CpE358/CS381

Switching Theory and Logical Design

Summer-1 2004

Class Schedule

Monday	Tuesday	Wednesday	Thursday	Friday
May 17	18	**19 - Class 1**	**20 - Class 2**	21
24 - Class 3	25	**26 - Class 4** **Quiz 1**	27 Commencement	28
31 Memorial Day	June 1	**2 - Class 5** **Project defined**	**3 - Class 6**	4
7 - Class 7	8	**9 - Class 8** **Quiz 2**	**10 - Class 9**	11
14 - Class 10	15	**16 - Class 11** **Quiz 5**	**17 - Class 12**	18
21 - Class 13	22	**23 - Class 14** **Quiz 4**	**24 - Class 15** **Projects due**	25
28 - Class 16	29	**30 - Class 17** **Quiz 5**	July 1	7/2

Course Introduction

- Logistics:
  - Instructor: Bruce McNair
    - Office: Burchard 206
    - Phone: 201-216-5549
    - email: bmcnair@stevens-tech.edu
    - Office hours: Class days ~9:30 - ~11
    - Web site: http://koala.ece.stevens-tech.edu/~bmcnair (course notes, solutions, etc. are here)
  - Homework
    - Must be typed or printed hardcopy, or electronic (i.e., *not* handwritten).
    - email is OK with MS/Office (2000 or previous), or program (e.g., .c, .m, ...) attachments. *Don't email me an executable or a macrovirus*. pdf is OK.
    - ***VERIFY THAT PROGRAMMING SUBMISSIONS INCLUDE ENOUGH ENVIRONMENT TO BE BUILT AND RUN*** (e.g., .h files, initialization, etc.)
    - Include the problem statement with solution. Keep a copy of your hardcopy or electronic homework (it may not be returned)
    - **If you submit homework as an email attachment, make sure your name appears in the file. The file name must include your name (or login), course number, and assignment number, e.g.: bmcnair-CPE358-HW2.doc**
    - **To ensure proper credit for the homework, indicate the *due date* on the homework**
    - Homework will be due at the second class after it is assigned. (E.g., Class 1 homework is due during Class 3) My goal is to grade it and post the solution within a week.
    - Problem solutions will be posted on my web site – I do not penalize late homework, but **HOMEWORK WILL NOT BE ACCEPTED AFTER THE SOLUTION IS POSTED**
  - Grading – All items are INDIVIDUAL effort
    - Homework: 25%
    - Project: 25%
    - Weekly tests: 10% each (50%)

    - Detailed grades and status will be posted on WebCT

Course Introduction (continued)

- Course project requirements will be defined in Class 5
- Project will be due in Class 15
- Project will be an individual (paper) design of a modestly complex digital system using all of the design and analysis techniques covered in the class
- Specific examples of design problems will be provided (e.g., traffic light controller, digital clock, safety interlocks on door controllers, etc.)
- Multiple projects can be completed of varying levels of difficulty

Course Introduction (continued)

- Course description:
  - Digital systems; number systems and codes; Boolean algebra; application of Boolean algebra to switching circuits; minimization of Boolean functions using algebraic, Karnaugh map, and tabular methods; programmable logic devices; sequential circuit components; design and analysis of synchronous and asynchronous sequential circuits

- Textbook
  - M. Morris Mano, Digital Design, Third Edition, Prentice Hall, Engelwood Cliffs, NJ, 2002. ISBN 0-13-062121-8

- My approach:
  - Practical, real-world examples
  - Multiple perspectives on an issue

Topics

- Fundamental concepts of digital systems
- Binary codes and number systems
- Boolean algebra
- Simplification of switching equations
- Digital device characteristics (e.g., TTL, CMOS) and design considerations
- Combinatoric logical design including LSI implementation
- Hazards, Races, and time related issues in digital design
- Flip-flops and state memory elements
- Sequential logic analysis and design
- Synchronous vs. asynchronous design
- Counters, shift register circuits
- Memory and Programmable logic
- Minimization of sequential systems
- Introduction to Finite Automata

ABET Course Objectives

By the end of this course, students should be able to:

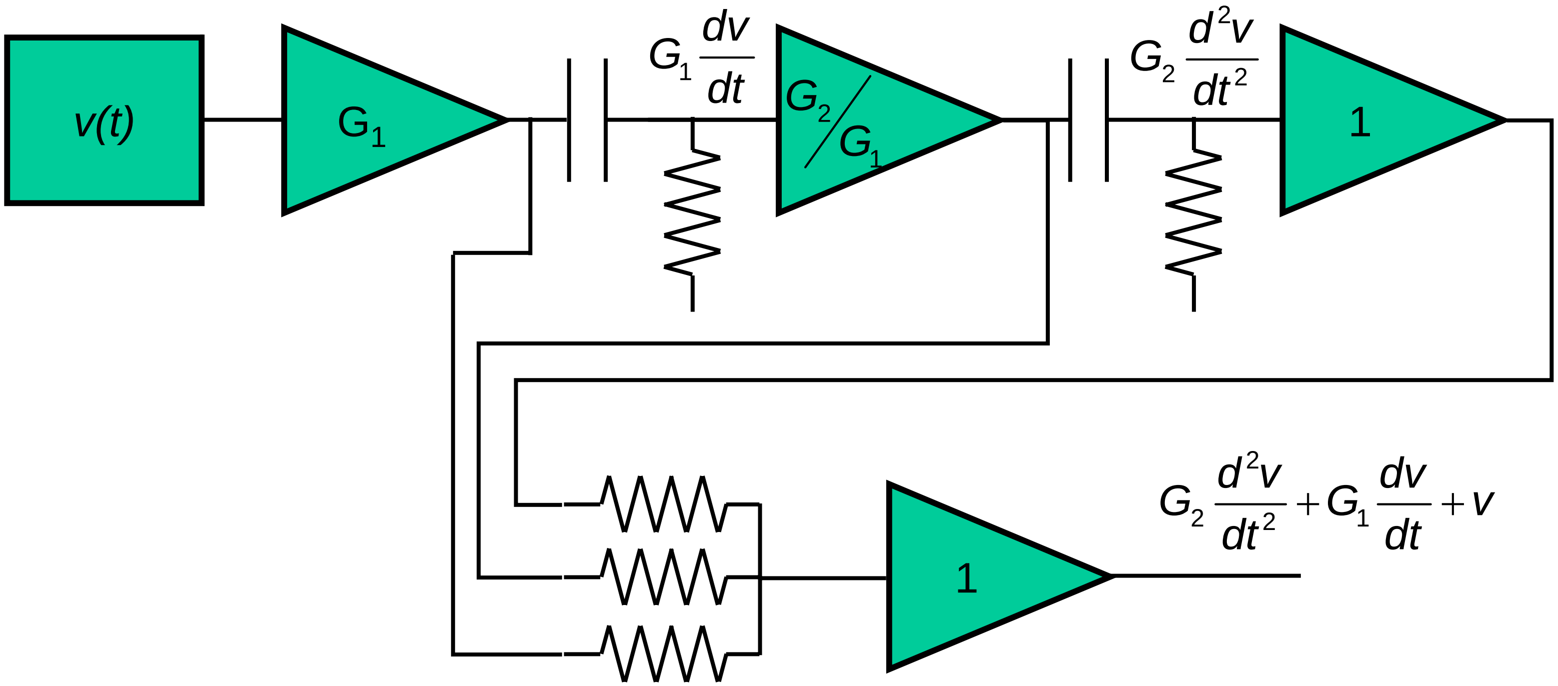
- Understand number systems and codes and their application to digital circuits; understand Boolean algebra and its application to the design and characterization of digital circuits (1A)
- Understand the mathematical characteristics of logic gates (4A)
- Use truth tables, Boolean algebra, Karnaugh maps, and other methods to obtain design equations
- Use design equations and procedures to design combinatorial and sequential systems consisting of gates and flip-flops (4C)
- Combine combinatorial circuits and flip-flops to design combinatorial and sequential system (5B)
- Consider alternatives to traditional design techniques to simplify the design process to yield innovative designs (5E)

Today’s Material

- Fundamental concepts of digital systems (Mano Chapter 1)
- Binary codes, number systems, and arithmetic (Ch 1)
- Boolean algebra (Ch 2)
- Simplification of switching equations (Ch 3)
- Digital device characteristics (e.g., TTL, CMOS)/design considerations (Ch 10)
- Combinatoric logical design including LSI implementation (Chapter 4)
- Hazards, Races, and time related issues in digital design (Ch 9)
- Flip-flops and state memory elements (Ch 5)
- Sequential logic analysis and design (Ch 5)
- Synchronous vs. asynchronous design (Ch 9)
- Counters, shift register circuits (Ch 6)
- Memory and Programmable logic (Ch 7)
- Minimization of sequential systems
- Introduction to Finite Automata

Why Digital?

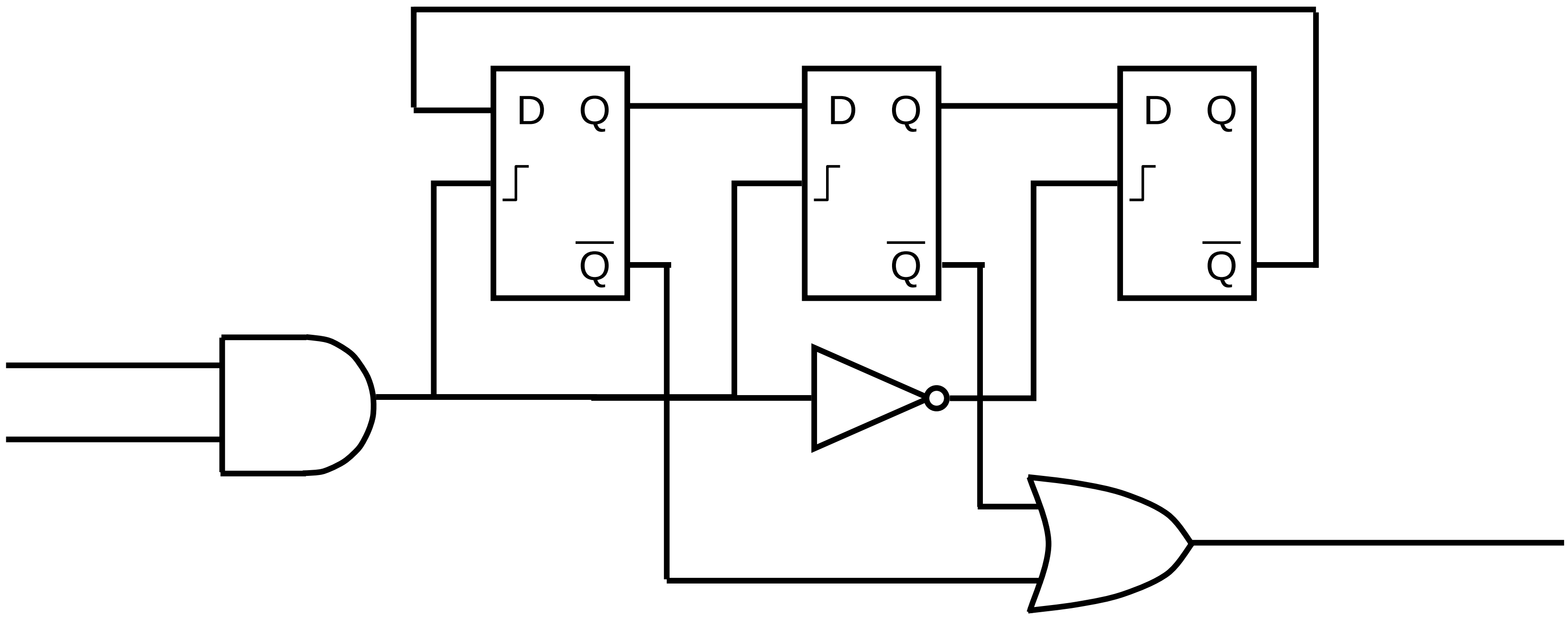
- Analog computers were the standard for simulation in the 1940s and 50s:



- Issues: precision, stability, accuracy, aging, noise, …
- Manufacturing and testing are labor intensive processes

Why Digital?

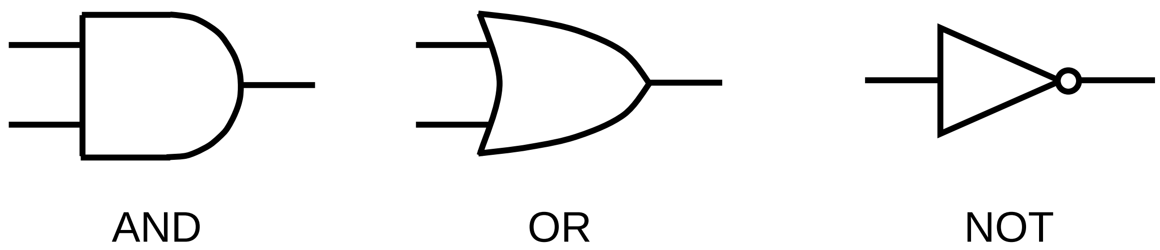
- Digital circuits have become the standard for computing, control, and many other applications



- Functions can be created with a small set of functional elements
- Designs are stable and repeatable
- Costs and size are rapidly dropping while speed and functionality increase

Simplicity of Logic Design

- Any digital logic design can be created with three fundamental building blocks:



- Signals are represented by only two states:
  - ON TRUE 1
  - OFF FALSE 0

Basic Number Systems - Decimal

- Integers:

$$23,678 = 2 \cdot 10^4 + 3 \cdot 10^3 + 6 \cdot 10^2 + 7 \cdot 10^1 + 8 \cdot 10^0$$

- Rational numbers:

$$3.1412 = 3 \cdot 10^0 + 1 \cdot 10^{-1} + 4 \cdot 10^{-2} + 1 \cdot 10^{-3} + 2 \cdot 10^{-4}$$

- In general:

$$a_3 a_2 a_1 a_0 \bullet a_{-1} a_{-2} = a_3 \cdot 10^3 + a_2 \cdot 10^2 + a_1 \cdot 10^1 + a_0 \cdot 10^0 + a_{-1} \cdot 10^{-1} + a_{-2} \cdot 10^{-2}$$

Decimal point

Basic Number Systems – Arbitrary Base

- Integers:

$$23678_{12} = 2 \cdot 12^4 + 3 \cdot 12^3 + 6 \cdot 12^2 + 7 \cdot 12^1 + 8 \cdot 12^0 = 47612_{10}$$

- Rational numbers:

$$3.1412_5 = 3 \cdot 5^0 + 1 \cdot 5^{-1} + 4 \cdot 5^{-2} + 1 \cdot 5^{-3} + 2 \cdot 5^{-4} = 3.3712_{10}$$

- In general:

$$(a_n \ldots a_1 a_0 \bullet a_{-1} \ldots a_{-m})_r = a_n \cdot r^n + \ldots + a_1 \cdot r^1 + a_0 \cdot r^0 + a_{-1} \cdot r^{-1} + \ldots + a_{-m} r^{-m}$$

“Radix” point

$$a_i \in \{0, 1, \ldots, r-1\}$$

Basic Number Systems – Base 2

- Integers:

$$1011_2 = 1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 = 11_{10}$$

- Rational numbers:

$$101.011_2 = 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 + 0\cdot 2^{-1} + 1\cdot 2^{-2} + 1\cdot 2^{-3} = 5.375_{10}$$

- In general:

$$(a_n \ldots a_1 a_0 \bullet a_{-1} \ldots a_{-m})_2 = a_n \cdot 2^n + \ldots + a_1 \cdot 2^1 + a_0 \cdot 2^0 + a_{-1} \cdot 2^{-1} + \ldots + a_{-m} 2^{-m}$$

↑ “binary” point

$$a_i \in \{0,1\}$$

Numbers to Remember

$2 \times 2 =$

$4 \times 2 =$

$8 \times 2 =$

$16 \times 2 =$

$32 \times 2 =$

$64 \times 2 =$

$128 \times 2 =$

$256 \times 2 =$

$512 \times 2 =$

$1024 \times 2 =$

$2048 \times 2 =$

$4096 \times 2 =$

$8192 \times 2 =$

$16384 \times 2 =$

$32768 \times 2 =$

65535

$.5 \times .5 =$

$.25 \times .5 =$

$.125 \times .5 =$

$.0625 \times .5 =$

$.03125 \times .5 =$

$.015625$

Basic Arithmetic

- Addition:

```
  101001
 +011011
 1000100
```

+	0	1
0	0	1
1	1	0*

* carry

- Subtraction:

```
  101000
 -010110
  010010
```

- Multiplication:

```
   1001
  x 110
   0000
  1001
 1001
 110110
```

x	0	1
0	0	0
1	0	1

Number Base Conversion

- Binary to Decimal
  - The hard way:

$$\begin{aligned} 1011_2 &= 1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 \\ &= 1\cdot 8 + 0\cdot 4 + 1\cdot 2 + 1\cdot 1 \\ &= 8 + 2 + 1 \\ &= 11_{10} \end{aligned}$$

  - An easier way:

$$\begin{aligned} 1011_2 &= 1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 \\ &= \left(\left(\left(1\right)\cdot 2 + 0\right)\cdot 2 + 1\right)\cdot 2 + 1 \\ &= 11_{10} \end{aligned}$$

Number Base Conversion

- Decimal to Binary:

$$87_{10} = 2 \bullet \left\lfloor \frac{87}{2} \right\rfloor + (87 \bmod 2) = 2 \bullet 43 + 1$$

$$43_{10} = 2 \bullet \left\lfloor \frac{43}{2} \right\rfloor + (43 \bmod 2) = 2 \bullet 21 + 1$$

$$21_{10} = 2 \bullet \left\lfloor \frac{21}{2} \right\rfloor + (21 \bmod 2) = 2 \bullet 10 + 1$$

$$10_{10} = 2 \bullet \left\lfloor \frac{10}{2} \right\rfloor + (10 \bmod 2) = 2 \bullet 5 + 0$$

$$5_{10} = 2 \bullet \left\lfloor \frac{5}{2} \right\rfloor + (5 \bmod 2) = 2 \bullet 2 + 1$$

$$2_{10} = 2 \bullet \left\lfloor \frac{2}{2} \right\rfloor + (2 \bmod 2) = 2 \bullet 1 + 0$$

$$1_{10} = 2 \bullet \left\lfloor \frac{1}{2} \right\rfloor + (1 \bmod 2) = 0 + 1$$

$$87_{10} = 1\ 0\ 1\ 0\ 1\ 1\ 1$$

Number Base Conversion – Fractional Numbers

- Decimal to Binary:

$$0.765_{10} = 0.110000111\ldots$$

$$0.765_{10} = \left(.765 - \tfrac{1}{2}\bullet\left\lfloor 2\bullet.765 \right\rfloor\right) + \tfrac{1}{2}\bullet\left\lfloor 2\bullet.765 \right\rfloor$$

$$= \left(.765 - .5\right) + \tfrac{1}{2}\bullet 1 = \tfrac{1}{2}\bullet 2\bullet.265 + \tfrac{1}{2}\bullet 1 = \tfrac{1}{2}\left(.53\right) + \tfrac{1}{2}\bullet 1$$

$$0.53_{10} = \left(.53 - \tfrac{1}{2}\bullet\left\lfloor 2\bullet.53 \right\rfloor\right) + \tfrac{1}{2}\bullet\left\lfloor 2\bullet.53 \right\rfloor$$

$$= \left(.53 - .5\right) + \tfrac{1}{2}\bullet 1 = \tfrac{1}{2}\bullet 2\bullet.03 + \tfrac{1}{2}\bullet 1 = \tfrac{1}{2}\left(.06\right) + \tfrac{1}{2}\bullet 1$$

$$.06_{10} = \left(.06 - \tfrac{1}{2}\bullet\left\lfloor 2\bullet.06 \right\rfloor\right) + \tfrac{1}{2}\bullet\left\lfloor 2\bullet.06 \right\rfloor$$

$$= \left(.06\right) + \tfrac{1}{2}\bullet 0 = \tfrac{1}{2}\bullet 2\bullet.06 + \tfrac{1}{2}\bullet 0 = \tfrac{1}{2}\left(.12\right) + \tfrac{1}{2}\bullet 0$$

$$.12_{10} = \tfrac{1}{2}\left(.24\right) + \tfrac{1}{2}\bullet 0$$

$$.24_{10} = \tfrac{1}{2}\left(.48\right) + \tfrac{1}{2}\bullet 0$$

$$.48_{10} = \tfrac{1}{2}\left(.96\right) + \tfrac{1}{2}\bullet 0$$

$$.96_{10} = \tfrac{1}{2}\left(.92\right) + \tfrac{1}{2}\bullet 1$$

$$.92_{10} = \tfrac{1}{2}\left(.84\right) + \tfrac{1}{2}\bullet 1$$

$$.84_{10} = \tfrac{1}{2}\left(.68\right) + \tfrac{1}{2}\bullet 1$$

$\vdots$

Binary Equivalents of Decimal Numbers

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

Other Popular Bases

- Octal

$$a_i \in \{0,1,2,3,4,5,6,7\}$$

- Binary-Octal mapping:

$$1562.152_8 = 001\ \ 101\ \ 110\ \ 010\ \ .\ \ 001\ \ 101\ \ 010_2$$

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

Other Popular Bases

- Hexadecimal

$$a_i \in \{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$$

- Binary-Hexadecimal Mapping

$$7EF9.2C_{16} = 0111\ \ 1110\ \ 1111\ \ 1001\ \ .\ \ 0010\ \ 1100_2$$

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	**1010**
B	**1011**
C	**1100**
D	**1101**
E	**1110**
F	**1111**

Complements

- Subtracting by adding:

$$A - B = A + (-B)$$

- How can we create *"-B"* ?

Complements

- Subtracting by adding:

$$A - B = A + (-B)$$

- How can we create *"-B"* ?
- If *B* is an integer and *B* < r^N

Define:

$$\bar{x} = r - x$$

so

$$\bar{B} = \overline{b_{N-1}} r^{N-1} + \overline{b_{N-2}} r^{N-2} + \ldots + \overline{b_2} r^2 + \overline{b_1} r^1 + \overline{b_0} r^0$$

Notice

$$B + \bar{B} + 1 = r^N$$

so

$$-B = r^N + \bar{B} + 1 \triangleq \bar{B} + 1$$

Decimal: 9's and 10's Complement

- 9's Complement

$$B = 314159_{10} \qquad B + \overline{B} = 999999_{10}$$
$$\overline{B} = 685840_{10}$$

- 10's Complement

$$B = 314159_{10} \qquad B + \overline{B} = 1000000_{10}$$
$$\overline{B} = 685841_{10}$$

- Subtraction

```
 31416
 -2718
 28698
```

- With 10's Complement

```
  31416
 +97282
 128698
```

Binary: 1's and 2's Complement

- 1's Complement

$$b \in \{0,1\}$$
$$B = 101100101_2$$
$$\overline{B} = 010011010_2$$

- 2's Complement

$$B = 101100101_2$$
$$\overline{\overline{B}} = 010011010_2 + 1 = 010011011_2$$

Working With Negative Binary Numbers

- Represent $+13_{10}$ in binary as a 6 bit number:

  001101

- How can you represent -13?

- Signed Magnitude

  101101 Add use the MSB to represent +/-

  Useful for multiplication, but not addition/subtraction

- Signed 1's Complement

  110010 Complement the bits

  Not particularly useful for arithmetic

- Signed 2's Complement

  110011 Complement the bits and add 1

  Most widely used means of dealing with signed arithmetic

Arithmetic With Signed Numbers

- Add the following numbers (all base 10) in binary using 6-bit 2's complement representation:

  +17   -23   +8   -4 = -2

010001	+17	
101001	-23	
111010	- 6	
001000	+ 8	
1000010	+ 2	Carry out of 6-bit range occurs
111100	- 4	
111110	- 2	

An Interesting Side Effect of 2's Complement

- Add the following numbers (all base 10) in binary using 6-bit 2's complement representation:

  +17 +23 -8 -4 = +28

  (intermediate results cannot be represented in 6-bit 2's complement!

  e.g., 17+23=40 > 32)

```
  010001    +17
  010111    +23
  101000    -24   (+40 is out of range)
  111000    - 8
 1100000    -32   (+32 is out of range)
  111100    - 4
10011100    +28
```

- Intermediate overflows can be tolerated, as long as the final result is within the range that can be represented

Range of Values That Can Be Represented in N-bit 2's Complement

Value	4-bit Representation
+7	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
0	0000
-1	1111
-2	1110
-3	1101
-4	1100
-5	1011
-6	1010
-7	1001
-8	1000

Value	8-bit Representation
+127	0111 1111
+126	0111 1110
+125	0111 1101
…	. . .
…	. . .
+2	0000 0010
+1	0000 0001
0	0000 0000
-1	1111 1111
-2	1111 1110
…	. . .
…	. . .
-125	1000 0011
-126	1000 0010
-127	1000 0001
-128	1000 0000

Binary Codes

- Hexadecimal or binary numbers are not easily translated into human-understandable forms, e.g.:
  - How old is a person born in $(0111\ 1101\ 0011)_2$?
  - Is it any easier to understand as $(7E3)_{16}$?

Binary Codes

- Hexadecimal or binary numbers are not easily translated into human-understandable forms, e.g.:
  - How old is a person born in $(0111\ 1101\ 0011)_2$?
  - Is it any easier to understand as $(7E3)_{16}$?

  - How about:

    $(0010\ 0000\ 0000\ 0011)_{BCD} =$

    $(0010)_2 x10^3 + (0000)_2 10^2 + (0000)_2 10^1 + (0011)_2 10^0$

Binary Codes

- Typical binary counting order:

0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111
0	000

Transitions in multiple bits may create systems issues:

- Extra bit errors on communications links
- Noise pulses in digital systems

Binary Codes

- Typical binary counting order:

0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111
0	000

Transitions in multiple bits may create systems issues:

- Extra bit errors on communications links
- Noise pulses in digital systems

- Gray code order:

0	000
1	001
2	011
3	010
4	110
5	111
6	101
7	100
0	000

Adjacent code words differ in only one bit position

Binary Codes

- Errors sometimes occur as data is being stored or transmitted.
- How can we design a system that is capable of responding to this possibility?

- Consider:

  Correct value: `0101 1001 0011`

  Result: `0111 1001 0011`

Binary Codes

- Errors sometimes occur as data is being stored or transmitted.
- How can we design a system that is capable of responding to this possibility?

- Consider:

  Correct value: `0101 1001 0011`

  Result: `0111 1001 0011`

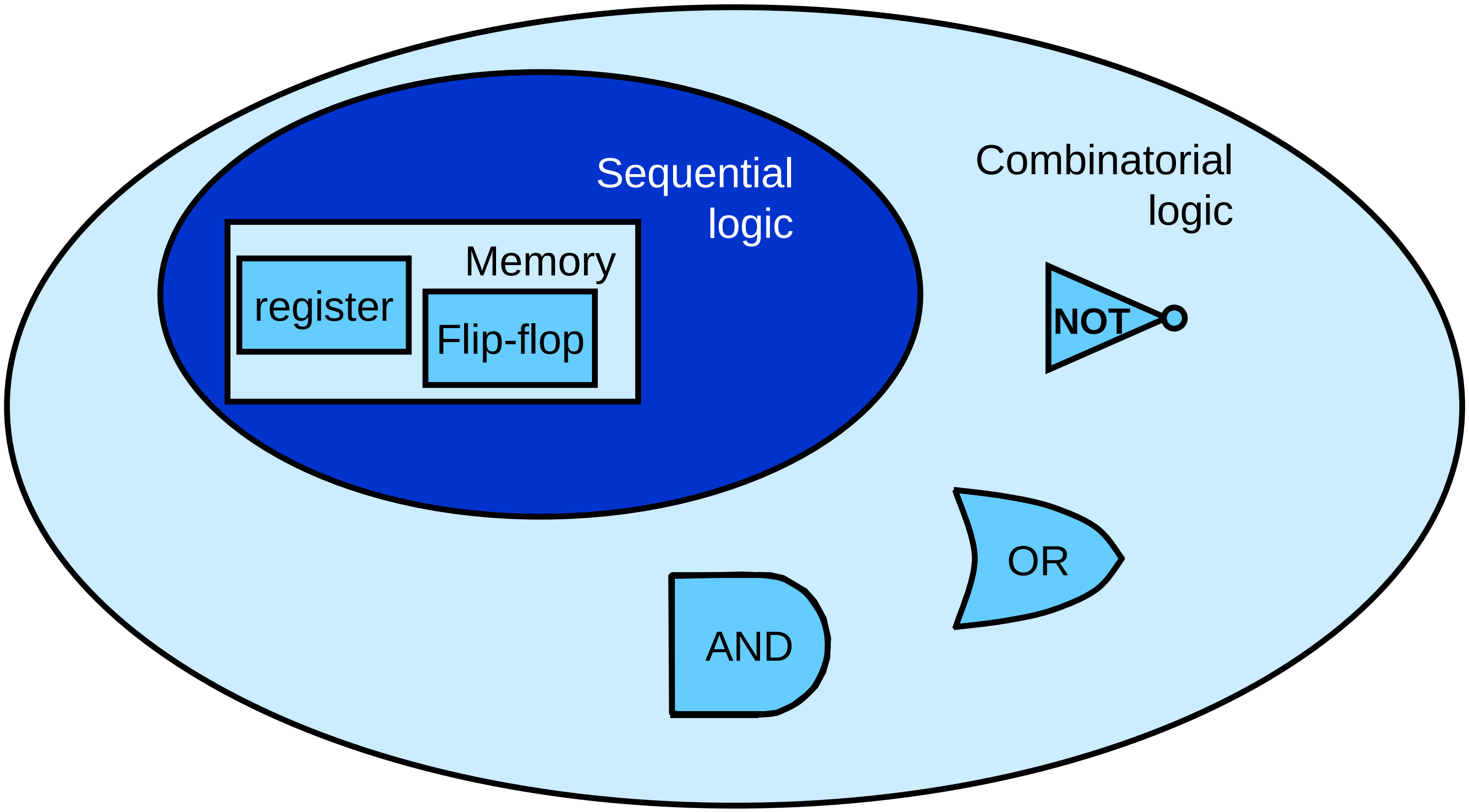
- Add redundancy – bits which convey no information, but protect other bits

  Correct value: `0101 1001 0011` **0** Even Parity - even number of 1s sent

  Result: `0111 1001 0011` **0** Parity Error – odd number of 1s received

- Parity can be even or odd.
- Parity detect a single error in a protected block

Binary Storage and Registers

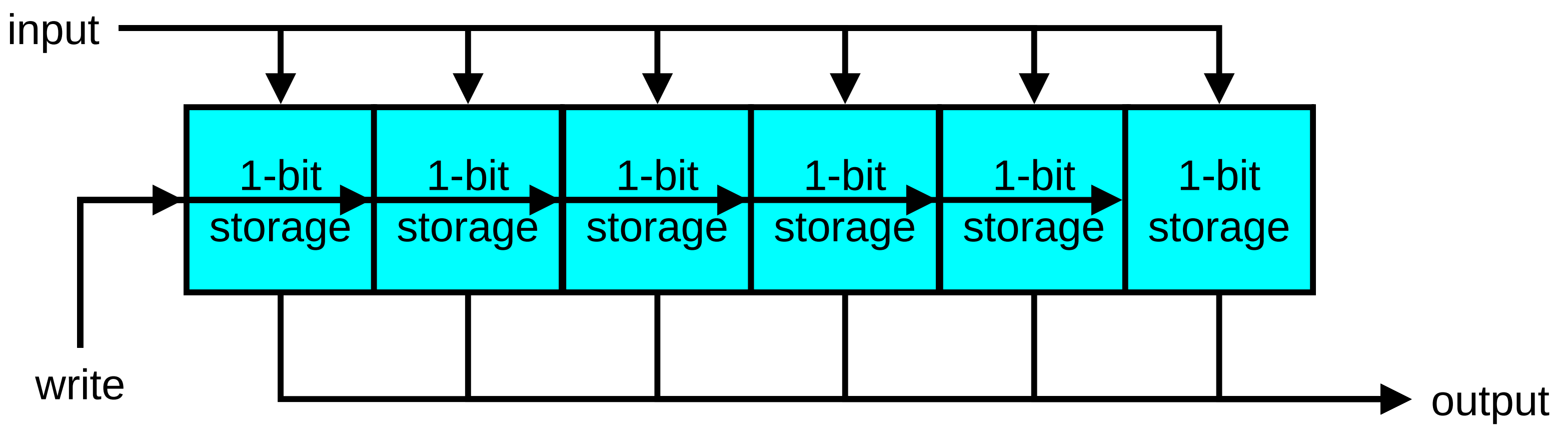


Binary Storage and Registers

- Sequential systems require that binary information be stored
- Storage is in multiples of 1 bit
- Assume that a storage element holds a value (0 or 1) until it is changed by a strobe signal

Binary Storage and Registers

- Sequential systems require that binary information be stored
- Storage is in multiples of 1 bit
- Assume that a storage element holds a value (0 or 1) until it is changed by a strobe signal
- Multiple storage elements can be used in unison to form a *register* to store associated binary information, e.g., an N-bit number

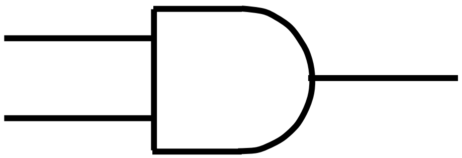


Binary Logic

- All possible combinatorial logic systems can be implemented with three functions:

A

B



A **AND** B

•	0	1
0	0	0
1	0	1

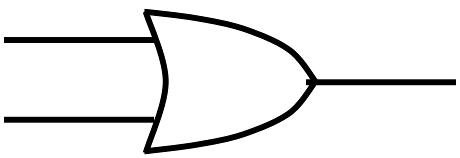
A	B	A⅄B
0	0	0
0	1	0
1	0	0
1	1	1

Binary Variables

Logical Operations

C

D



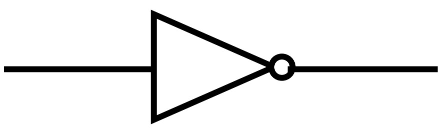
C **OR** D

+	0	1
0	0	1
1	1	1

C	D	CϒD
0	0	0
0	1	1
1	0	1
1	1	1

Truth Tables

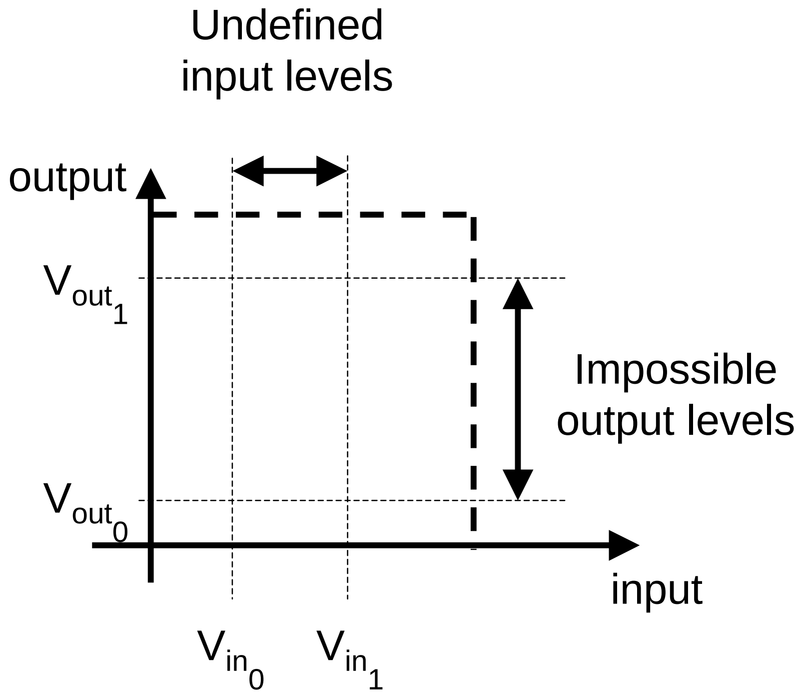
E



NOT(E)

E	¬E
0	1
1	0

Logic Levels



Summary

- Fundamental concepts of digital systems
- Binary codes, number systems, and arithmetic
- Boolean algebra
- Simplification of switching equations
- Digital device characteristics (e.g., TTL, CMOS)/design considerations
- Combinatoric logical design including LSI implementation
- Hazards, Races, and time related issues in digital design
- Flip-flops and state memory elements
- Sequential logic analysis and design
- Synchronous vs. asynchronous design
- Counters, shift register circuits
- Memory and Programmable logic
- Minimization of sequential systems
- Introduction to Finite Automata

Homework 1 – due in Class 3

- Problems 1-7, 1-11, 1-13, 1-20. Show all work