. If a carry results when 6 is added, simply add the carry to the next 4-bit group.

42

BCD Addition - Examples

- 4+5

	0	1	0	0
+	0	1	0	1
	1	0	0	1

- 4+8

		0	1	0	0
+		1	0	0	0
		1	1	0	0	← Sum is greater than 1001
+		0	1	1	0	← Add $(0110)_2$ or $(6)_{10}$
	1	0	0	1	0

43

BCD Addition – Examples - Activity

- 184+576

				1					1							
	0	0	0	1		1	0	0	0		0	1	0	0		184
+	0	1	0	1		0	1	1	1		0	1	1	0		576
	0	1	1	1	1	0	0	0	0		1	0	1	0		
						0	1	1	0	+	0	1	1	0		
	0	1	1	1		0	1	1	0		0	0	0	0		760

44

The Excess-3 Code

- Add 3 to each digit of decimal and convert to 4-bit binary form

Decimal	Binary	+3	Excess-3
0	0000	0011	0011
1	0001	0011	0100
2	0010	0011	0101
3	0011	0011	0110
4	0100	0011	0111
5	0101	0011	1000
6	0110	0011	1001
7	0111	0011	1010
8	1000	0011	1011
9	1001	0011	1100

Sample Problem:

Decimal 3 5 9

Excess-3 0110 1000 1100

45

Some Weighted Codes

- Weight codes – Different position has different weight
- 8421 code 0101=0x8+1x4+0x2+1x1=5
- 84-2-1 0101=0x8+1x4+0x(-2)+1x(-1)=3

Four Different Binary Codes for the Decimal Digits

Decimal Digit	BCD 8421	2421	Excess-3	8, 4, −2, −1
0	0000	0000	0011	0000
1	0001	0001	0100	0111
2	0010	0010	0101	0110
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1011
6	0110	1100	1001	1010
7	0111	1101	1010	1001
8	1000	1110	1011	1000
9	1001	1111	1100	1111

47

Warning: Conversion or Coding

- Do NOT mix up conversion of a decimal number to a binary number with coding a decimal number with a BINARY CODE.
- $13_{10} = 1101_2$ (This is conversion)
- 13 ⇔ 0001|0011 (This is coding)

51

Character Codes - ASCII

- Many applications require handling of not only numbers but letters and special characters
- ASCII – American Standard Code for Information Interchange
- 7 Bits to store 128 characters
- In ASCII, every letter, number, and punctuation symbol has a corresponding number, or ASCII code

52

ASCII Table

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
64	40	@	78	4E	N	96	60	`	109	6D	m
65	41	A	79	4F	O	97	61	a	110	6E	n
66	42	B	80	50	P	98	62	b	111	6F	o
67	43	C	81	51	Q	99	63	c	112	70	p
68	44	D	82	52	R	100	64	d	113	71	q
69	45	E	83	53	S	101	65	e	114	72	r
70	46	F	84	54	T	102	66	f	115	73	s
71	47	G	85	55	U	103	67	g	116	74	t
72	48	H	86	56	V	104	68	h	117	75	u
73	49	I	87	57	W	105	69	i	118	76	v
74	4A	J	88	58	X	106	6A	j	119	77	w
75	4B	K	89	59	Y	107	6B	k	120	78	x
76	4C	L	90	5A	Z	108	6C	l	121	79	y
77	4D	M							122	7A	z

53

Error Detection

54

Error Detection

- Errors can occur during data transmission. They should be detected, so that re-transmission can be requested.
- With binary numbers, usually single-bit errors occur.
- Example: 0010 erroneously transmitted as 0011 or 0000 or 0110 or 1010.

55

Error Detection

- To detect errors, an eighth bit is sometimes added to the ASCII character to indicate its parity.
- We insert an extra bit in the leftmost position of the code
- Two types of parity

56

Even Parity

- Even parity – set bit to make number of 1's even
- Examples

  1000001 with even parity is 01000001

  1000011 with even parity is 11000011

57

Odd Parity

- Similar except make the number of 1's odd
- Examples

  1000001 with odd parity is 11000001

  1000011 with odd parity is 01000011

58

Error Detection

- Parity bit
  - Even parity: additional bit added to make total number of 1's even.
  - Odd parity: additional bit added to make total number of 1's odd.
- Example of odd parity on ASCII values.

Parity bits

Character	ASCII Code
0	1 0110000
1	0 0110001
. . .	. . .
9	1 0111001
:	1 0111010
A	1 1000001
B	1 1000010
. . .	. . .
Z	1 1011010
[	0 1011011
\	1 1011100

59

Error Detection Procedure

- The parity bit is helpful in detecting errors during the transmission of information from one location to another.
- This function is handled by generating an even parity bit at the sending end for each character.
- The eight-bit characters that include parity bits are transmitted to their destination.
- The parity of each character is then checked at the receiving end.
- If the parity of the received character is not even, then at least one bit has changed value during the transmission.

60

Error Detection

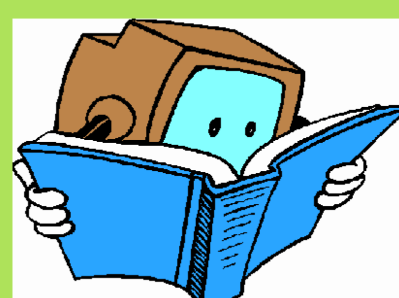
- Keeping even parity is more common.
- Parity bit can detect odd number of errors but not even number of errors.

- Example: Assume odd parity,
- 10011 → 10001 (detected)
- 10011 → 10101 (not detected)

61

Reading Assignment

Binary Storage and Registers



62

Binary Logic

63

Binary Logic

- Binary logic consists of binary variables and a set of logical operations.
-
- The binary variables can have only two values:
- 0 and 1.

64

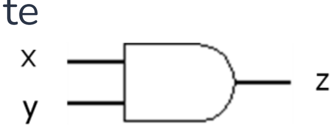
Binary Logic

- The basic logical operations are AND, OR, and NOT.
- For x and y binary variables the three logical operations are shown as:
  - AND operation . x.y
  - OR operation + x+y
  - NOT operation ' x'

65

The AND Operation

- The circuit represents AND operation and is called AND gate



- Two or more input bits produce one output bit.
- Both inputs must be true (1) for the output to be true.
- Otherwise the output is false (0).

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

Truth Table

66

The OR Operation

- The circuit represents OR operation and is called OR gate

x	y	z
0	0	0
0	1	1
1	0	1
1	1	1

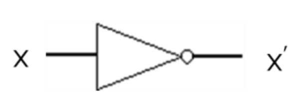
Truth Table

- Two or more input bits produce one output bit.
- Either inputs must be true (1) for the output to be true.

67

The NOT Operation - Inverter

- The circuit represents NOT operation and is called NOT gate or INVERTER



Symbol

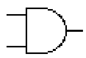
x	x’
0	1
1	0

Truth Table

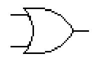
- One bit as input produces its opposite as output.

68

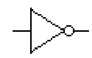
Truth Tables of Logic Operations



AND



OR



NOT

A	B	A & B
0	0	0
0	1	0
1	0	0
1	1	1

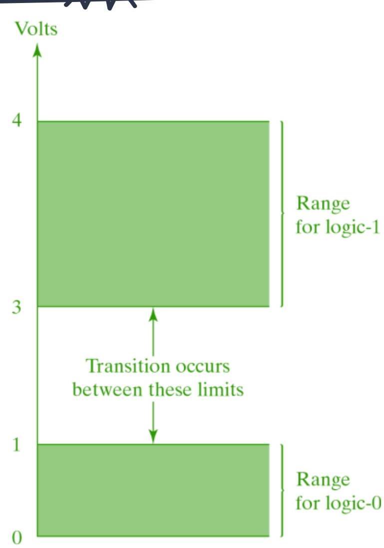
A	B	A | B
0	0	0
0	1	1
1	0	1
1	1	1

A	$\bar{A}$
0	1
1	0

69

Representation of Binary Variables

- Different Digital Systems represent 0 and 1 differently
  - Logical 0 as 0 volts. Logical 1 as 4 volts
  - Range



70

Binary Logic Functions

x	0	1	1	0	0
y	0	0	1	1	0
AND: $x \cdot y$	0	0	1	0	0
OR: $x + y$	0	1	1	1	0
NOT: x'	1	0	0	1	1

71

References

- Chapter 1 – Digital Design Morris Mano
- Digital Logic and Computer Design – M. Singh, University of North Carolina
- Digital Design – O. Ozturk, Bilknet University
- Template is taken from slides carnival.

72

Credits

These slides are adapted from Morris Mano Book 5th /6th Edition

Slides Carnival

73

73