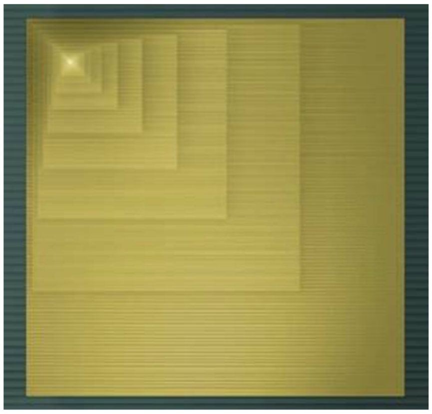


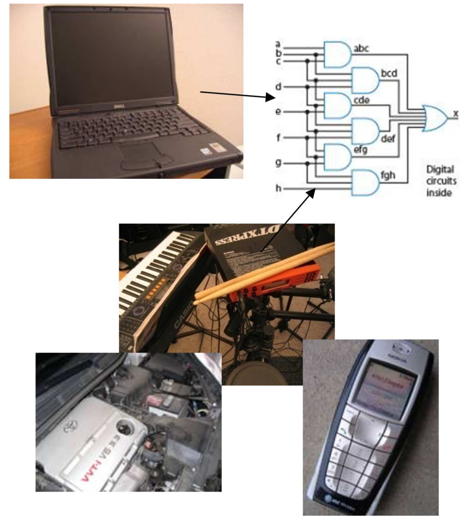
Digital Design

Chapter 1: Introduction

Slides to accompany the textbook *Digital Design*, First Edition,
by Frank Vahid, John Wiley and Sons Publishers, 2007.
http://www.ddvahid.com

1.1

Why Study Digital Design?



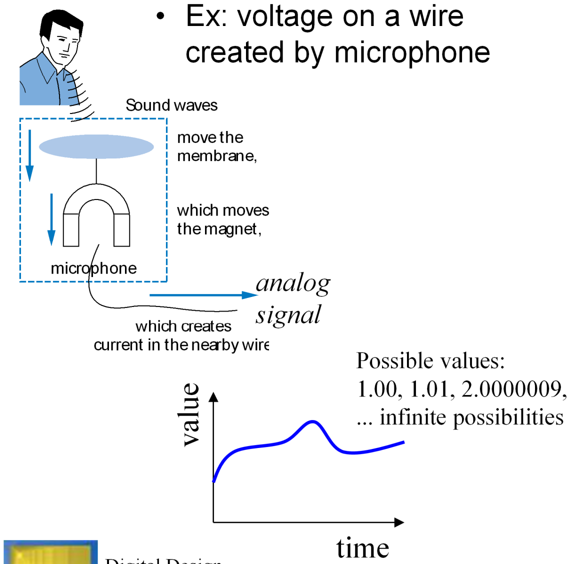
- Look "under the hood" of computers
  - Solid understanding --> confidence, insight, even better programmer when aware of hardware resource issues
- Electronic devices becoming digital
  - Enabled by shrinking and more capable chips
  - Enables:
    - Better devices: Better sound recorders, cameras, cars, cell phones, medical devices,...
    - New devices: Video games, PDAs, ...
  - Known as "embedded systems"
    - Thousands of new devices every year
    - Designers needed: Potential career direction

2

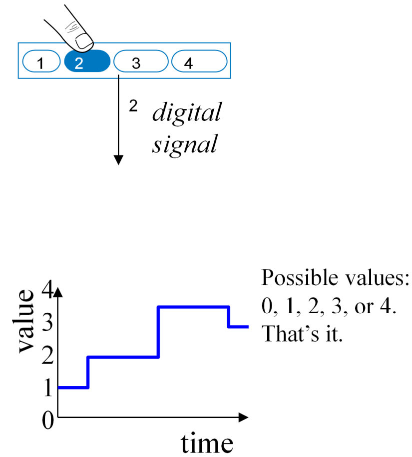
Note: Slides with animation are denoted with a small red "a" near the animated items

What Does "Digital" Mean?

- Analog signal
  - Inifinite possible values
    - Ex: voltage on a wire created by microphone

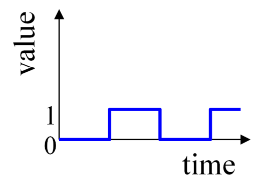


- Digital signal
  - Finite possible values
    - Ex: button pressed on a keypad



Digital Signals with Only Two Values: Binary

- ***Binary*** digital signal -- only *two* possible values
  - Typically represented as **0** and **1**
  - One *b*inary dig*it* is a ***bit***
  - We'll only consider *binary* digital signals
  - Binary is popular because
    - Transistors, the basic digital electric component, operate using *two* voltages (more in Chpt. 2)
    - Storing/transmitting one of *two* values is easier than three or more (e.g., loud beep or quiet beep, reflection or no reflection)

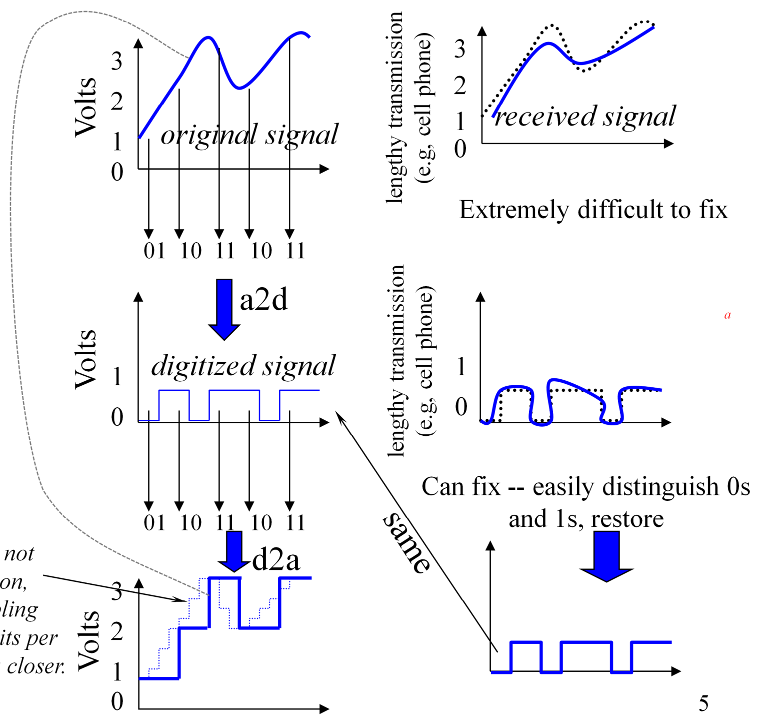


Example of Digitization Benefit

- Analog signal (e.g., audio) may lose quality
  - Voltage levels not saved/copied/transmitted perfectly
- Digitized version enables near-perfect save/cpy/trn.
  - "Sample" voltage at particular rate, save sample using bit encoding
  - Voltage levels still not kept perfectly
  - But we can distinguish 0s from 1s

Let bit encoding be:
1 V: "01"
2 V: "10"
3 V: "11"

Digitized signal not perfect re-creation, but higher sampling rate and more bits per encoding brings closer.



Digitized Audio: Compression Benefit

- Digitized audio can be compressed
  - e.g., MP3s
  - A CD can hold about 20 songs uncompressed, but about 200 compressed
- Compression also done on digitized pictures (jpeg), movies (mpeg), and more
- Digitization has many other benefits too

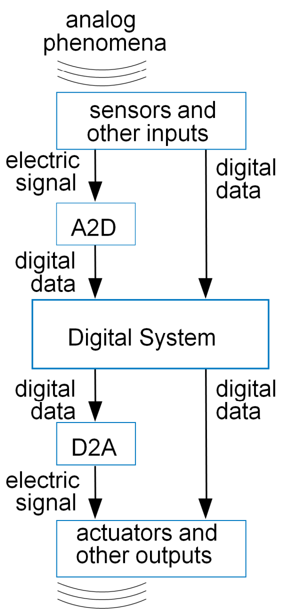
Example compression scheme:

00 --> 0000000000
01 --> 1111111111
1X --> X

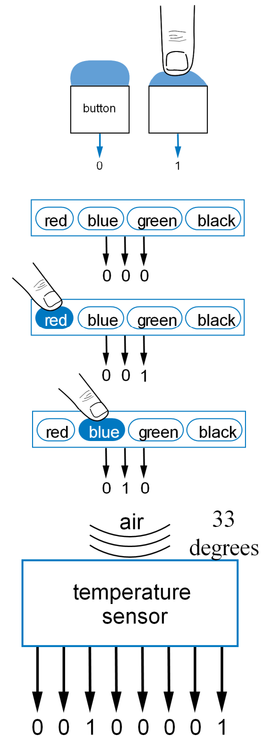
0000000000 0000000000 0000001111 1111111111

00 00 10000001111 01

How Do We Encode Data as Binary for Our Digital System?



- Some inputs inherently binary
  - Button: not pressed (0), pressed (1)
- Some inputs inherently digital
  - Just need encoding in binary
  - e.g., multi-button input: encode red=001, blue=010, …
- Some inputs analog
  - Need analog-to-digital conversion
  - As done in earlier slide -- sample and encode with bits



How to Encode Text: ASCII, Unicode

- ASCII: 7- (or 8-) bit encoding of each letter, number, or symbol
- Unicode: Increasingly popular 16-bit bit encoding
  - Encodes characters from various world languages

Symbol	Encoding	Symbol	Encoding
R	1010010	r	1110010
S	1010011	s	1110011
T	1010100	t	1110100
L	1001100	l	1101100
N	1001110	n	1101110
E	1000101	e	1100101
0	0110000	9	0111001
.	0101110	!	0100001
<tab>	0001001	<space>	0100000

Question:
What does this ASCII bit sequence represent?
1010010 1000101 1010011 1010100

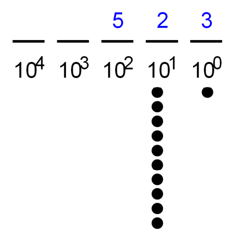
R E S T

- http://www.asciitable.com

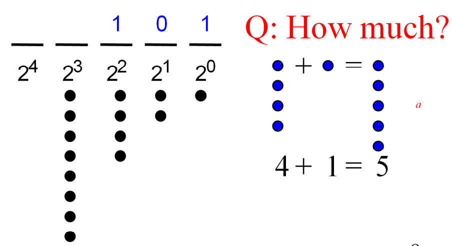
Note: small red "a" (a) in a slide indicates animation

How to Encode Numbers: Binary Numbers

- Each position represents a quantity; symbol in position means how many of that quantity
  - Base ten (***decimal***)
    - Ten symbols: 0, 1, 2, ..., 8, and 9
    - More than 9 -- next position
      - So each position power of 10
    - Nothing special about base 10 -- used because we have 10 fingers



  - Base two (***binary***)
    - Two symbols: 0 and 1
    - More than 1 -- next position
      - So each position power of 2



How to Encode Numbers: Binary Numbers

- Working with binary numbers
  - In base ten, helps to know powers of 10
    - one, ten, hundred, thousand, ten thousand, ...
  - In base two, helps to know powers of 2
    - one, two, four, eight, sixteen, thirty two, sixty four, one hundred twenty eight
      - (Note: unlike base ten, we don't have common names, like "thousand," for each position in base ten -- so we use the base ten name)
    - Q: count up by powers of two

2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
512	256	128	64	32	16	8	4	2	1

512 256 128 64 32 16 8 4 2 1

Converting from Decimal to Binary Numbers: Subtraction Method (Easy for Humans)

- Get the binary weights to add up to the decimal quantity
  - Work from left to right
  - (Right to left – may fill in 1s that shouldn't have been there – try it).
- To make the job easier (especially for big numbers), we can just subtract a selected binary weight from the (remaining) quantity
  - Then, we have a new remaining quantity, and we start again (from the present binary position)
  - Stop when remaining quantity is 0

Desired decimal number: **17**

32	16	8	4	2	1	
_	_	_	_	_	_	
0	_	_	_	_	_	= 32 too much
0	1	_	_	_	_	= 16 (17-16= 1) ok, keep going
0	1	0	0	0	_	= 8, 4, 2 too much
0	0	1	0	0	1	=1-1=0 DONE
0	**1**	**0**	**0**	**0**	**1**	answer

Converting from Decimal to Binary Numbers: Subtraction Method Example

- Q: Convert the number "29" from decimal to binary

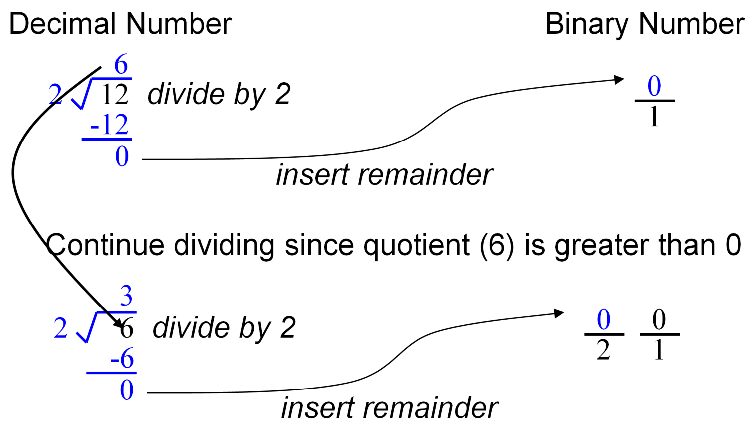
A: Remaining quantity	Binary Number (32, 16, 8, 4, 2, 1)
29	0 0 0 0 0 0
29 − 16 = 13	0 1 0 0 0 0
13 − 8 = 5	0 1 1 0 0 0
5 − 4 = 1	0 1 1 1 0 0
1 − 1 = 0	0 1 1 1 0 1

8 is more than 7, can't use

Done! 29 in decimal is 10111 in binary.

Converting from Decimal to Binary Numbers: Division Method (Good for Computers)

- Divide decimal number by 2 and insert remainder into new binary number.
  - Continue dividing quotient by 2 until the quotient is 0.
- Example: Convert decimal number 12 to binary



Continue dividing since quotient (3) is greater than 0

Converting from Decimal to Binary Numbers: Division Method (Good for Computers)

- Example: Convert decimal number 12 to binary (continued)

Decimal Number — Binary Number

$2\,\overline{)\,3}$ = 1, -2, remainder 1 — *divide by 2*

insert remainder → $\frac{1}{4}\ \frac{0}{2}\ \frac{0}{1}$

Continue dividing since quotient (1) is greater than 0

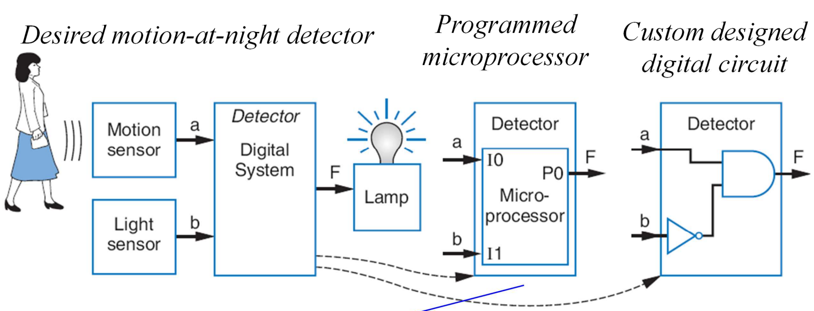
$2\,\overline{)\,1}$ = 0, -0, remainder 1 — *divide by 2*

insert remainder → $\frac{1}{8}\ \frac{1}{4}\ \frac{0}{2}\ \frac{0}{1}$

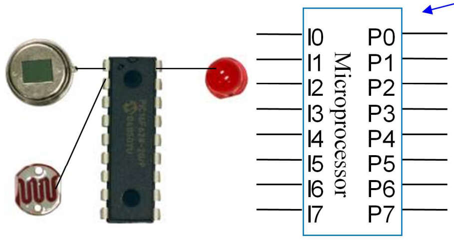
Since quotient is 0, we can conclude that 12 is 1100 in binary

Implementing Digital Systems:

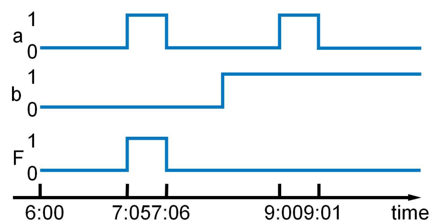
Programming Microprocessors Vs. Designing Digital Circuits



- Microprocessors a common choice to implement a digital system
  - Easy to program
  - Cheap (as low as $1)
  - Available now




```
void main()
{
  while (1) {
    P0 = I0 && !I1;
    // F = a and !b,
  }
}
```

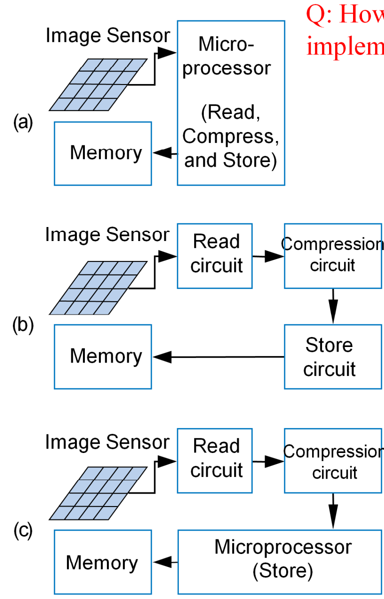


Digital Design: When Microprocessors Aren't Good Enough

- With microprocessors so easy, cheap, and available, why design a digital circuit?
  - Microprocessor may be too slow
  - Or too big, power hungry, or costly

Sample digital camera task execution times (in seconds) on a microprocessor versus a digital circuit:

Task	Microprocessor	Custom Digital Circuit
Read	5	0.1
Compress	8	0.5
Store	1	0.8



Q: How long for each implementation option?

5+8+11 =24 sec

.1+.5+.8 =1.4 sec

.1+.5+1 =1.6 sec

Good compromise

Chapter Summary

- Digital systems surround us
  - Inside computers
  - Inside huge variety of other electronic devices (embedded systems)
- Digital systems use 0s and 1s
  - Encoding analog signals to digital can provide many benefits
    - e.g., audio -- higher-quality storage/transmission, compression, etc.
  - Encoding integers as 0s and 1s: Binary numbers
- Microprocessors (themselves digital) can implement many digital systems easily and inexpensively
  - But often not good enough -- need custom digital circuits