Sistemi digitali e informazione

Corso di Architettura degli Elaboratori (modulo Reti Logiche)

Dott. Francesco De Angelis
francesco.deangelis@unicam.it

Scuola di Scienze e Tecnologie - Sezione di Informatica

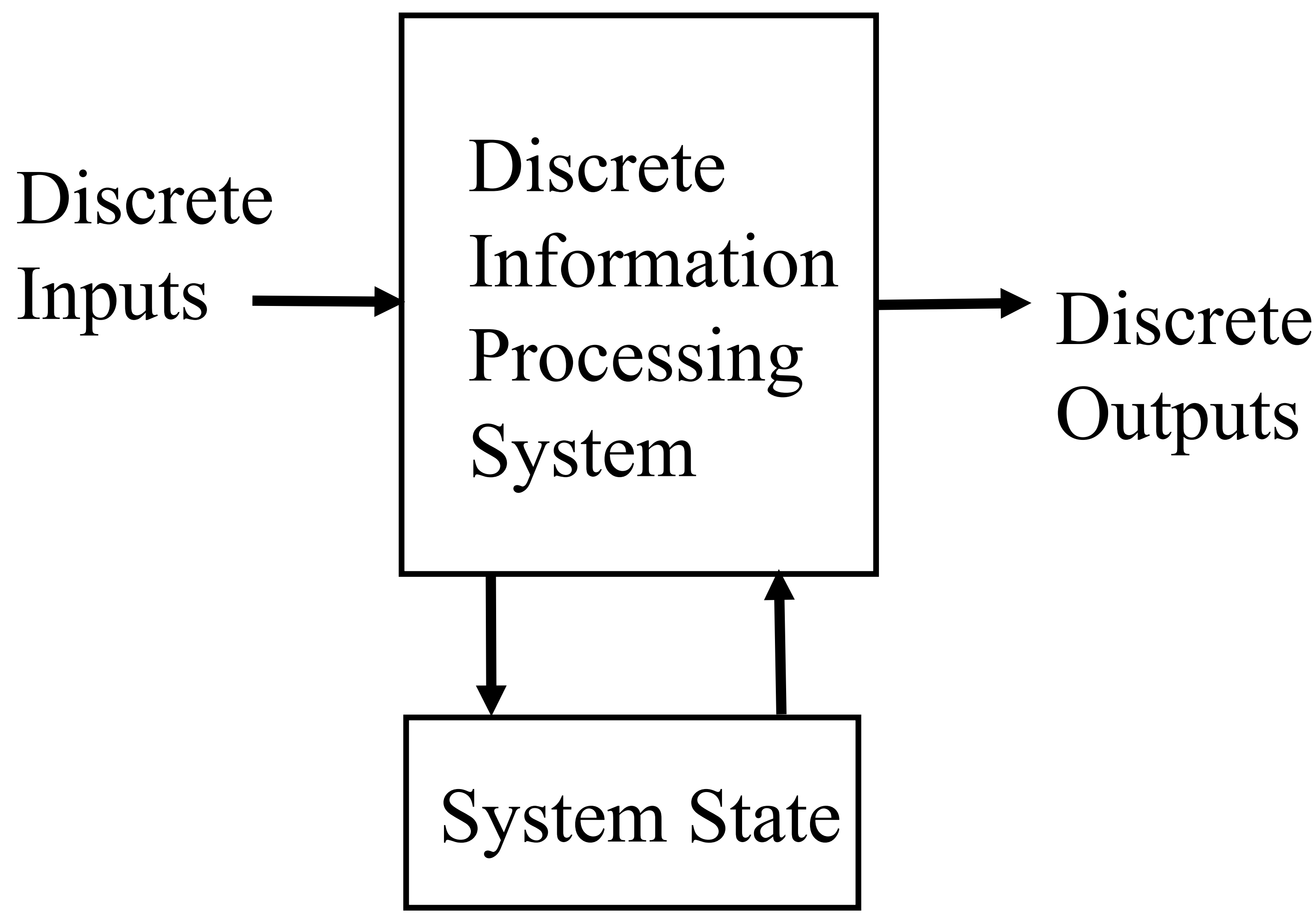
Architettura degli Elaboratori – Reti Logiche

Overview

- Digital Systems, Computers, and Beyond
- Information Representation
- Number Systems [binary, octal and hexadecimal]
- Arithmetic Operations
- Base Conversion
- Decimal Codes [BCD (binary coded decimal)]
- Alphanumeric Codes
- Parity Bit
- Gray Codes

DIGITAL & COMPUTER SYSTEMS
Digital System

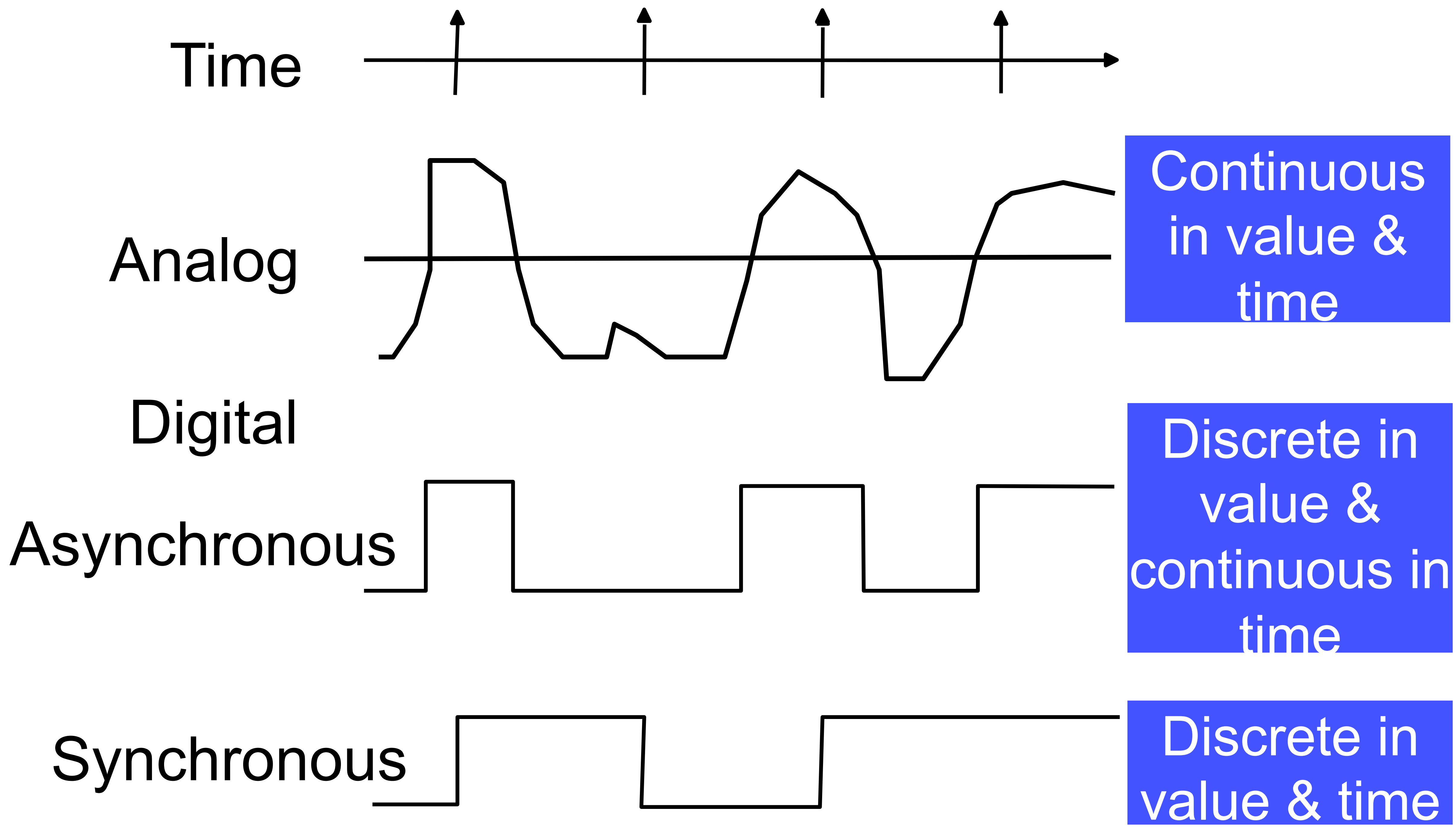
- Takes a set of discrete information inputs and discrete internal information (system state) and generates a set of discrete information outputs.



INFORMATION REPRESENTATION - Signals

- Information variables represented by physical quantities.
- For digital systems, the variables take on discrete values.
- Two level, or binary values are the most prevalent values in digital systems.
- Binary values are represented abstractly by:
  - digits 0 and 1
  - words (symbols) False (F) and True (T)
  - words (symbols) Low (L) and High (H)
  - and words On and Off.
- Binary values are represented by values or ranges of values of physical quantities

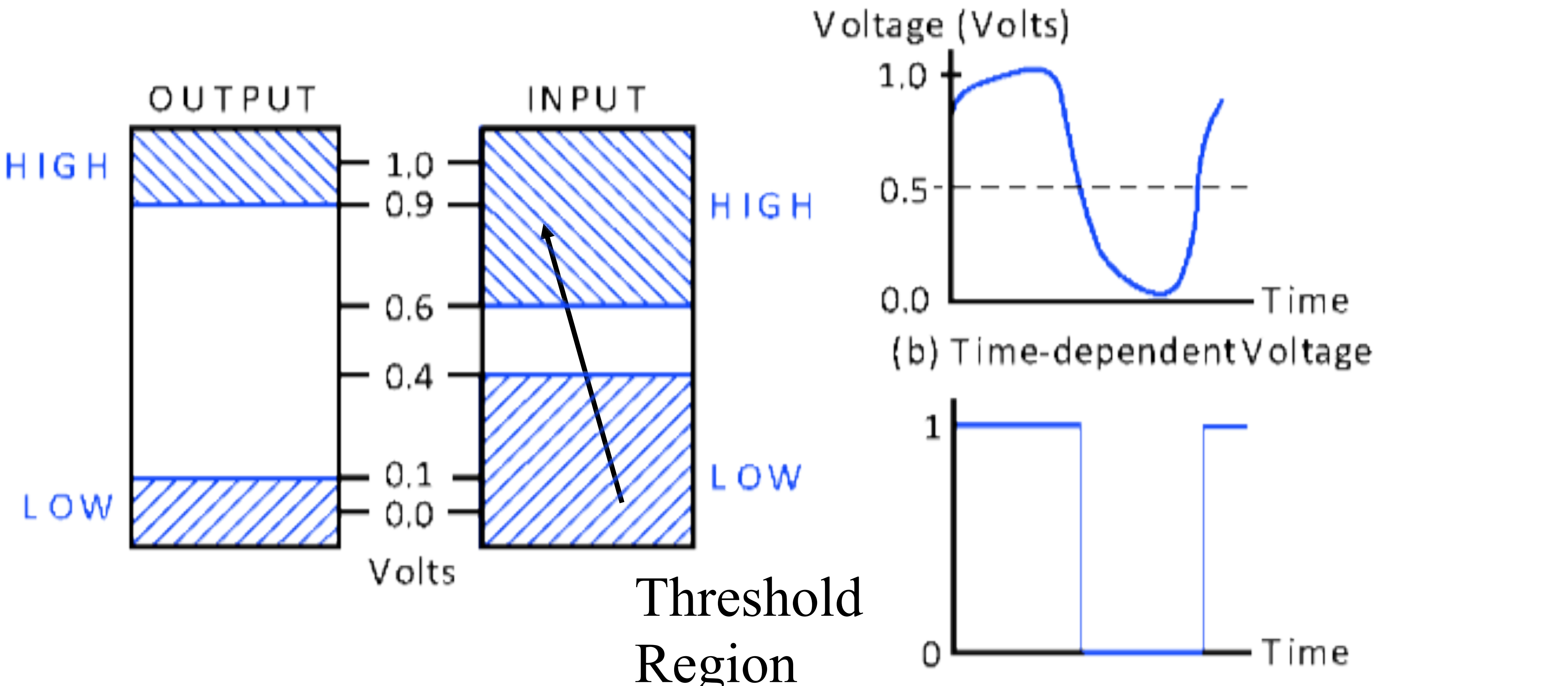
Signal Examples Over Time



Binary Values: Other Physical Quantities

- What are other physical quantities represent 0 and 1?
  - CPU Voltage
  - Disk Magnetic Field Direction
  - CD Surface Pits/Light
  - Dynamic RAM Electrical Charge

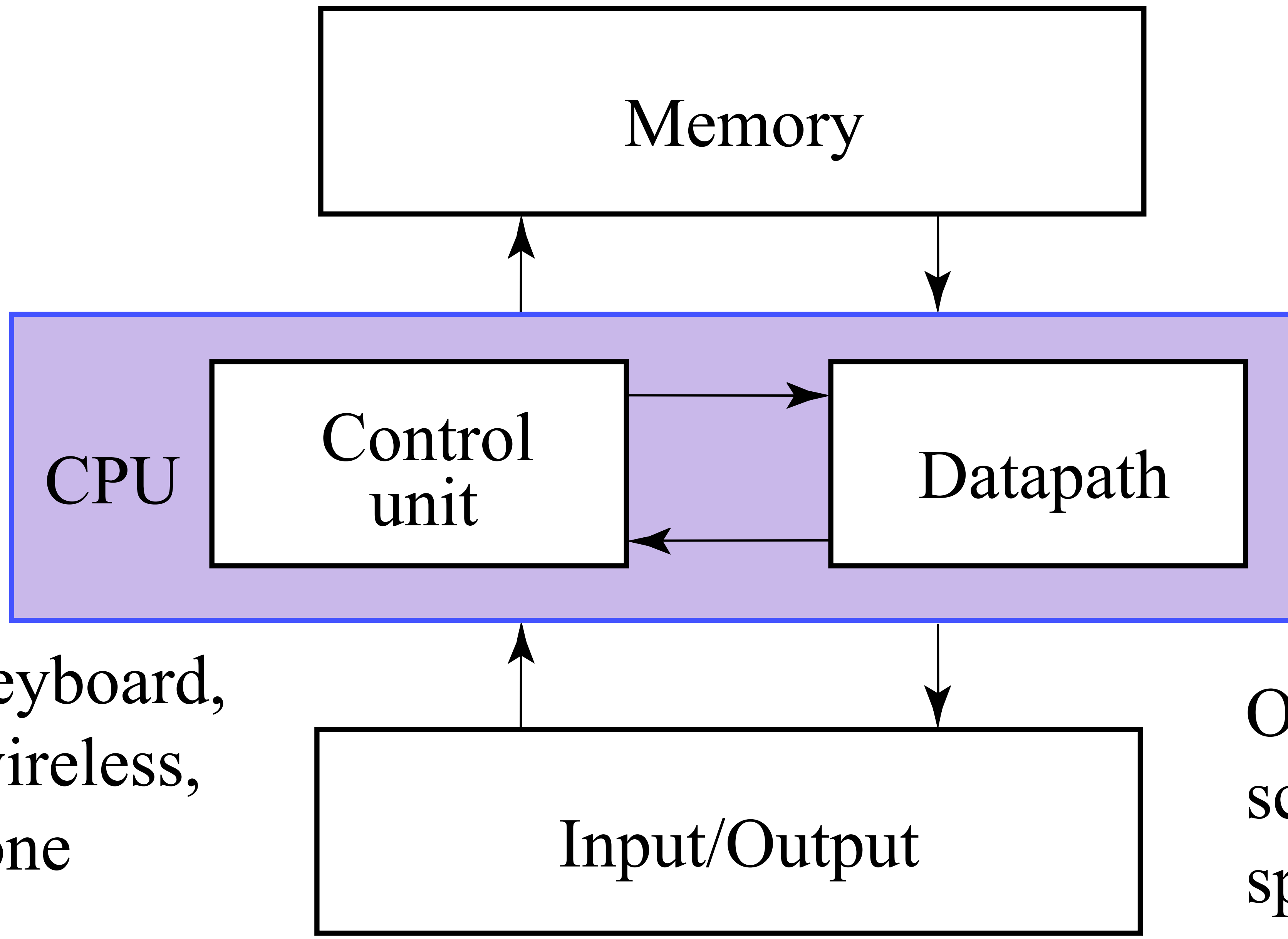
Signal Example – Physical Quantity: Voltage



(a) Example voltage ranges

(c) Binary model of time-dependent voltage

Digital Computer Example



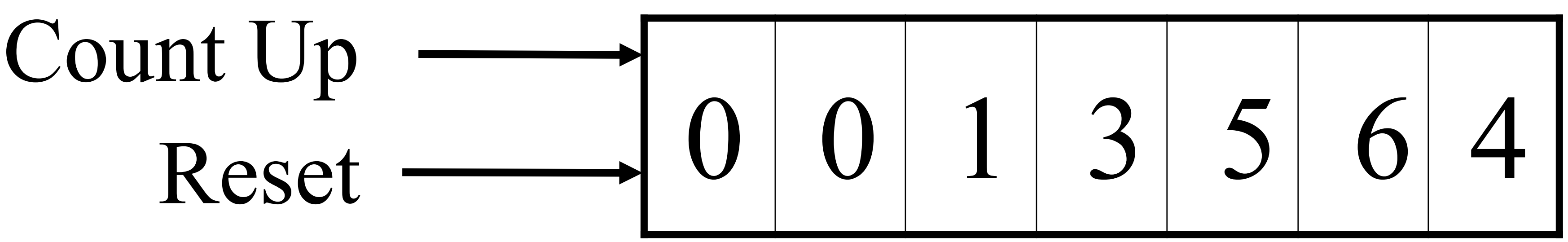
Inputs: keyboard, mouse, wireless, microphone

Outputs: LCD screen, wireless, speakers

Synchronous or Asynchronous?

Digital System Example:

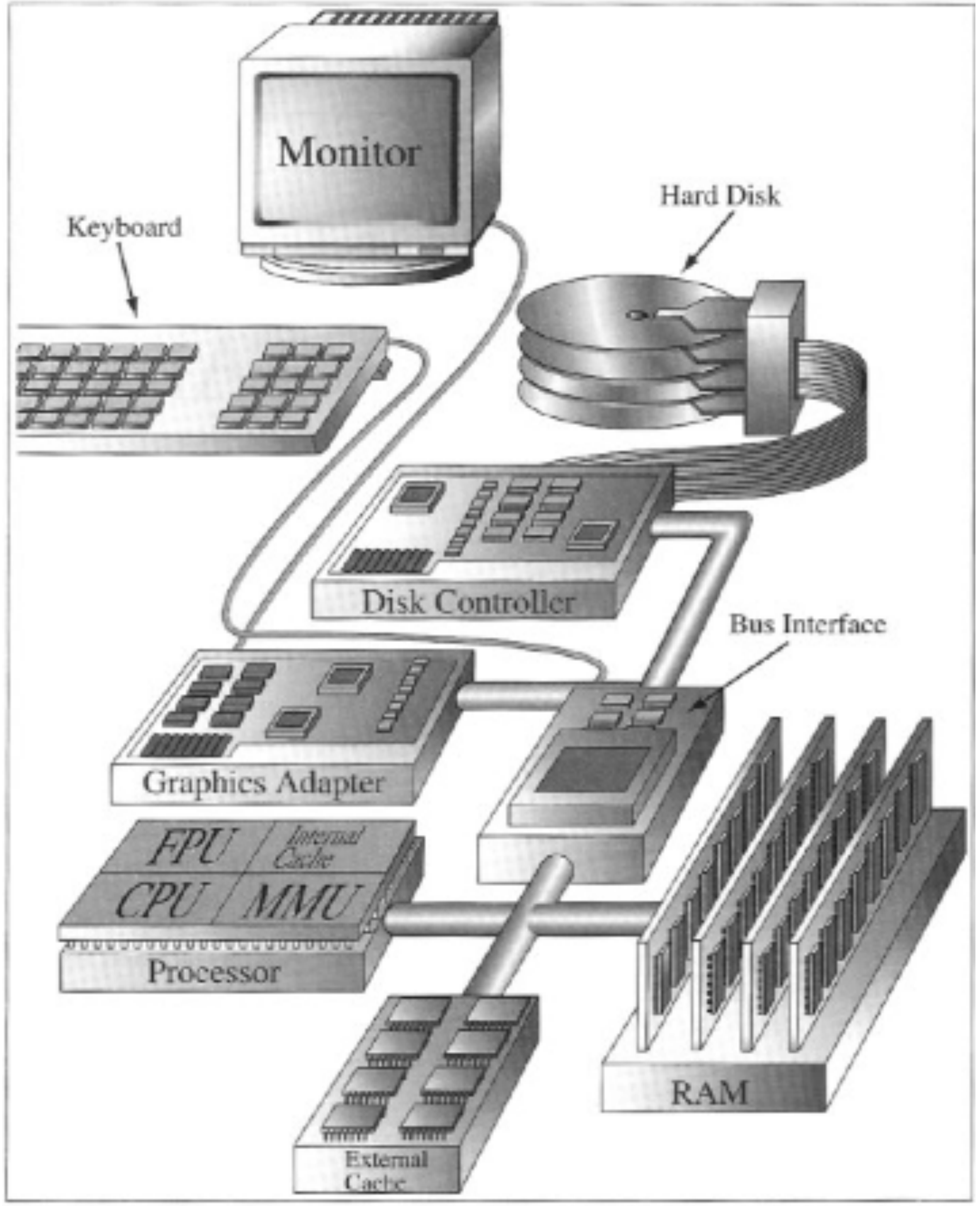
A Digital Counter (e. g., odometer):



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

Synchronous or Asynchronous?

"Computer Architecture"

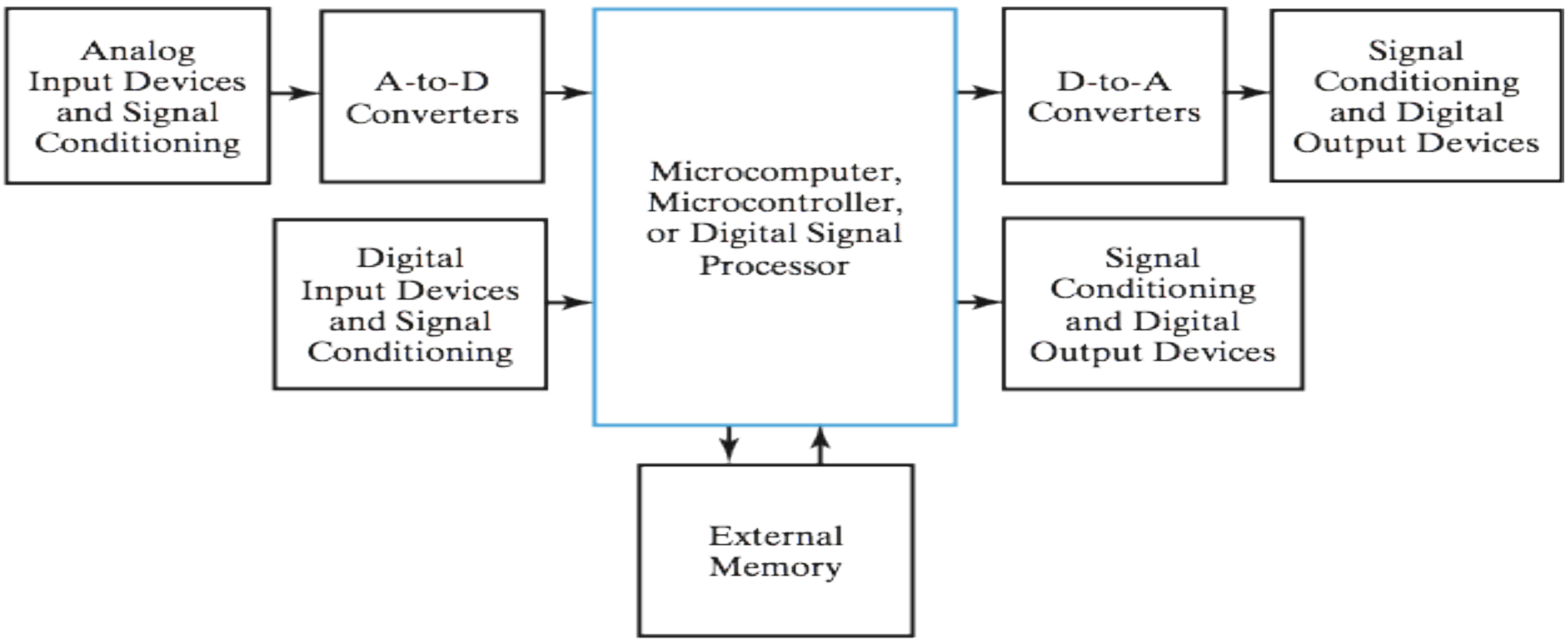


And Beyond – Embedded Systems

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

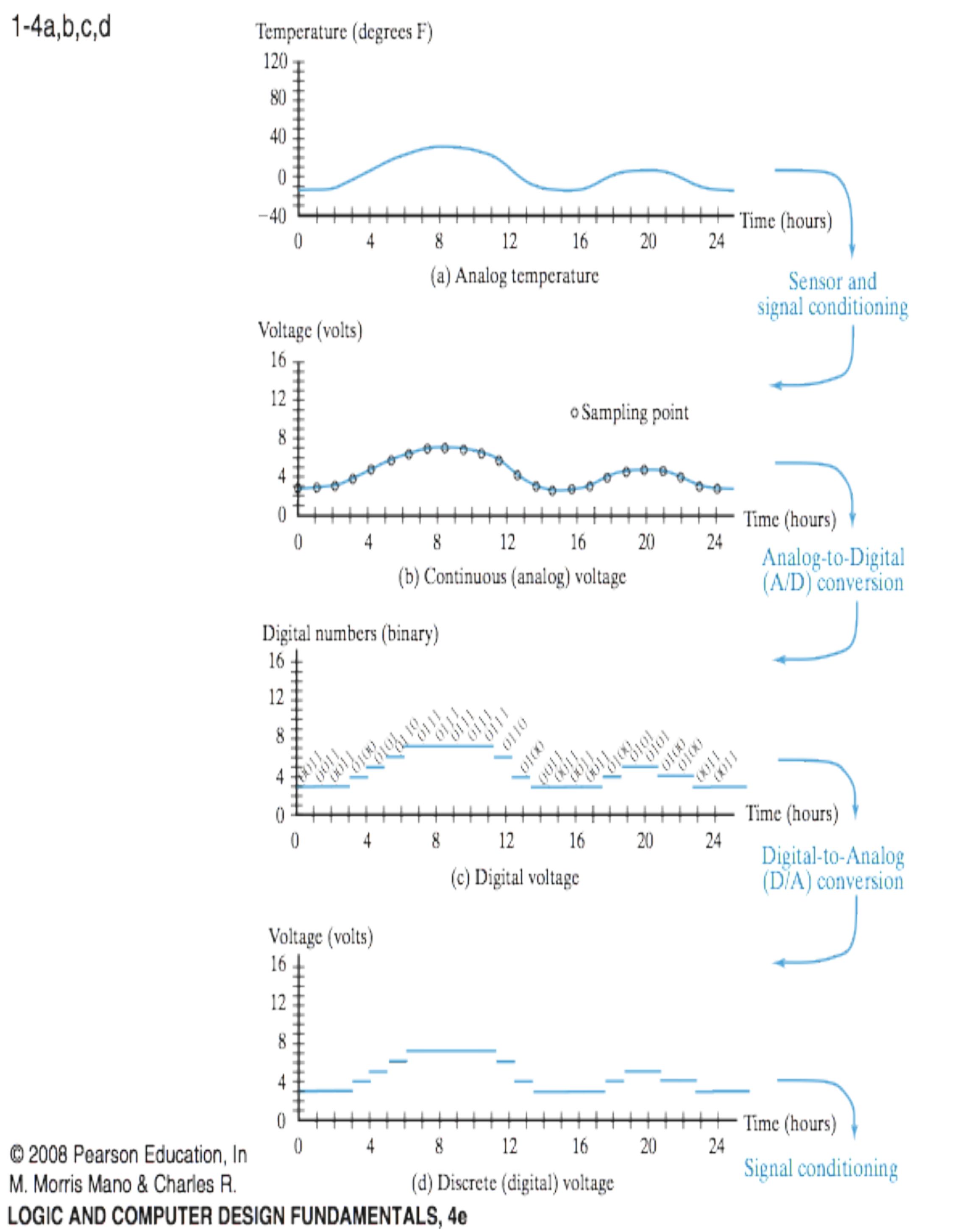
DIGITAL & COMPUTER SYSTEMS - Embedded System

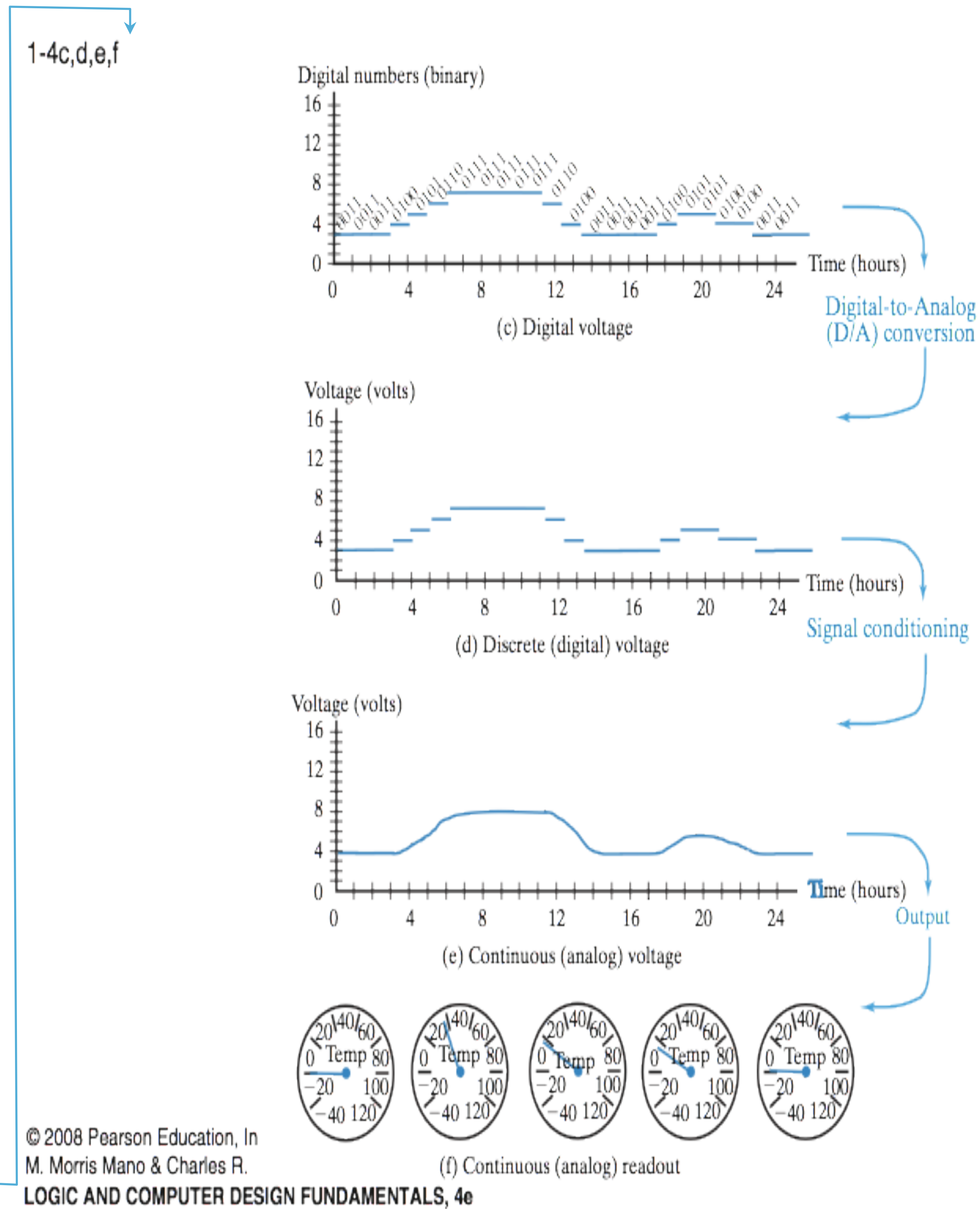
1-3



M. Morris Mano & Charles R. Kime
LOGIC AND COMPUTER DESIGN FUNDAMENTALS, 4e

DIGITAL & COMPUTER SYSTEMS - Embedded System - Example





DIGITAL & COMPUTER SYSTEMS - Embedded System - Applications

T 1-1

TABLE 1-1
Embedded System Examples

Application Area	Product
Banking, commerce and manufacturing	Copiers, FAX machines, UPC scanners, vending machines, automatic teller machines, automated warehouses, industrial robots
Communication	Cell phones, routers, satellites
Games and toys	Video games, handheld games, talking stuffed toys
Home appliances	Digital alarm clocks, conventional and microwave ovens, dishwashers
Media	CD players, DVD players, flat panel TVs, Digital cameras, digital video cameras
Medical equipment	Pacemakers, incubators, magnetic resonance imaging
Personal	Digital watches, MP3 players, personal digital assistants
Transportation and navigation	Electronic engine controls, traffic light controllers, aircraft flight controls, global positioning systems

Types of Digital Systems

- No state present
  - Combinational Logic System
  - Output = Function(Input)
- State present
  - State updated at discrete times
    - => Synchronous Sequential System
  - State updated at any time
    - =>Asynchronous Sequential System
  - State = Function (State, Input)
  - Output = Function (State)
    or Function (State, Input)

NUMBER SYSTEMS – Representation

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

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

in which $0 \leq A_i < r$ 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"

2^{40} (1,099,511,627,776) is Tera, denoted “T"

ARITHMETIC OPERATIONS - Binary Arithmetic

- Single Bit Addition with Carry
- Multiple Bit Addition
- Single Bit Subtraction with Borrow
- Multiple Bit Subtraction
- Multiplication
- BCD Addition

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

Multiple Bit Binary Addition

- Extending this to two multiple bit examples:

Carries	00000	101100
Addend	01100	10110
Addend	+10001	+10111
Sum	11101	101101

- Note: The 0 is the default Carry-In to the least significant bit.

Single Bit Binary Subtraction with Borrow

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

Z	0	0	0	0
X	0	0	1	1
- Y	-0	-1	-0	-1
BS	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
BS	11	1 0	0 0	1 1

Multiple Bit Binary Subtraction

- Extending this to two multiple bit examples:

Borrows	00000	00110
Minuend	10110	10110
Subtrahend	-10010	-10011
Difference	00100	00011

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

Multiple Bit Binary Subtraction

Borrows			00110
Minuend	10011		11110
Subtrahend	-11110		-10011
Difference			-01011

Binary Multiplication

- The binary multiplication table is simple
- 0*0=0 | 1*0=0 | 0*1=0 | 1*1=1
- Extending multiplication to <u>multiple digits</u>
- Multiplicand 1011
- Multiplier * 101
- Partial Products 1011
- 0000-
- 1011--
- Product 110111

BASE CONVERSION - 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 Σ (digit × respective power of 2).
- Example: Convert 11010_2 to N_{10}

- $1 \times 2^4 = 16$
- $+ 1 \times 2^3 = 8$
- $+ 0 \times 2^2 = 0$
- $+ 1 \times 2^1 = 2$
- $+ 0 \times 2^0 = 0$
- 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.

Converting Decimal to Binary

- 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

- 9 8 7 6 5 4 3 2 1 0
- 1 0 0 1 1 1 0 0 0 1

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 (given in decimal)

Numbers in Different Bases

- Good idea to memorize!

Decimal (Base 10)	Binary (Base 2)	Octal (Base 8)	Hexa decimal (Base 16)
00	00000	00	00
01	00001	01	01
02	00010	02	02
03	00011	03	03
04	00100	04	04
05	00101	05	05
06	00110	06	06
07	00111	07	07
08	01000	10	08
09	01001	11	09
10	01010	12	0A
11	01011	13	0B
12	01100	14	0C
13	01101	15	0D
14	01110	16	0E
15	01111	17	0F
16	10000	20	10

Conversion Between Bases

- Method 2
- 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

Conversion Details

- To Convert the Integral Part:
  - Repeatedly divide the number by the new radix and save the remainders. The digits for the new radix are the remainders in reverse order of their computation. If the new radix is > 10, then convert all remainders > 10 to digits A, B, …
- To Convert the Fractional Part:
  - Repeatedly multiply the fraction by the new radix and save the integer digits that result. The digits for the new radix are the integer digits in order of their computation. If the new radix is > 10, then convert all integers > 10 to digits A, B, …

Example: Convert 46.6875_{10} To Base 2

- Convert 46 to Base 2

- Convert 0.6875 to Base 2:

- Join the results together with the radix point:

Example: Convert 46.6875_{10} To Base 2

- Answer 1: Converting 46 as integral part:
- 46/2 = 23 rem = 0
- 23/2 = 11 rem = 1
- 11/2 = 5 remainder = 1
- 5/2 = 2 remainder = 1
- 2/2 = 1 remainder = 0
- 1/2 = 0 remainder = 1
- Reading off in the reverse direction: 101110_2
- Answer 2: Converting 0.6875 as fractional part:
- 0.6875 * 2 = 1.3750 int = 1
- 0.3750 * 2 = 0.7500 int = 0
- 0.7500 * 2 = 1.5000 int = 1
- 0.5000 * 2 = 1.0000 int = 1
- 0.0000
- Reading off in the forward direction: 0.1011_2
- Answer 3: Combining Integral and Fractional Parts: 101110.1011_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 Problem: Convert 0.65_{10} to N_2
  - 0.65 = 0.1010011001001 …
  - 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.

Checking the Conversion

- To convert back, sum the digits times their respective powers of r.
- From the prior conversion of 46.6875_{10}
  - 101110_2 = 1*32+0*16+1*8+1*4+1*2+0*1
  - = 32 + 8 + 4 + 2
  - = 46
  - 0.1011_2 = 1/2 + 1/8 + 1/16
  - = 0.5000 + 0.1250 + 0.0625
  - = 0.6875

Why Do Repeated Division and Multiplication Work?

- Divide the integer portion of the power series by radix r. The remainder of this division is A_0, represented by the term A_0/r.
- Discard the remainder and repeat, obtaining remainders A_1, …
- Multiply the fractional portion of the power series by radix r. The integer part of the product is A_{-1}.
- Discard the integer part and repeat, obtaining integer parts A_{-2}, …
- This demonstrates the algorithm for any radix r >1.

Octal (Hexadecimal) to Binary and Back

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

Octal to Hexadecimal via Binary

- Convert octal to binary.
- Use groups of four bits and convert as above to hexadecimal digits.
- Example: Octal to Binary to Hexadecimal
  - $6\ 3\ 5\ .\ 1\ 7\ 7_8$
- Why do these conversions work?

Octal to Hexadecimal via Binary

- Answer 1:
- 6 3 5 . 1 7 7_8
- 110|011|101 . 001|111|111_2
- Regroup:
- 1|1001|1101 . 0011|1111|1000_2
- Convert:
- 1|1001|1101 . 0011|1111|1000_2
- 1 9 D . 3 F 8_{16}
- Answer 2: Marking off in groups of three (four) bits corresponds to dividing or multiplying by $2^3 = 8$ ($2^4 = 16$) in the binary system.

A Final Conversion Note

- You can use arithmetic in other bases if you are careful:
- Example: Convert 101110_2 to Base 10 using binary arithmetic:
  - Step 1 101110 / 1010 = 100 r 0110
  - Step 2 100 / 1010 = 0 r 0100
  - Converted Digits are 0100_2 | 0110_2
  - or 4 6 $_{10}$

Binary Numbers and Binary Coding

- Flexibility of representation
  - Within constraints below, can assign any binary combination (called a code word) to any data as long as data is uniquely encoded.
- Information Types
  - Numeric
- Must represent range of data needed
- Very desirable to represent data such that simple, straightforward computation for common arithmetic operations permitted
- Tight relation to binary numbers
  - Non-numeric
- Greater flexibility since arithmetic operations not applied.
- Not tied to binary numbers

Non-numeric Binary Codes

- Given n binary digits (called 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 Number
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 minimum number of bits, n, needed, satisfies the following relationships:
  - $2^n \geq M > 2^{(n-1)}$
  - $n = \lceil \log_2 M \rceil$ where $\lceil x \rceil$, called the ceiling function, is the integer greater than or equal to x.

- Example: How many bits are required to represent <u>decimal digits</u> with a binary code?

Number of Bits Required

- Answer:
- M = 10
- Therefore n = 4 since:
- $2^4 \geq 10 > 2^3 = 8$
- and the ceiling function for $\log_2 10$ is 4.

Number of Elements Represented

- Given n digits in radix r, there are r^n distinct elements that can be represented.
- But, you can represent m elements, $m < r^n$
- Examples:
  - You can represent 4 elements in radix r = 2 with n = 2 digits: (00, 01, 10, 11).
  - You can represent 4 elements in radix r = 2 with n = 4 digits: (0001, 0010, 0100, 1000).
  - This second code is called a "one hot" code.

DECIMAL CODES - Binary Codes for Decimal Digits

- There are over 8,000 ways that you can chose 10 elements from the 16 binary numbers of 4 bits. A few are useful:

Decimal	8,4,2,1	Excess3	8,4,-2,-1	Gray
0	0000	0011	0000	0000
1	0001	0100	0111	0100
2	0010	0101	0110	0101
3	0011	0110	0101	0111
4	0100	0111	0100	0110
5	0101	1000	1011	0010
6	0110	1001	1010	0011
7	0111	1010	1001	0001
8	1000	1011	1000	1001
9	1001	1100	1111	1000

Binary Coded Decimal (BCD)

- The BCD code is the 8,4,2,1 code.
- 8, 4, 2, and 1 are weights
- BCD is a weighted code
- This code 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.
- Example: 1001 (9) = 1000 (8) + 0001 (1)
- How many "invalid" code words are there?
- What are the "invalid" code words?

Excess 3 Code and 8, 4, –2, –1 Code

Decimal	Excess 3	8, 4, –2, –1
0	0011	0000
1	0100	0111
2	0101	0110
3	0110	0101
4	0111	0100
5	1000	1011
6	1001	1010
7	1010	1001
8	1011	1000
9	1100	1111

- What interesting property is common to these two codes?

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)

BCD Arithmetic

- Given a BCD code, we use binary arithmetic to add the digits:

```
 8         1000  Eight
+5        +0101  Plus 5
13         1101  is 13 (> 9)
```

  - Note that the result is MORE THAN 9, so must be represented by two digits!
  - To correct the digit, subtract 10 by adding 6 modulo 16.

```
 8           1000 Eight
+5          +0101 Plus 5
13           1101 is 13 (> 9)
            +0110 so add 6
carry = 1    0011 leaving 3 + cy
      0001 | 0011 Final answer (two digits)
```

  - If the digit sum is > 9, add one to the next significant digit

BCD Addition Example

- Add 2905_{BCD} to 1897_{BCD} showing carries and digit corrections.

				0
	0001	1000	1001	0111
+	0010	1001	0000	0101
	____	____	____	____

BCD Addition Example

	1	1	1	0
	0001	1000	1001	0111
+	0010	1001	0000	0101
	0100	10010	1010	1100
+	0000	+0110	+0110	+0110
	0100	11000	10000	10010

ALPHANUMERIC CODES - ASCII Character Codes

- American Standard Code for Information Interchange
- This code is a popular code used to represent information sent as character-based data. It uses 7-bits to represent:
  - 94 Graphic printing characters.
  - 34 Non-printing characters
- Some non-printing characters are used for text format (e.g. BS = Backspace, CR = carriage return)
- Other non-printing characters are used for record marking and flow control (e.g. STX and ETX start and end text areas).

ALPHANUMERIC CODES - ASCII Character Codes

T 1-5

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

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

ASCII Properties

ASCII has some interesting properties:

- Digits 0 to 9 span Hexadecimal values 30_{16} to 39_{16}.
- Upper case A-Z span 41_{16} to $5A_{16}$.
- Lower case a-z span 61_{16} to $7A_{16}$.
  - Lower to upper case translation (and vice versa) occurs by flipping bit 6.
- Delete (DEL) is all bits set, a carryover from when punched paper tape was used to store messages.
- Punching all holes in a row erased a mistake!

PARITY BIT Error-Detection Codes

- Redundancy (e.g. extra information), in the form of extra bits, can be incorporated into binary code words to detect and correct errors.
- A simple form of redundancy is parity, an extra bit appended onto the code word to make the number of 1's odd or even. Parity can detect all single-bit errors and some multiple-bit errors.
- A code word has even parity if the number of 1's in the code word is even.
- A code word has odd parity if the number of 1's in the code word is odd.

4-Bit Parity Code Example

- Fill in the even and odd parity bits:

Even Parity Message - Parity	Odd Parity Message - Parity
000 -	000 -
001 -	001 -
010 -	010 -
011 -	011 -
100 -	100 -
101 -	101 -
110 -	110 -
111 -	111 -

- The codeword "1111" has even parity and the codeword "1110" has odd parity. Both can be used to represent 3-bit data.

GRAY CODE – Decimal

Decimal	8,4,2,1	Gray
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

- What special property does the Gray code have in relation to adjacent decimal digits?

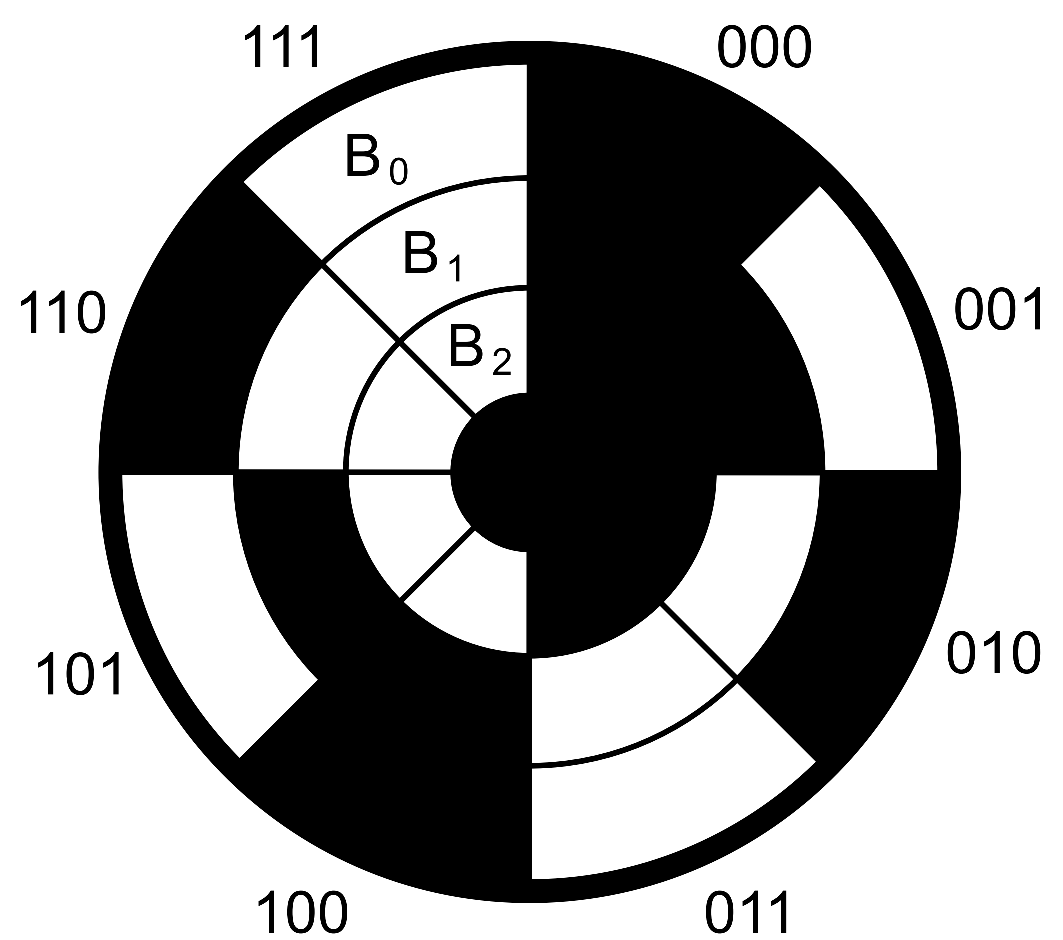
GRAY CODE – Construction

0	0	00	000
1	1	01	001
		11	011
		10	010
			110
			111
			101
			100

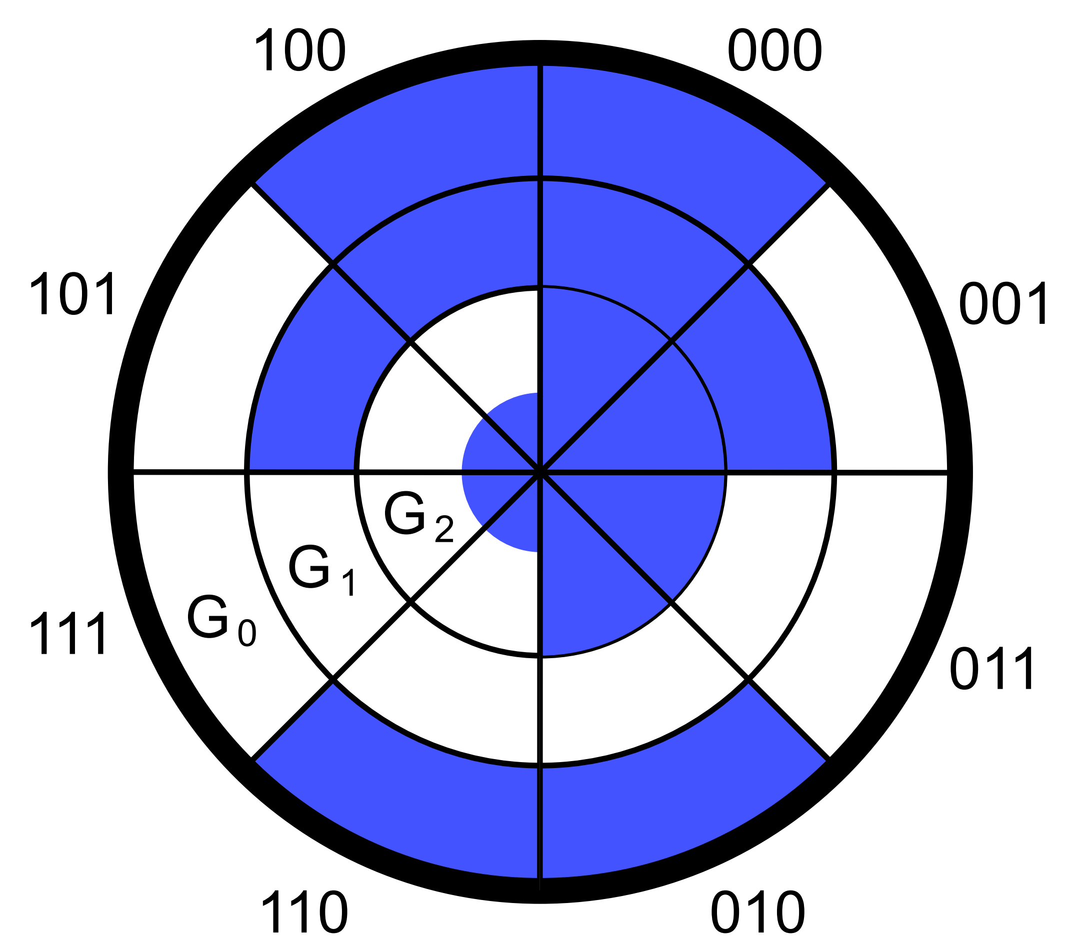
To build a code for n bit reflect the code for n-1 bit and add 0's and 1's

Optical Shaft Encoder

- Does this special Gray code property have any value?
- An Example: Optical Shaft Encoder



(a) Binary Code for Positions 0 through 7



(b) Gray Code for Positions 0 through 7

Shaft Encoder (Continued)

- How does the shaft encoder work?

- For the binary code, what codes may be produced if the shaft position lies between codes for 3 and 4 (011 and 100)?

- Is this a problem?

Shaft Encoder (Continued)

- For the Gray code, what codes may be produced if the shaft position lies between codes for 3 and 4 (010 and 110)?

- Is this a problem?

- Does the Gray code function correctly for these borderline shaft positions for all cases encountered in octal counting?

UNICODE

- UNICODE extends ASCII to 65,536 universal characters codes
  - For encoding characters in world languages
  - Available in many modern applications
  - 2 byte (16-bit) code words

Terms of Use

- All (or portions) of this material © 2008 by Pearson Education,Inc.
- Permission is given to incorporate this material or adaptations thereof into classroom presentations and handouts to instructors in courses adopting the latest edition of Logic and Computer Design Fundamentals as the course textbook.
- These materials or adaptations thereof are not to be sold or otherwise offered for consideration.
- This Terms of Use slide or page is to be included within the original materials or any adaptations thereof.