WEEK 1-2018

Introduction/Number systems

Information representation - signals

- **Information represents physical parameters or man-made parameters.**

- **Most physical parameters are continuous- can take all possible values over a defined range.**

  Example – temperature, humidity

- **Man-made parameters are discrete – can take only finite possible values over a defined range.**

  Example – currency,

Information representation

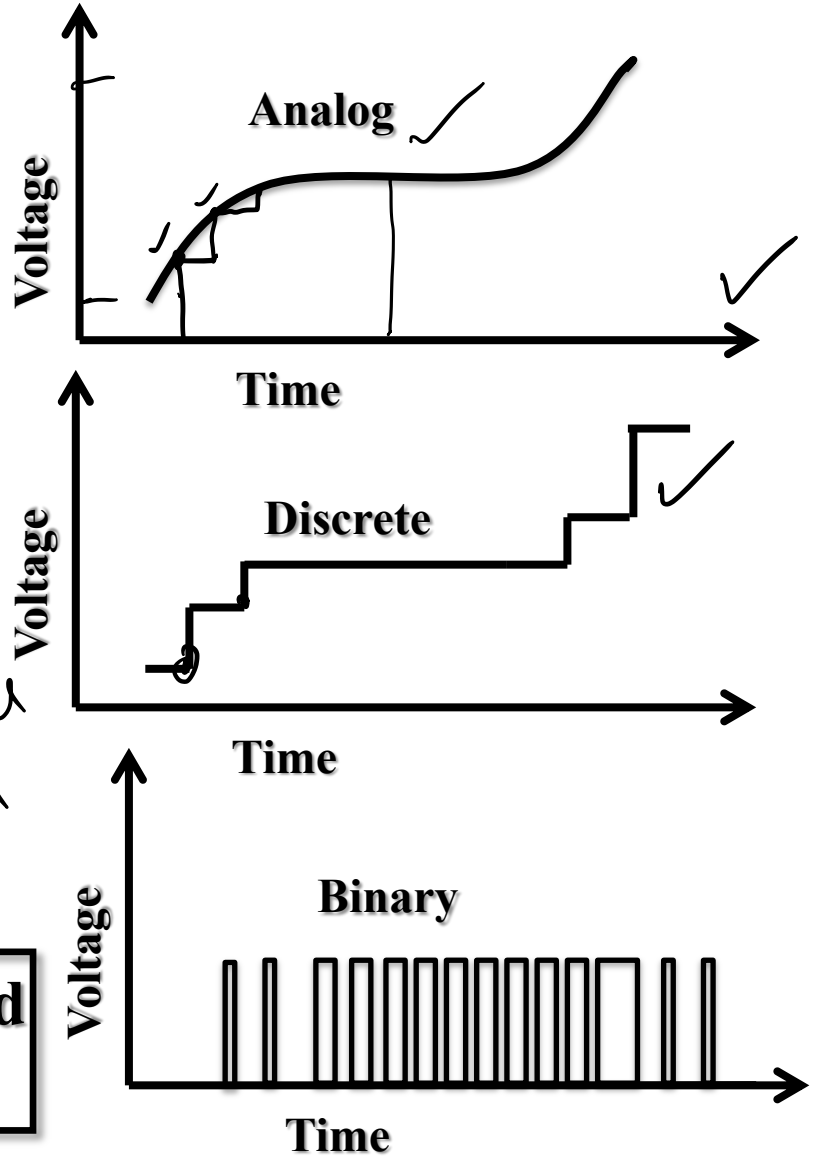
Physical parameter

Sensor

Analog voltage

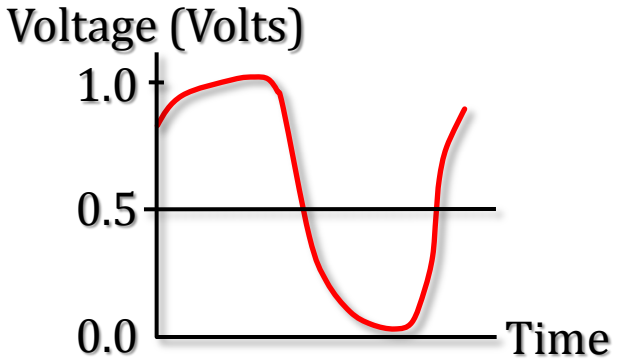
Discrete voltage

Sum of multiple of two discrete valued (binary) signal

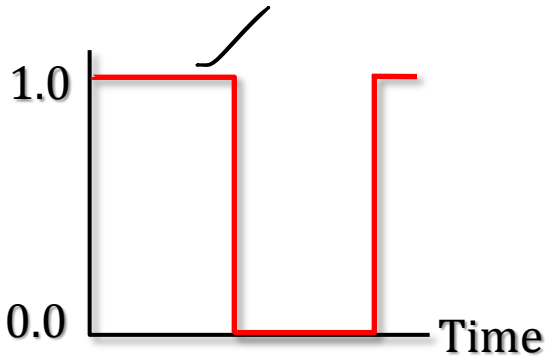


Digital circuits

- **Digital Circuits are circuits which can understand (manipulate and store) signals as only binary .**
- **Signals in digital circuits are in fact analog, but interpreted as binary**
- **How?**

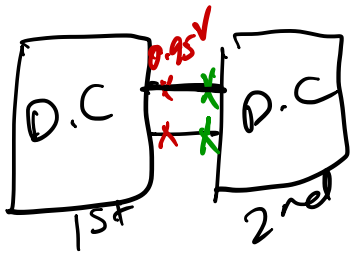


Time-dependent voltage

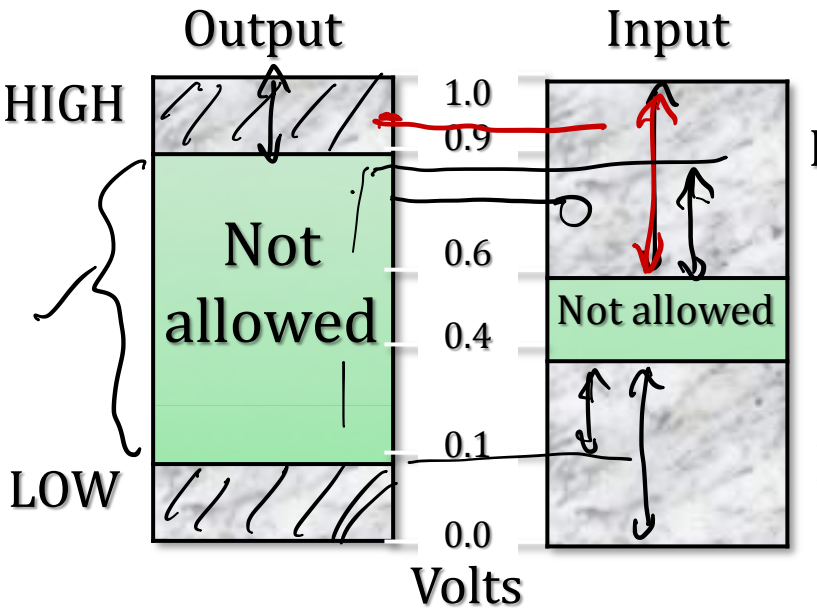


Binary model of Time-dependent voltage

Digital Circuits



- **Binary signals – (1,0) or (HIGH, LOW) or (TRUE,FALSE)**
- **Positive Logic: 1, HIGH, TRUE are equivalent as are 0, LOW, FALSE .**



Example voltage ranges

Binary signals are identified as HIGH or LOW by their range of values.

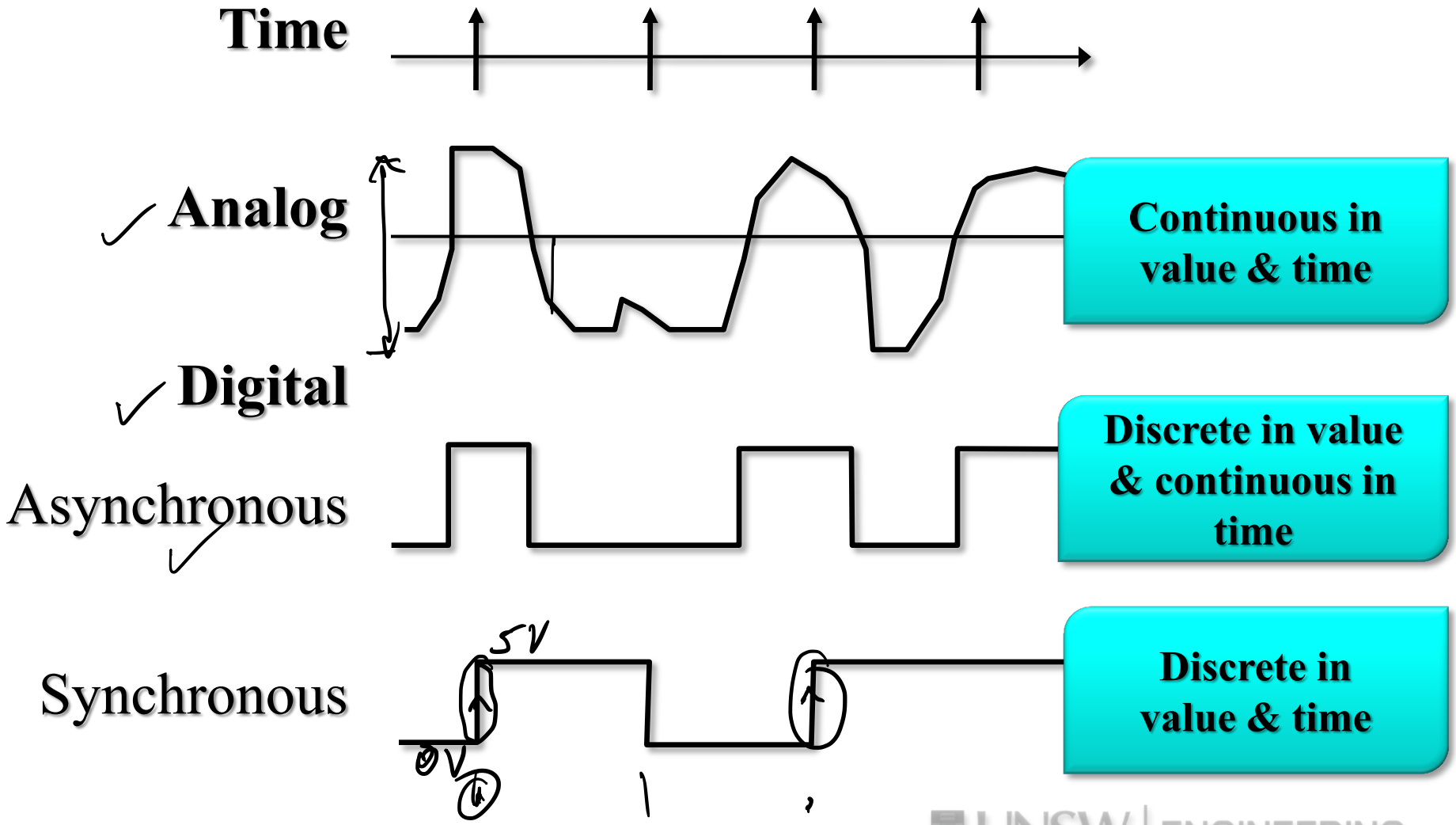
Eg. Output [0.9,1.0] – HIGH
Output [0,0.1]– LOW
Input [0.6 ,1.0] – HIGH
Input [0,0.4]- LOW

Noise margin – difference in range of values for input and output

Digital Circuits

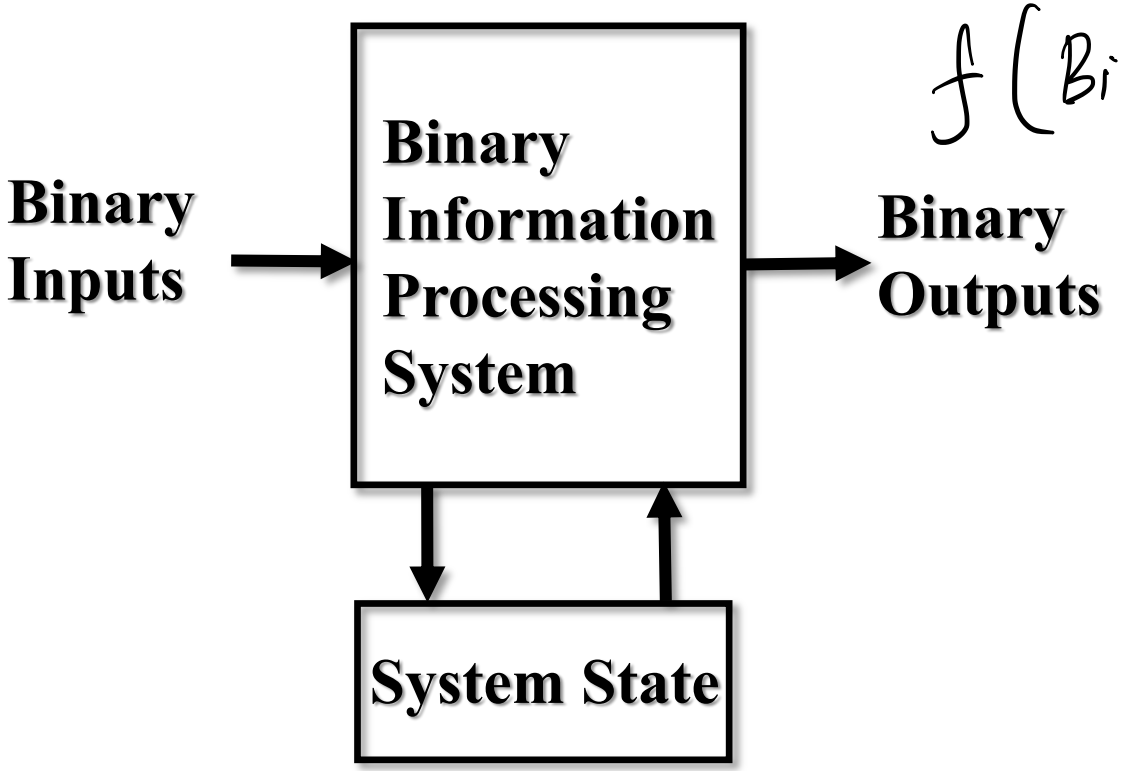
	TTL		3.3 V CMOS		5V CMOS	
	LOW	HIGH	LOW	HIGH	LOW	HIGH
INPUT	0 – 0.8	2 - 5	0 - 0.8	2 – 3.3	0 – 1.5	2.7 - 5
OUTPUT	0 – 0.5	2.7 - 5	0 – 0.4	2.4 –3.3	0 – 0.5	3.5 - 5
Noise Margin	0.3 ✓	0.7 ✓	0.4	0.4	1.0 ✓	0.8 ✓

Signals Over Time Example



Digital circuits

- **Digital circuits have binary inputs, process binary signals, store binary signals (states), and generate binary outputs.**



f (Binary input, state)

Types of Digital Circuits

- **No state present**
  - **Combinational logic circuits**
- **State present**
  - **Sequential logic circuits**
    - **State updated at discrete times**
      => Synchronous Sequential circuits
    - **State updated at any time**
      => Asynchronous Sequential circuits

output = f(input, state)

Mealy

Moore

output = f(state)

Adder

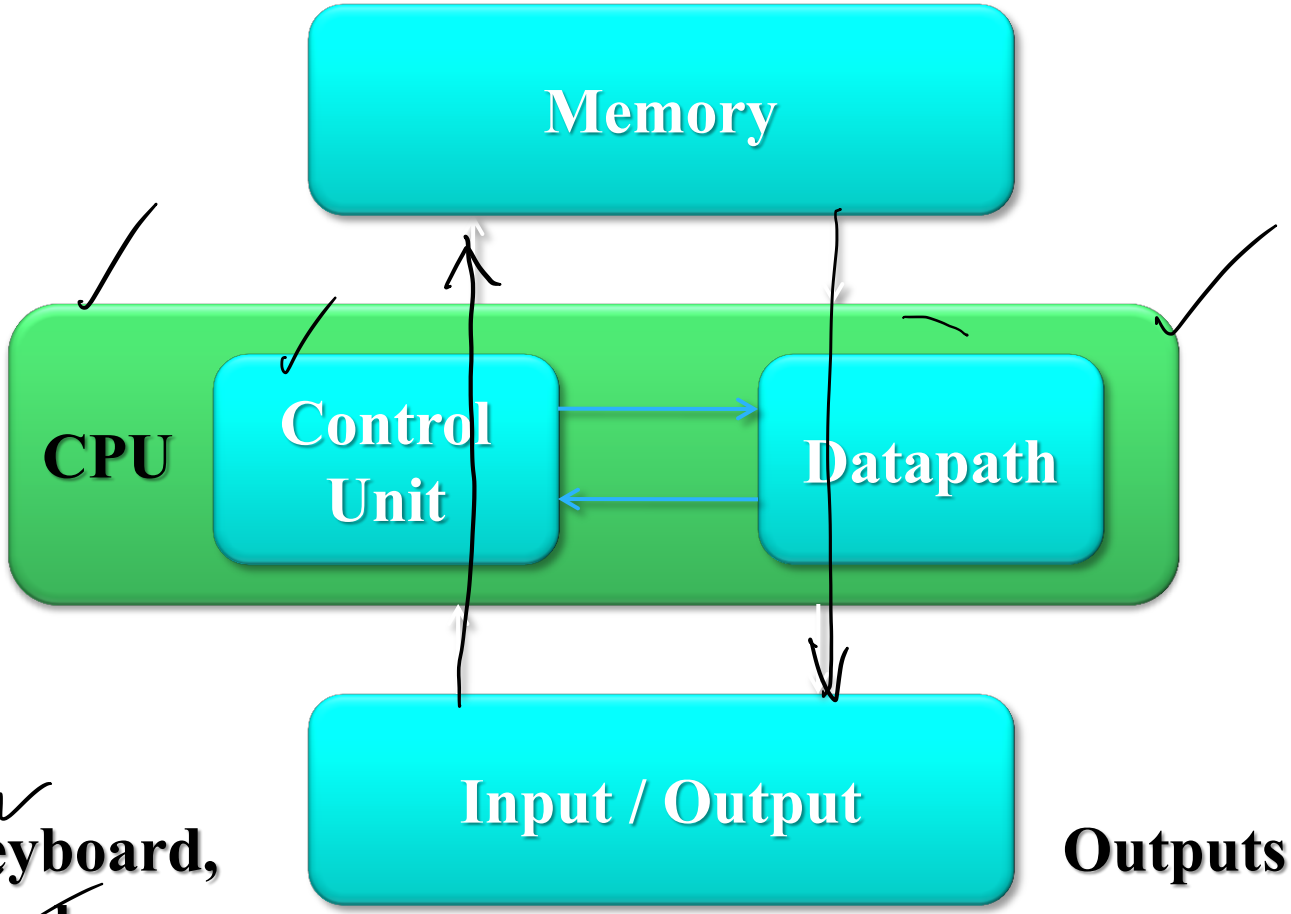
Digital System Example

A Digital Counter:

Count Up
Reset

Inputs: Count Up, Reset
Outputs: Visual Display
State: "Value" of stored digits ✓

Digital Computer Example



Inputs: keyboard, mouse, wireless, microphone

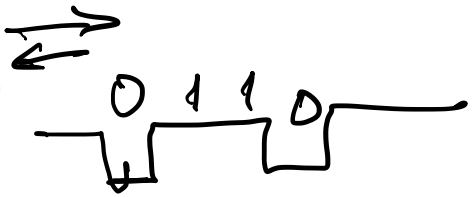
Outputs: LCD screen, wireless, speakers

And Beyond – Embedded Systems

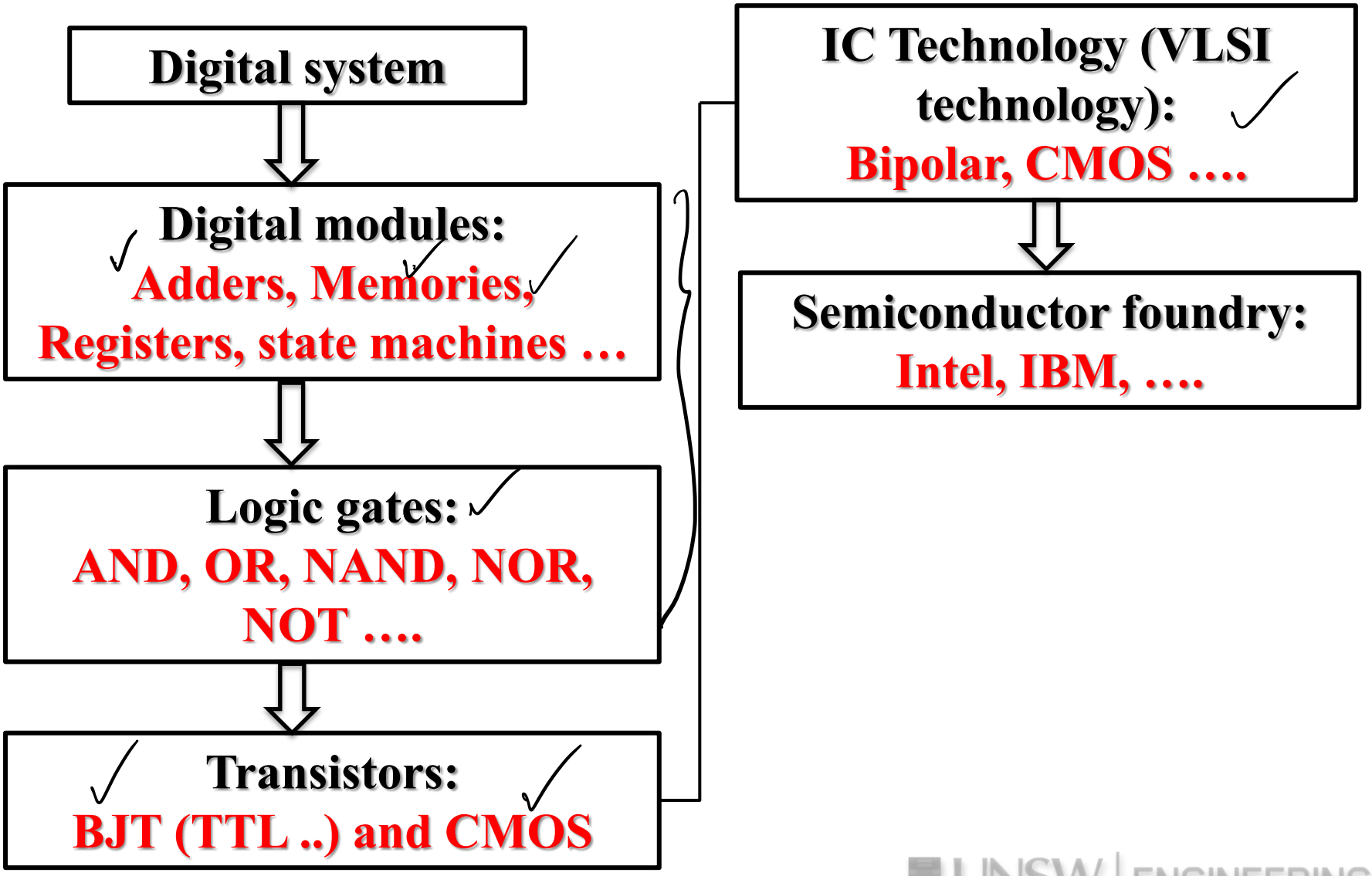
- Computers as integral parts of other products
- Examples of embedded computers
  - Microcomputers
  - Microcontrollers
  - Digital signal processors

Binary Values: Other Physical Quantities

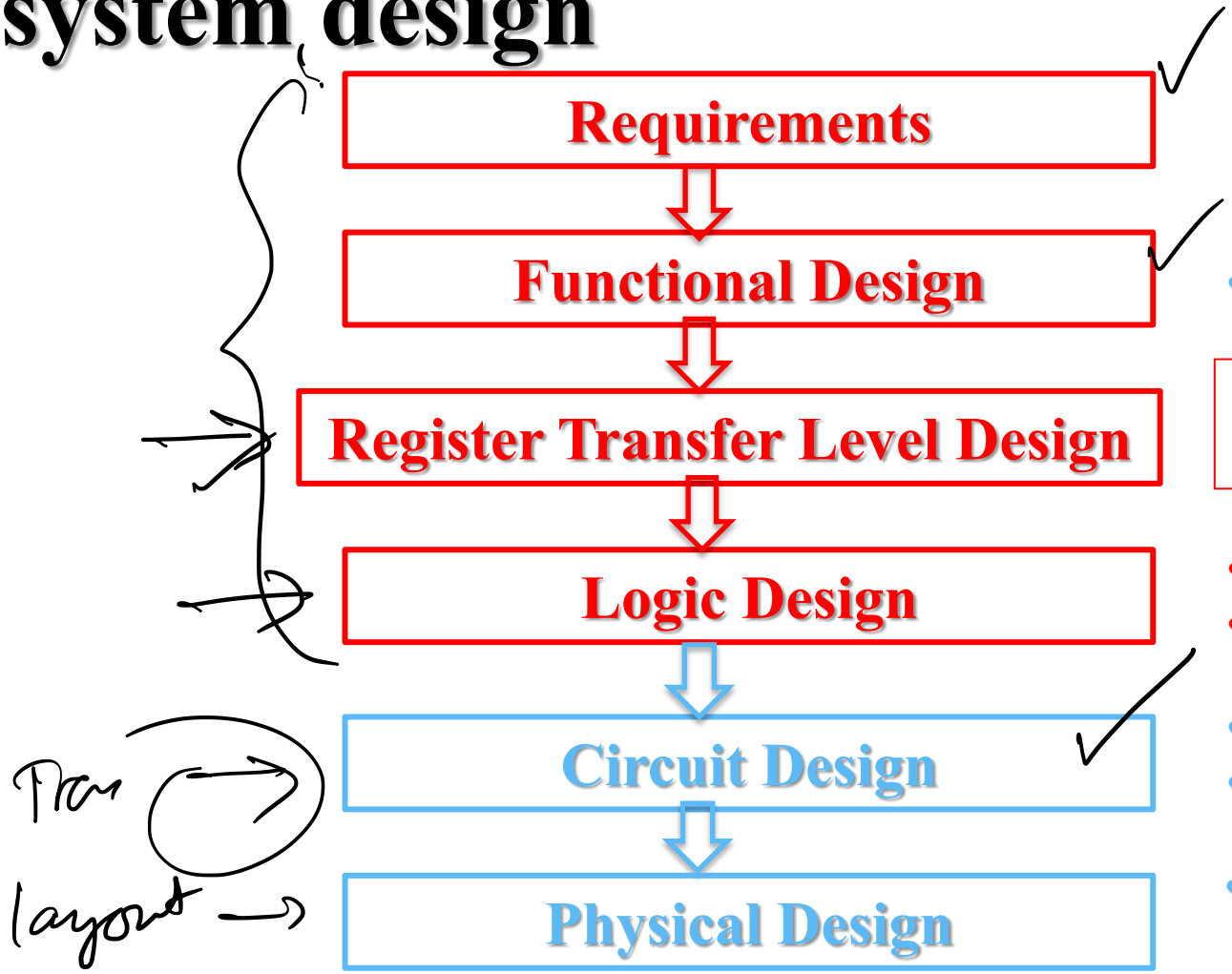
- **Example of some other physical representations for 0 and 1:**
  - **CPU:** *Voltage*
  - **Disk:** *Magnetic Field Direction*
  - **CD:** *Surface Pits*
  - **Dynamic RAM:** *Electrical Charge*



Hierarchy in a digital system

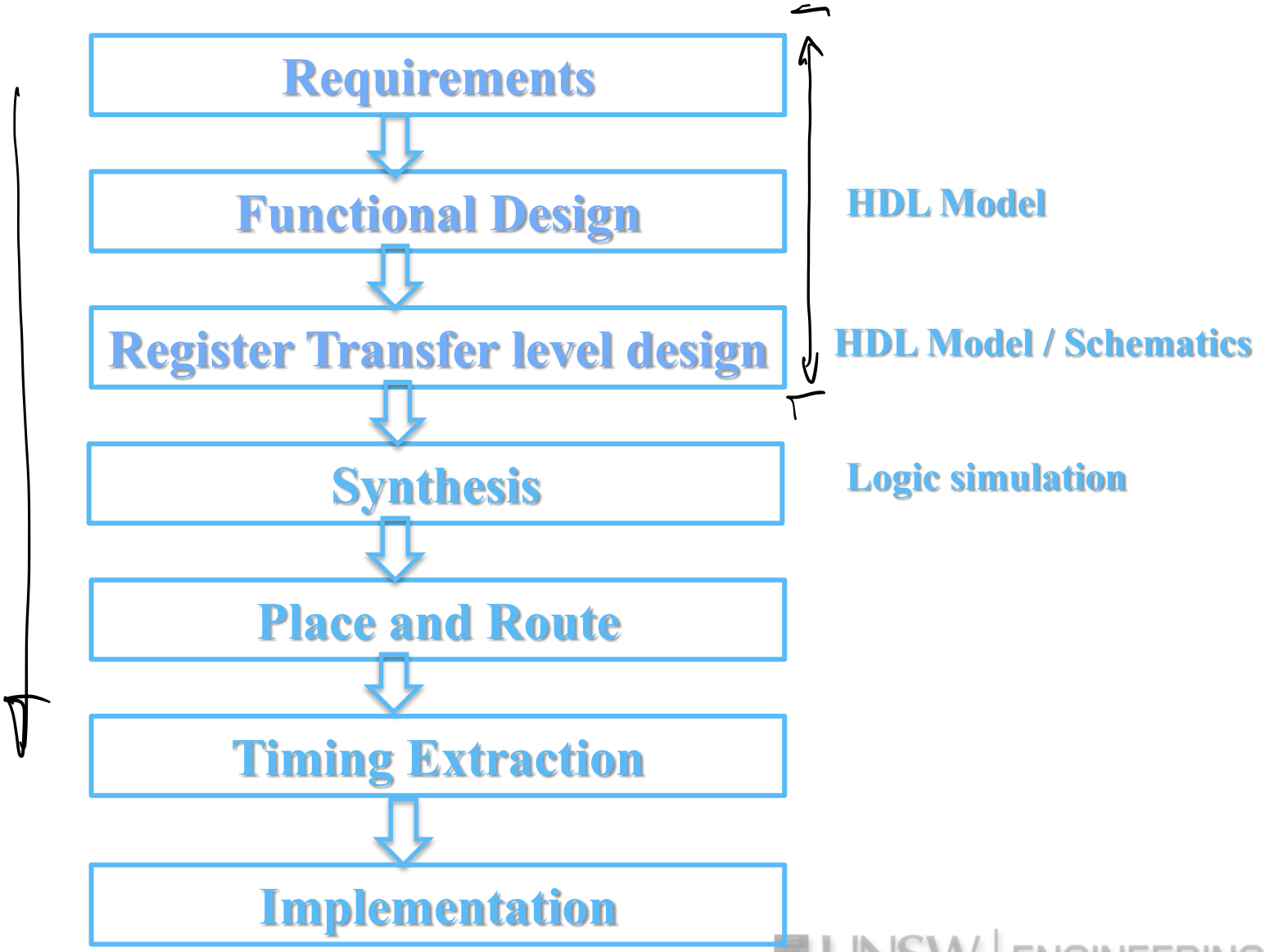


Design activity flow in top-down digital system design



- Behavioral simulation
- RTL simulation validation
- Logic simulation
- Verification
- Timing simulation
- Circuit Analysis
- Design rule check

Synthesis design flow for FPGAs



Number Systems

- Consider a decimal number 724.5

$$724.5 = 7\times10^{2} + 2\times10^{1} + 4\times10^{0} + 5\times10^{-1}$$

- A decimal number with n digits to the left of the decimal point and m digits to the right of the decimal point is represented by a string of coefficients as

$$A_{n-1}A_{n-2}\ldots A_1A_0.A_{-1}A_{-2}\ldots A_{-m+1}A_{-m} = A_{n-1}10^{n-1} + A_{n-2}10^{n-2} + \ldots + A_0 10^0 + A_{-1}10^{-1} + \ldots + A_{-m}10^{-m}$$

- Each coefficient A_i can be one of 10 digits (0,1,2,3,4,5,6,7,8,9)
- Generally, a number in base or radix r with n-digits to the left of the radix point and m-digits to the right of the radix point is represented by a string of coefficients, and is expressed as a power series in r with the general form

Number Systems

$$(A_{n-1}A_{n-2}\ldots A_1A_0.A_{-1}A_{-2}\ldots A_{-m+1}A_{-m})_r = A_{n-1}r^{n-1} + A_{n-2}r^{n-2} + \ldots + A_0r^0 + A_{-1}r^{-1} + \ldots + A_{-m}r^{-m}$$

where

Each coefficient A_i can assume a value of 0,1,2…r-1

A_{n-1} is the Most Significant Digit (MSD)

A_{-m} is the Least Significant Digit (LSD)

Eg.

$$(312.4)_5 = 3x5^2 + 1x5^1 + 2x5^0 + 4x5^{-1} = 75 + 5 + 2 + 0.8$$

$$= (82.8)_{10}$$

base 4, 0, 1, 2, 3

Binary Numbers

- Binary (base 2) numbers are widely used in digital systems
- Base 2 system only has two possible digits: 0 and 1
- Digits in binary numbers are called *"bits"* (Binary Digits)
- The rightmost bit in a binary number is referred to as the *Least Significant Bit* (LSB)
- Similarly, the leftmost bit is referred to as the *Most Significant Bit* (MSB)

Binary Numbers

- **Example:**

$$\overset{4\,3\,2\,1\,0}{11010_2} = 1\times2^4 + 1\times2^3 + 1\times2^1 = 16 + 8 + 2 = 26$$

- **In practice, when calculating binary values, omit all 0-bits and add values for 1-bits**
- **Example (with fractions):**

$$\overset{5\,4\,3\,2\,1\,0\,-1\,-2}{110101.11_2} = 1\times2^5 + 1\times2^4 + 1\times2^2 + 1\times2^0 + 1\times2^{-1} + 1\times2^{-2}$$

$$= 32 + 16 + 4 + 1 + 0.5 + 0.25$$

$$(53.75)_{10}$$

Positive Powers of 2

- **Useful for base conversion**

Important to remember!

Exponent	Value
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1,024

2^1

2^{10}

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

2^{15}

Memorize if you can (and have too much time on your hands)

Special Powers of 2

- 2^{10} (1,024) is Kilo, denoted *"K"*
- 2^{20} (1,048,576) is Mega, denoted *"M"*
- 2^{30} (1,073,741,824) is Giga, denoted *"G"*
- 2^{40} (1,099,511,627,776) is Tera, denoted *"T"*

Conversion of Decimal to Binary

- **Repeatedly divide integers by 2 to obtain quotient and remainder**
- **Read remainders in reverse order**
- **Example – convert 41_{10} to binary:**

$1 = 2 \times 0 + 1$

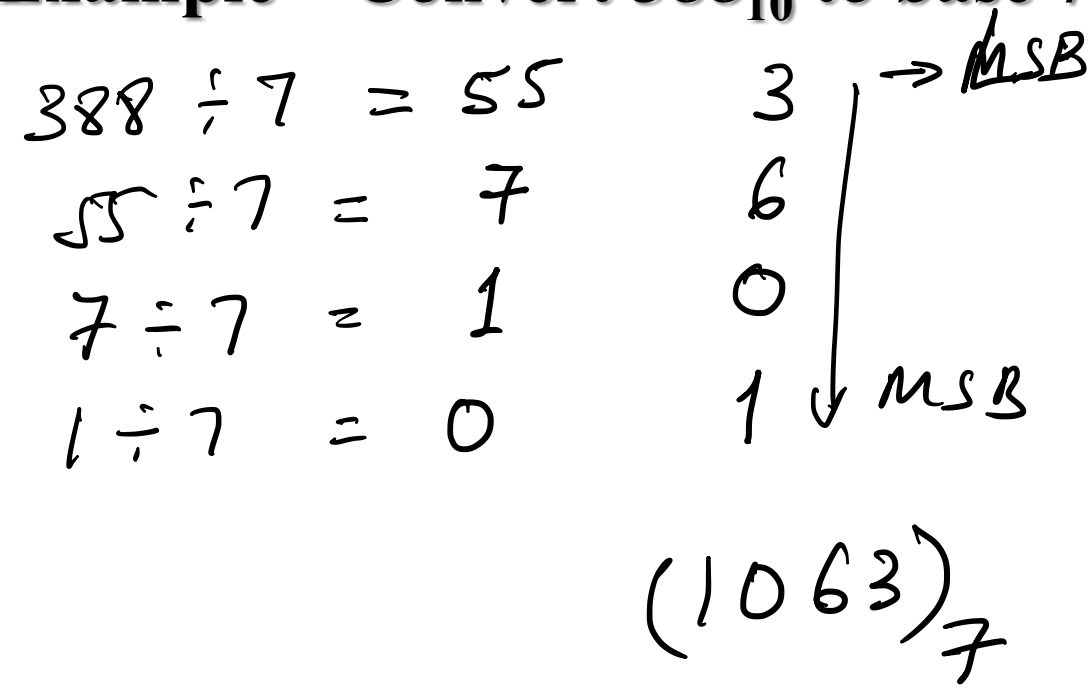
$41 \div 2 =$	20	1 → LSB
$20 \div 2 =$	10	0
$10 \div 2 =$	5	0
$5 \div 2 =$	2	1
$2 \div 2 =$	1	0
$1 \div 2 =$	0	1 → MSB
$0 \div 2 =$	0	0

MSB LSB

$(101001)_2$

Another Conversion Example

- **Can convert decimal to any other base**
- **Example – Convert 388_{10} to base 7:**



$$(A_{n-1}\, A_{n-2} \cdots A_1 A_0)_r$$

$$= r\left[A_{n-1} r^{n-1} + A_{n-2} r^{n-2} + \cdots + A_1\right] + A_0$$

$$r$$

$$(a_{-1} A_{-2} \cdots A_{-m})$$

$$= \left[A_{-1} r^{-1} + A_{-2} r^{-2} + \cdots + A_{-m} r^{-m}\right] r$$

$$= A_{-1} + A_{-2} r^{-1} + \cdots + A_{-m} r^{-m+1}$$

Conversion of Decimal Fractions to Binary

Next Class

- **Repeatedly multiply fractions by 2 to obtain integer product and fraction**
- **Read integer products in order**
- **Example – convert 0.6875_{10} to binary:**

$0.6875 \times 2 = 1.3750$ → 1

$0.375 \times 2 = 0.750$ → 0

$0.75 \times 2 = 1.5$ → 1

$0.5 \times 2 = 1.0$ → 1

$0 \times 2 = 0.0$ → 0

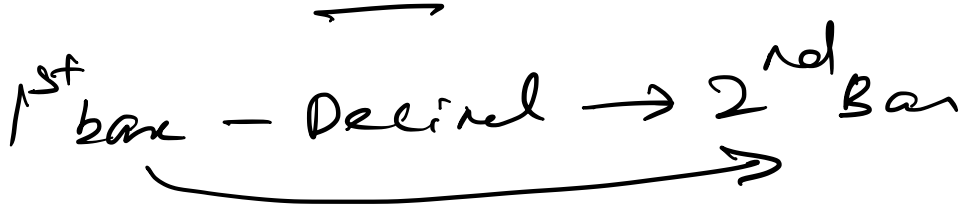
$(.1011)_2$

Additional Issue – Fractional Part

- Note that in this conversion, the fractional part can become 0 as a result of the repeated multiplications
- In general, it may take many bits to get this to happen or it may never happen
- Example: Convert 0.65_{10} to binary
  - $0.65 = 0.1\overline{01001}\ldots$ 1001
  - The fractional part begins repeating every 4 steps, yielding repeating 1001 forever!
- Solution: specify number of bits to right of radix point and round or truncate to this number

Conversion Between Bases

- **In general, to convert between bases, you need to convert to decimal first, and then from decimal to the target base (exceptions follow)**

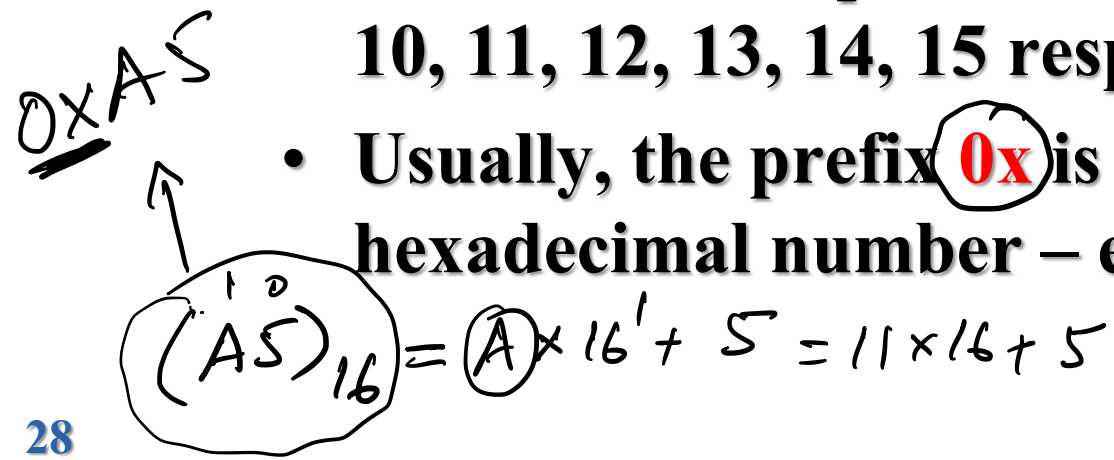


- **So, to convert from one base to another:**
  1) **Convert the integer part**
  2) **Convert the fraction part**
  3) **Join the two results with a radix point**

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 A, B, C, D, E, and F in hexadecimal represent the digits for values 10, 11, 12, 13, 14, 15 respectively**
- **Usually, the prefix 0x is added to indicate a hexadecimal number – e.g. 0xA5**



Numbers in Different Bases

Decimal (Base 10)	Binary (Base 2)	Octal (Base 8)	Hexadecimal (Base 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

$11 = 8 + 3$

$= 8^1 + 3$

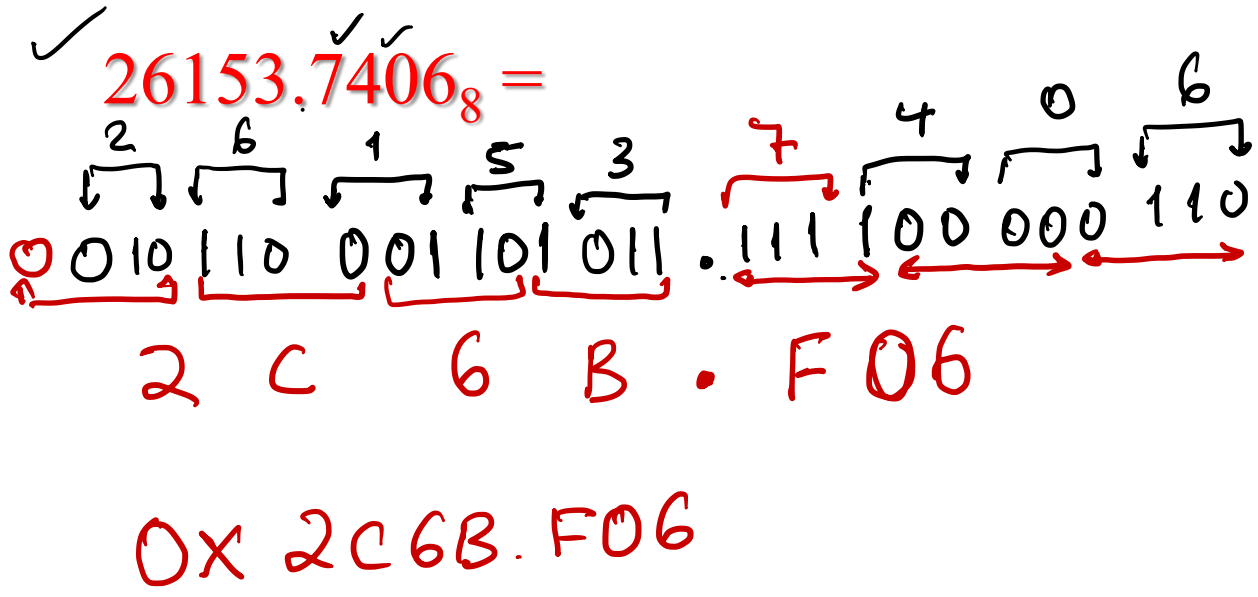
Hexadecimal to Binary and Back

- **Hexadecimal (or Octal) to Binary:**
  - **Restate each hexadecimal (octal) digit as the corresponding four (three) bits starting at the radix point and going both ways**

- **Binary to Hexadecimal (or Octal):**
  - **Group the binary digits into four (three) bit groups starting at the radix point and going both ways, padding with zeros as needed in the fractional part**
  - **Convert each group of four (three) bits to a hexadecimal (octal) digit**

Octal to Hexadecimal via Binary

- **Example: Convert the octal number 26153.7406_8 to binary and therefore to hexadecimal**



Binary Coding

- **Flexibility of representation**
  - **Within the constraints below, can assign any binary combination (called a code word) to any data as long as data is uniquely encoded**

- **Information Types**
  - **Numeric**
  - **Non-numeric** ✓

00
01
10
11

1 bit 0, 1 → 2
2 bits 4
✓3 bit 2^3
4 bit 2^4
n bit 2^n

Non-numeric Binary Codes

- Given n bits, a binary code is a mapping from a set of represented elements to a subset of the 2^n binary numbers

- Example:
  - A binary code for the seven colors of the rainbow
  - Code 100 is not used

Color	Binary Code
Red	000
Orange	001
Yellow	010
Green	011
Blue	101
Indigo	110
Violet	111

Number of Bits Required

- **Given *M* elements to be represented by a binary code, the *n* minimum number of bits needed satisfies the following relationship:**

$$2^{(n-1)} < M \leq 2^n \quad or:$$

$$n \geq \lceil \log_2 M \rceil$$

Where ⌈ · ⌉ is the *ceiling function* – the integer greater than or equal to the argument

- **Example: How many bits are required to represent decimal digits with a binary code?**

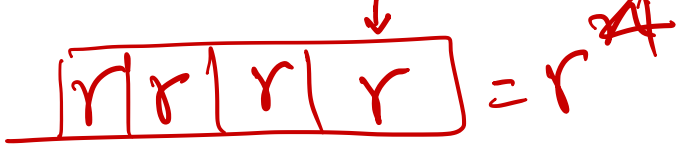
0 1 2 3 ... 9

$n \geq \log_2 10 \approx 3.3$

Number of Elements Represented

- Given n digits in radix r, there are r^n distinct elements that *can* be represented
- But can represent less elements, m, such that:

  $m < r^n$
- Examples:
  - You can represent 4 elements in radix $r = 2$ with $n = 2$ digits:
  - You can represent 4 elements in radix $r = 2$ with $n = 4$ digits:

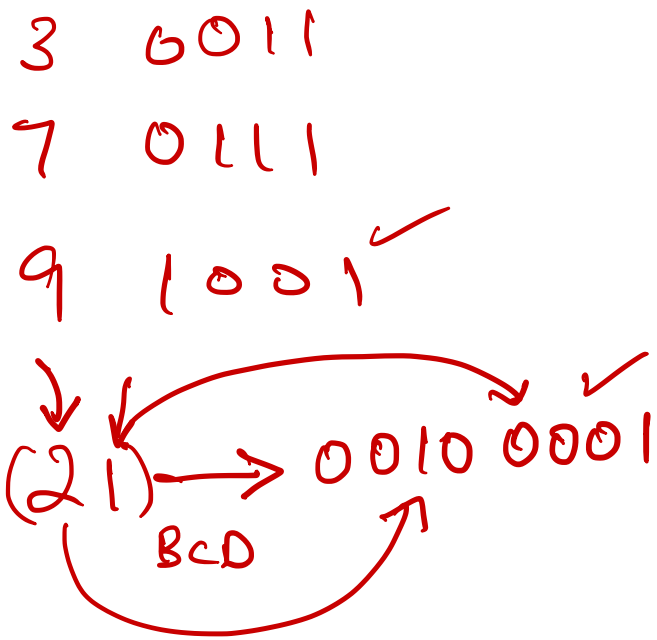


r^n 0, 1, 2 ... r, $r-1$

Decimal Codes – BCD and Gray Code

- **Two useful ways to code decimal digits into binary are *Binary Coded Decimal* (BCD) and *Gray Code:***

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



Binary Coded Decimal (BCD)

- **BCD is the simplest, most intuitive binary code for decimal digits and uses the same powers of 2 as a binary number, but only encodes the first ten values from 0 to 9**
- **Every digit in a decimal number is encoded separately and then combined together**
- **Example (BCD *coding*):**

$185_{10} = (0001\ 1000\ 0101)_{BCD}$

- **Compare with (binary *conversion*):**

$185_{10} = 128 + 32 + 16 + 8 + 1 = 2^7 + 2^5 + 2^4 + 2^3 + 2^0$

$(10111001)_2$

Gray Code

- **Gray code is a binary code where two successive values differ in only one bit change**
- **Example – using the gray code assignments given earlier for decimal digits, only one bit changes on the transition from 3_{10} to 4_{10}:**

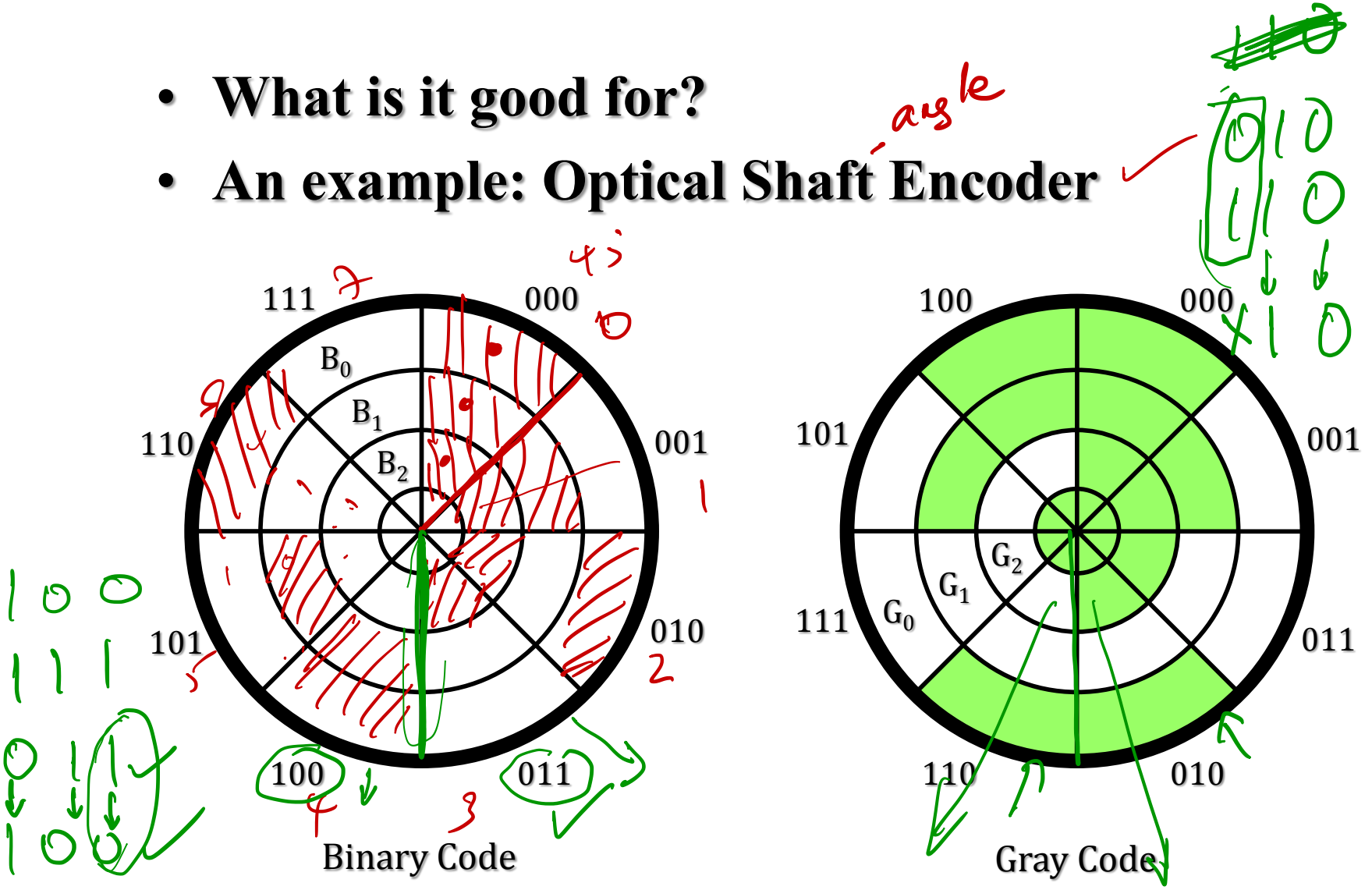
  $0111_{GC} \rightarrow 0110_{GC}$

- **Compare with regular binary code, three bits change on the same transition:**

  $0011_2 \rightarrow 0100_2$

Optical Shaft Encoder using Gray Code

- What is it good for?
- An example: Optical Shaft Encoder



Alphanumeric codes

Next class

- Digital systems need to handle both numeric and non-numeric (characters or symbols) data.
- It is necessary to formulate a binary code for letters of the alphabets, numerals and special characters.
- Alphanumeric character set is a set of elements that include the 10 decimal digits, the 26 letters of the alphabet (lower and upper case), and a number of special characters.

Alphanumeric Codes – ASCII Code

- There are between 64 and 128 elements in this character set.
- The standard binary code for the alphanumeric characters is the American Standard Code for Information Interchange (ASCII), which uses seven bits to code 128 characters.
- The ASCII include 10 numerals, 26 upper case letters, 26 lower case letters, 32 special printable characters such as %,@, and $, and 34 non-printing characters.

Alphanumeric codes – ASCII Code

TABLE 1-5
American Standard Code for Information Interchange (ASCII)

	$B_7B_6B_5$							
$B_4B_3B_2B_1$	000	001	010	011	100	101	110	111
0000	NULL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

Alphanumeric code – ASCII code

Control Characters

NULL	NULL	DLE	Data link escape
SOH	Start of heading	DC1	Device control 1
STX	Start of text	DC2	Device control 2
ETX	End of text	DC3	Device control 3
EOT	End of transmission	DC4	Device control 4
ENQ	Enquiry	NAK	Negative acknowledge
ACK	Acknowledge	SYN	Synchronous idle
BEL	Bell	ETB	End of transmission block
BS	Backspace	CAN	Cancel
HT	Horizontal tab	EM	End of medium
LF	Line feed	SUB	Substitute
VT	Vertical tab	ESC	Escape
FF	Form feed	FS	File separator
CR	Carriage return	GS	Group separator
SO	Shift out	RS	Record separator
SI	Shift in	US	Unit separator
SP	Space	DEL	Delete