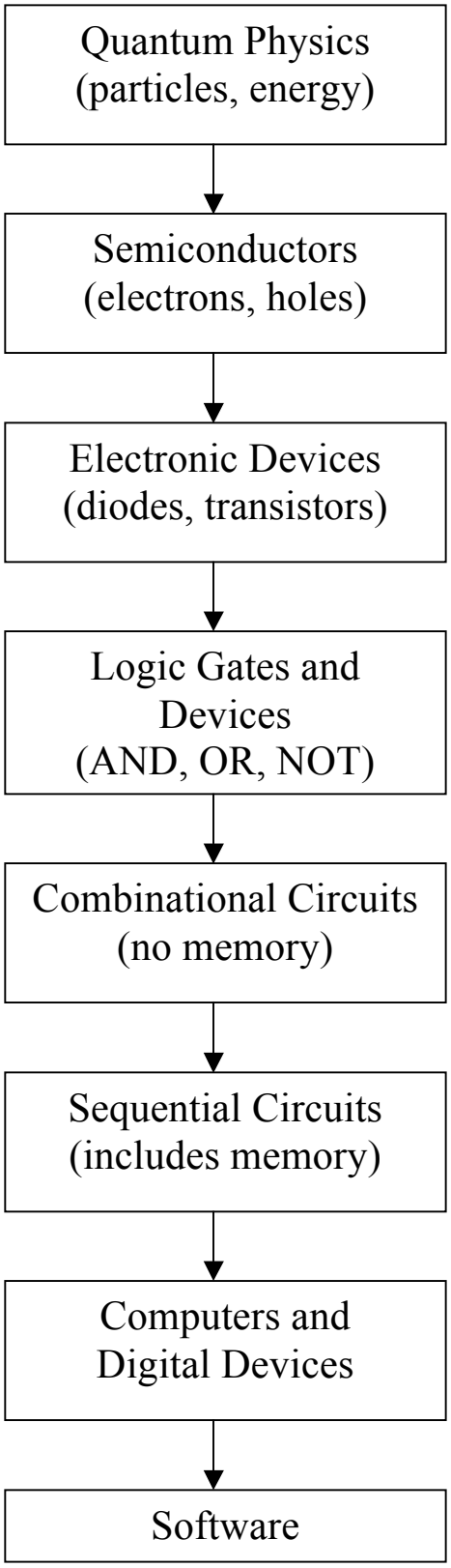
I. INTRODUCTION



Physics

Physics 3000 /
Electronics I (ENGI 4854)

Electronics I (ENGI 4854)

Digital Logic (ENGI 3861)

Digital Logic (ENGI 3861)

Digital Logic (ENGI 3861)

Microprocessors (ENGI 4862)
Digital Systems (ENGI 5865)
Computer Arch. (ENGI 6861)

Intro to Progr. (ENGI 1020)
Advanced Prog. (ENGI 3891)
Data Structures (ENGI 4892)

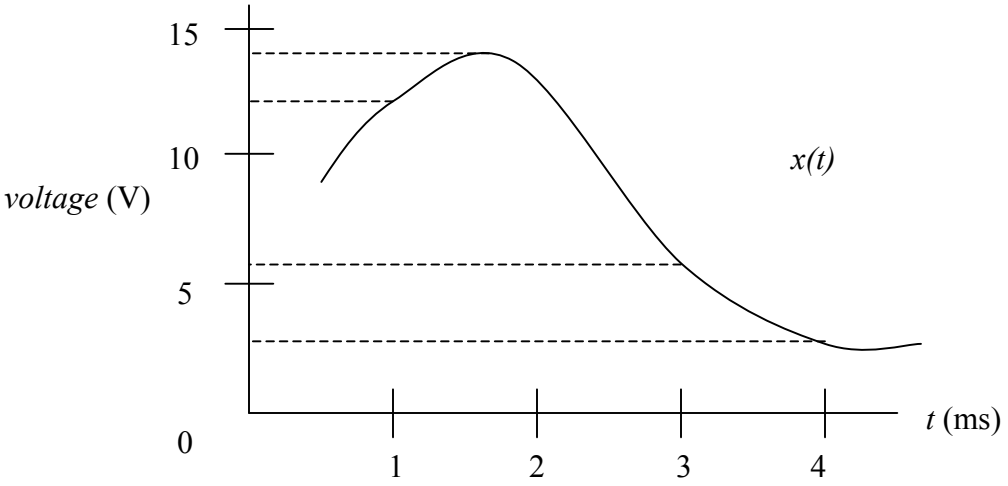
What is a "digital" system? Why use digital methods?

Analog System ≡ data can take on continuous values over continuous points in time

Digital System ≡ values of data represented using a discrete number of levels at discrete points in time
→ typically 2 levels: "binary systems"

Consider

analog signal



⇒ sampled and quantized

time t (ms)	sampled $x(t)$	quantized $\hat{x}(t)$

→ discrete values converted to binary (digital) code
consisting of 4 binary digits or bits

→ now, signal can be represented as codewords

→ data can be transmitted or stored in digital form

e.g., transmission of digital signal

Advantage of Digital Transmission and Storage

→ easy to interpret digital data in presence of noise

Consider

original signal + noise = received signal

Problems of Converting from Analog to Digital:

- quantization error → discrete values lack precision, whereas analog signal has, in theory, infinite precision
- interpolation error → what is signal value between samples?
- generally, ease of recovery of digital data outweighs drawback of quantization and interpolation problems

Other Advantages of Digital Systems

- in many cases, original data is discrete in nature (e.g., binary encoding of English text)
- binary data is easy to manipulate using "digital circuits" → hence, ease of computing using digital computers
- many other motivations: reproducibility of results, speed, circuit size, etc. → See text for more discussion.
- these advantages generally apply to digital storage devices (e.g., computer memory, CDs, DVDs, hard drives, memory sticks, etc.) and communication channels (e.g., cell phones, the Internet, wireless networks, telephone networks, etc.)

Technology Transition

Examples:

	Analog →	Digital
storage:	Vinyl LP	CD, MP3
storage:	VCR/VHS	DVD
communication:	Analog cell phones	Digital cell phones
communication:	Broadcast TV	Digital TV, HDTV
communication:	Voice telephony (POTS)	Internet (VoIP)

History of Digital Circuits

1930s - mechanical switches used for computing machines
→ large, slow, hot

1940s - electronic transistor (i.e., electronic switch) invented
→ small, fast, cool

~1960 - transistors integrated into a "chip" (small ceramic or plastic package with wire leads sticking out)
→ *integrated circuit* or *IC*

Generations of ICs

Timeframe	Technology	Scale (transistors/chip)
early 1960s	Small Scale Integration (SSI)	~ 10s
late 1960s	Medium Scale Integration (MSI)	~ 100s
mid 1970s	Large Scale Integration (LSI)	~ 10,000s
1980s	Very Large Scale Integration (VLSI)	~ 100,000s
1990s	Ultra Large Scale Integration (ULSI)	~ 1,000,000s
Today	ICs exist with billions of transistors/chip	

Logic Gates

- manipulation of digital signals (i.e., digital variables that change over time) critical to modern digital computing and communication systems such as a desktop computer and the Internet.
  → processing in digital systems is based on *logic gates*

- binary values can be labelled "0" and "1", "TRUE (T)" and "FALSE (F)", or "LOW (L)" and "HIGH (H)" and can be stored / represented in many different ways

  → in electronics, such as commonly used Complementary Metal Oxide Semiconductor (CMOS) logic, voltage level represents binary value with typical levels being:

  "0" → 0–1.5 V, "1" → 3.5–5.0 V

  Note: electronic circuits have low voltages (~5 V) and currents (mA) versus motors and generators (120 V, A)

- a *truth table* is a convenient way to represent behaviour of a digital logic circuit

Basic Gate Types

AND

OR

NOT (or INVERTER)

Example: A light in a stairwell is to be controlled by a switch at the top of the stairs and a switch at the bottom of the stairs. Determine the truth table for the logical function F representing the control of the light. Assume when both the upstairs and downstairs switches are down, the light is off. Thereafter, toggling one of the switches will change the state of the light.

How can we use AND, OR, and NOT to implement logic?

- gates can involve more than 2 inputs

AND

XYZ	F
000	
001	
010	
011	
100	
101	
110	
111	

OR

XYZ	F
000	
001	
010	
011	
100	
101	
110	
111	

Example: What is the truth table of the following logic circuit?

XYZ	F
000	
001	
010	
011	
100	
101	
110	
111	

At home, draw truth tables for 4-input AND and OR gates.

More Standard Gates

- we have already seen truth tables for 2-input AND, OR, and NOT gates, but other gates are also of interest:

NAND

NOR

XOR

XNOR (sometimes called "Equivalence")

- some 3-input gates

NAND

XYZ	F
000	
001	
010	
011	
100	
101	
110	
111	

NOR

XYZ	F
000	
001	
010	
011	
100	
101	
110	
111	

XOR

XYZ	F
000	
001	
010	
011	
100	
101	
110	
111	

XNOR

XYZ	F
000	
001	
010	
011	
100	
101	
110	
111	

Can you determine truth tables for 4-input
NAND, NOR, XOR, XNOR ?

Binary Numbers

- one of the most fundamental applications of digital systems is arithmetic and numerical computations
  → how do we represent numbers for digital systems?

- we are used to numbers using base- or radix-10:

  Consider number D where

  e.g., 372.97_{10} =

  so digit d_i contributes $d_i \times 10^i$ to the value of D

  $\Rightarrow$ D =

- other radix can be used such as base-2:

  Consider number B where

  $\Rightarrow$ B =

  e.g., 110101_2 =

  =

  =

e.g., 110.011_2 =

=

=

Decimal to binary?

e.g., 41_{10} = ? in binary

e.g., 0.6875_{10} = ? in binary

e.g., Try $137.875_{10} = ?$ in binary

- an n-bit (unsigned) binary number can take on 2^n possible values

  e.g., if x is a positive integer and is represented by n-bit binary vector, then $0 \leq x \leq 2^n - 1$ corresponding to binary representations $000\ldots00_2$ to $111\ldots11_2$

- other radices can be used to conveniently represent binary data

  e.g., octal ≡ base 8, digits ∈ {0, 1, 2, 3, 4, 5, 6, 7}

  $110010_2 \equiv$

  $73_8 \equiv$

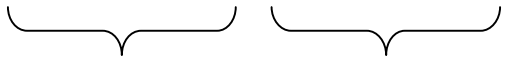
  e.g., hexadecimal ≡ base 16, digits ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}

  $10111001_2 \equiv$

  $C7_{16} \equiv$

- since, in computers, data tends to be organized in groups of 8 bits, where 8 bits is called a *byte*, hexadecimal is a very convenient representation

e.g., $1CE8_{16} \equiv$



byte 1 byte 2

Memorize binary to hex conversion! You will use them again and again!

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F