Lecture 1:
Introduction to Digital Logic Design

CSE 140: Components and Design Techniques for Digital Systems
Spring 2019

CK Cheng
Dept. of Computer Science and Engineering
University of California, San Diego
Outlines

• Class Schedule and Enrollment

• Staff
  – Instructor, TAs, Tutors

• Logistics
  – Websites, Textbooks, Grading Policy

• Motivation
  – Moore’s Law, Internet of Things, Quantum Computing

• Scope
  – Position among courses
  – Coverage
Class Schedule and Enrollment

• CSE140 A (enrollment 175, waitlist 19)
  – Lecture: TR 5-620PM, PCYNH 106
  – Discussion: F 11-1150AM, PCYNH 106
  – Final: S 1130AM-130PM, 6/8/2019

• CSE140 B (enrollment 180, waitlist 9)
  – Lecture: TR 2-320PM PCYNH 109
  – Discussion: F 8-850PM, PCYNH 109
  – Final: S 1130AM-130PM, 6/8/2019

• Waitlist: I welcome all students but have no control of the enrollment

• No discussion session on Friday 4/5/2019
Information about the Instructor

• Instructor: CK Cheng
• Education: Ph.D. in EECS UC Berkeley
• Industrial Experiences: Engineer of AMD, Mentor Graphics, Bellcore; Consultant for technology companies
• Email: ckcheng+140@ucsd.edu
• Office: Room 2130 CSE Building
• Office hours are posted on the course website
  – 12-1PM Monday; 10-1050AM Thursday
• Websites
  – http://cseweb.ucsd.edu/~kuan
  – http://cseweb.ucsd.edu/classes/sp19/cse140-a
Information about TAs and Tutors

TAs
- Wang, Ariel Xinyuan email:xiw193@ucsd.edu
- Hsu, Po-Ya email:p8hsu@ucsd.edu
- Assare, Omid email:omid@ucsd.edu

Tutors
- Lin, Xiaokang, xil671@ucsd.edu
- Liu, Hanshuang hal286@ucsd.edu
- Luo, Weisi wel205@ucsd.edu
- Nichols, Andrew ainichol@ucsd.edu
- Ren, Alissa, alren@ucsd.edu
- Zhang, Shirley, shz199@ucsd.edu
- Zhu, Zhuowen, zhz402@ucsd.edu

Office hours will be posted on the course website
Logistics: Sites for the Class

- Class website
  - http://cseweb.ucsd.edu/classes/sp19/cse140-a/index.html
  - Index: Staff Contacts and Office Hrs
- Syllabus
  - Grading policy
  - Class notes
  - Assignment: Homework and zyBook Activities
  - Exercises: Solutions and Rubrics
- Forum (Piazza): Online Discussion *make sure you have access
- Score keepers: Gradescope, TritonEd
- zyBook: UCSDCSE140ChengSpring2019
Logistics: Textbooks

Required text:
- Online Textbook: Digital Design by F. Vahid
  1. Sign in or create an account at learn.zybooks.com
  2. Enter zyBook code UCSDCSE140ChengSpring2019
  3. Fill email address with domain @ucsd.edu
  4. Fill section A or B
  5. Click Subscribe $50

Reference texts (recommended and reserved in library)
- Digital Systems and Hardware/Firmware Algorithms, Milos D. Ercegovac and Tomas Lang.
Lecture: iCliker for Peer Instruction

• I will pose questions. You will
  – Solo vote: Think for yourself and select answer
  – Discuss: Analyze problem in teams of three
    • Practice analyzing, talking about challenging concepts
    • Reach consensus
  – Class wide discussion:
    • Led by YOU (students) – tell us what you talked about in discussion that everyone should know.

• Many questions are open, i.e. no exact solutions.
  – Emphasis is on reasoning and team discussion
  – No solution will be posted
Logistics: Grading

Grade on style, completeness and correctness

• zyBook exercises: 10% (due Tuesday 2:00PM)
• iClicker: 0%
• Homework: 15% (grade based on a subset of problems)
• Midterm 1: 25% (T 4/23/19)
• Midterm 2: 25% (T 5/14/19)
• Final: 25% (1130AM-130PM, Saturday 6/8/19)
• Grading: The best of the following
  – The threshold: A- >90% ; B- >80% of 100% score
  – The curve: (A+,A,A-) top 33 ± ε% of class; (B+,B,B-) second 33 ± ε%
  – The bottom: C- above 45% of 100% score.
Logistic: grading components

• zyBook: Interactive learning experience
  – No excuse for delay (constrained by ZyBooks system)

• iClicker:
  – Clarification of the concepts and team discussion

• Homework:
  • Paper Work
  • Group discussion is encouraged. However, we are required to write individually.
    – Discount 10% loss of credit for each day after the deadline but no credit after the solution is posted.
    – Metric: Posted solutions and rubrics, but not grading results
Logistic: Midterms and Final

• Midterms: (Another) Indication of how well we have absorbed the material
  – Samples will be posted for more practices.
  – Solution and grading policy will be posted after the exam.
  – Midterm 2 is not cumulative but requires a good command of the Midterm 1 content.

• Final:
  – Two hours exam.
  – Samples will be posted for more practices.
  – Final is not cumulative but requires a good command of the whole class.
Logistic: Class Expectation

• Level 1: Definitions (zyBook)
  – Basic concepts
  – Motivation
• Level 2: Concepts and Methods (Lecture and slides)
  – Key ideas
• Level 3: Hands on Practices (Homeworks)
  – Exercises
• Level 4: Command of Materials (Samples of exams)
  – Review
Course Problems…Cheating

• What is cheating?
  – Studying together in groups is not cheating but encouraged
  – Turned-in work must be completely your own.
  – Copying someone else’s solution on a HW or Exam is cheating
  – Both “giver” and “receiver” are equally culpable

• We will be better off to work on the problem alone during the exam.
• We have to address the issue once the cheating is reported by someone (e.g. TA, Tutor, Student, etc.).
Motivation

• Microelectronic technologies have revolutionized our world: cell phones, internet, rapid advances in medicine, etc.

• The semiconductor industry has grown from $21 billions in 1985, $335 billions in 2015, to $478 billions in 2018.
The Digital Revolution

Integrated Circuit: Many digital operations on the same material

Vacuum tubes

Eniac

Moore's Law

Exponential Growth of Computation

WWII

1949

1965

Stored Program Model

1.6 x 11.1 mm
Building complex circuits

Transistor

<table>
<thead>
<tr>
<th>B</th>
<th>A</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Robert Noyce, 1927 - 1990

• Nicknamed “Mayor of Silicon Valley”
• Cofounded Fairchild Semiconductor in 1957
• Cofounded Intel in 1968
• Co-invented the integrated circuit
Gordon Moore

• Cofounded Intel in 1968 with Robert Noyce.

• Moore’s Law: the number of transistors on a computer chip doubles every 1.5 years (observed in 1965)
Technology Trends: Moore’s Law

- Since 1975, transistor counts have doubled every two years.
- Moore’s law: wider applications; larger market; higher revenue; more R&D
New Technologies

• New materials and fabrication for devices
  – Low power devices
  – Three dimensional integrated circuits
  – Graphene

• New architecture
  – Machine learning, deep learning

• Quantum computing

Understand the principles to explore the future
Artificial Intelligence

• Logic and Reasoning
• Boolean Satisfiability
  – Product of sum clauses
  – Diagnosis
• States and Sequences
  – Sequential Machines
  – Reachability
  – Controllability

One example of the applications and opportunities
Scope

The purpose of this course is that we:

• Learn the principles of digital design
• Learn to systematically debug increasingly complex designs
• Design and build digital systems
• Learn what’s under the hood of an electronic component
• Prepare for the future technology revolution
• Big idea: Coordination of many *levels of abstraction*
Principle of Abstraction

Abstraction: Hiding details when they are not important
Combinational Logic vs Sequential Network

**Combinational Logic:**

- \( y_i = f_i(x_1, \ldots, x_n) \)

**Sequential Networks**

1. Memory
2. Time Steps (Clock)

- \( y_i^t = f_i(x_1^t, \ldots, x_n^t, s_1^t, \ldots, s_m^t) \)
- \( s_i^{t+1} = g_i(x_1^t, \ldots, x_n^t, s_1^t, \ldots, s_m^t) \)
Scope: Overall Picture of CS140

Data Path Subsystem

- Memory File
- Pointer
- Select
- Mux
- ALU
- Memory
- Register
- Conditions

Control Subsystem

- Conditions
- Sequential machine
- Control
- CLK: Synchronizing Clock

BSV: Design specification and modular design methodology
# Scope

<table>
<thead>
<tr>
<th>Subjects</th>
<th>Building Blocks</th>
<th>Theory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Combinational Logic</td>
<td>AND, OR, NOT, XOR</td>
<td>Boolean Algebra</td>
</tr>
<tr>
<td>Sequential Network</td>
<td>AND, OR, NOT, FF</td>
<td>Finite State Machine</td>
</tr>
<tr>
<td>Standard Modules</td>
<td>Operators, Interconnects, Memory</td>
<td>Arithmetics, Universal Logic</td>
</tr>
<tr>
<td>System Design</td>
<td>Data Paths, Control Paths</td>
<td>Methodologies</td>
</tr>
</tbody>
</table>
Combinational Logic Basics
What is a combinational circuit?

- No memory
- Realizes one or more functions
- Inputs and outputs can only have two discrete values
  - Physical domain (usually, voltages) (Ground 0V, Vdd 1V)
  - Mathematical domain: Boolean variables (True, False)

Differentiate between different representations:

- **physical circuit**
- **schematic diagram**
- **mathematical expressions**
Binary Digital Logic

• Simplest representation is “1” and “0” (base-2).
• Choose a physical quantity to represent “1” and “0”
  – Usually voltage, but not always (e.g. current, resistance, magnetic polarization, quantum spin, …)
• Use a transistor to make the switch (operating as a digital instead of analog device)
  – Two states – on / off
  – Signals can be high voltage (“1”) or low voltage (“0”)
Basic CMOS

- Complementary Metal Oxide Semiconductor
- Invented in the 1960’s, but took over in the 80’s
- “on” means low resistance, ”off” means high resistance
- Logic “1” and Logic “0” values are arbitrary e.g. logic “1” == 1.0 V, logic “0” == 0.0 V
Transistors as Switches
The most basic CMOS gate - inverter

- When “inp” is “1”, then the nmos is on and the pmos is off – output will be ?
- When “inp” is “0”, then the nmos is off and the pmos is on – output will be ?

<table>
<thead>
<tr>
<th>inp</th>
<th>out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Equation:

\[ \text{out} = \overline{\text{in}}' \]
Basic Gates – (N)AND gate

AND \quad Y = A \& B \quad Y = AB

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

NAND \quad Y = (A \& B)' \quad Y = (AB)'

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

What kind of gate is this?

a) AND  
b) NAND  
c) Inverter  
d) None of the above
Basic Gates – (N)OR gate

**OR** \[ Y = A + B \quad Y = A \mid B \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**NOR** \[ Y = (A + B)' \quad Y = (A \mid B)' \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bubble means Invert

NOR is universal gate
Boolean Algebra

A branch of algebra in which the values of the variables belong to a set $B$ (e.g. $\{0, 1\}$), has two operations $\{+, \cdot\}$ that satisfy the following four sets of laws.

- **Associative laws**: $(a+b)+c = a+(b+c)$, $(a\cdot b)\cdot c = a\cdot (b\cdot c)$
- **Commutative laws**: $a+b = b+a$, $a\cdot b = b\cdot a$
- **Distributive laws**: $a+(b\cdot c) = (a+b)\cdot (a+c)$, $a\cdot (b+c) = a\cdot b + a\cdot c$
- **Identity laws**: $a+0 = a$, $a\cdot 1 = a$
- **Complement laws**: $a+a' = 1$, $a\cdot a' = 0$

($x'$: the complement element of $x$)
Duality

• Swap (+, ⋅) and complement all 0’s and 1’s
• If we can prove a statement using laws of Boolean algebra true, then the duality of the statement is also true.

Laws and their duals

<table>
<thead>
<tr>
<th>Laws</th>
<th>Expression 1</th>
<th>Expression 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Associative</td>
<td>(a ⋅ b) ⋅ c = a ⋅ (b ⋅ c)</td>
<td>(a + b) + c = a + (b + c)</td>
</tr>
<tr>
<td>Commutative</td>
<td>a ⋅ b = b ⋅ a</td>
<td>a + b = b + a</td>
</tr>
<tr>
<td>Distributive*</td>
<td>a ⋅ (b+c) = a ⋅ b + a ⋅ c</td>
<td>a+(b⋅c)=(a+b)⋅(a+c)</td>
</tr>
<tr>
<td>Identity</td>
<td>a ⋅ 1 = a</td>
<td>a + 0 = a</td>
</tr>
<tr>
<td>Compliment</td>
<td>a ⋅ a’ = 0</td>
<td>a + a’ = 1</td>
</tr>
</tbody>
</table>
Representations of combinational circuits: The Schematic

- What is the simplest combinational circuit that you know?
Representations of combinational circuits

Truth Table: Enumeration of all combinations

Example: AND

<table>
<thead>
<tr>
<th>id</th>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

A

\[ Y = AB \]

B
Boolean Algebra

Similar to regular algebra but defined on sets with only three basic ‘logic’ operations:

1. Intersection: AND (2-input); Operator: $\cdot$, $\&$
2. Union: OR (2-input); Operator: $+$, $|$ 
3. Complement: NOT (1-input); Operator: $\prime$, $!$

“&, |, !” Symbols in BSV
Some Def’s

- Complement: variable with a “BAR” over it or ‘ after it
  \( A’ \)
- Literal: variable or its complement
- Implicant: product of literals
  \( ABC \)
- Implicate: sum of literals
  \( (A+B+C) \)

Minterm, maxterm (implicant or implicate that includes all the inputs)

\( F(A,B,C,D): ABCD, \ (A+B+C+D) \)
**Boolean algebra and switching functions**

**Two-input AND (·)**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

For an AND gate, 0 at input blocks the other inputs and dominates the output 1 at input passes signal A

**Two-input OR (+)**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

For an OR gate, 1 at input blocks the other inputs and dominates the output 0 at input passes signal A

**One-input NOT (Complement, ’)**

<table>
<thead>
<tr>
<th>A</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

For an AND gate, 0 at input blocks the other inputs and dominates the output 1 at input passes signal A
Boolean Algebra

iClicker Q: For two Boolean variables X and Y with X=1, Y=0, what is function F(X,Y)=X+Y?

A. F(X,Y)=0
B. F(X,Y)=1
C. F(X,Y)=2
Boolean Algebra

iClicker Q: For two Boolean variables X and Y with X=1, Y=0, what is function F(X,Y)=X+X+Y?
A. F(X,Y)=0
B. F(X,Y)=1
C. F(X,Y)=2
Boolean Algebra

iClicker Q: For two Boolean variables \(X\) and \(Y\) with \(X=1\), \(Y=0\), what is function \(F(X,Y)=X+XY\)?

A. \(F(X,Y)=0\)
B. \(F(X,Y)=1\)
C. \(F(X,Y)=2\)
Boolean Algebra

iClicker Q: For two Boolean variables X and Y with X=1, Y=0, what is function F(X,Y)=(X+Y)Y?
A. F(X,Y)=0
B. F(X,Y)=1
C. F(X,Y)=2
So, what is the point of representing gates as symbols and Boolean expressions?

• Given the Boolean expression, we can draw the circuit it represents by cascading gates (and vice versa)

Logic circuit vs. Boolean Algebra Expression
Simplify the Boolean expression: Reduce the complexity of the circuit
Switching Expression and Logic Diagrams

• Switching Expression –
  – Equations - # literals, # variables, # operators
    • Literal is a variable or its complement (e.g. a, a’)
    • Variables (e.g. x)
    • Operator (e.g. +, ·)
  – Schematic / Logic Diagram - # of gates, # nets (wires), # of pins
    • Gate (and, or, etc) – can be more than 2 inputs (e.g. 3 input AND gate)
    • Net – wire that connects gates
    • Pin - input or output of a gate.
Laws and Logic Diagrams

Associativity Laws

\[(A+B) + C = A + (B+C)\]

\[(AB)C = A(BC)\]
Laws and Logic Diagrams

Distributive Laws

\[ A \cdot (B+C) = A \cdot B + A \cdot C \]

\[ A+B \cdot C = (A+B) \cdot (A+C) \]
Switching Expression and Logic

Schematic Diagram:
- 5 primary inputs
- 1 primary output
- 4 components (gates)
- 9 signal nets
- 12 pins

Boolean Algebra:
- 5 variables
- 1 expression
- 4 operators
- 5 literals

\[ y = e \cdot (a \cdot b + c \cdot d) \]
Switching Expression and Logic

Schematic Diagram:
6 primary inputs
1 primary output
4 gates (3 ANDs, 1 OR)
10 signal nets
11 pins
Nets are wires, Gates -> transistors

Switching Expression:
5 variables
1 expression
4 operators (3 ANDs, 1 OR)
6 literals

Cost: #gates, #nets, #pins

\[ y = \overline{d} \cdot \overline{e} \cdot (\overline{a} \cdot b + b' \cdot c) \]
Example: $f(a, b, c) = ab + a'c + a'b'$
Which statement is not true in general?

Schematic Diagram:
5 primary inputs
4 components (gates)
9 signal nets
12 pins

A. #primary inputs = # literals
B. #gates = # operators
C. #nets = #variables + # operators
D. #pins = # literals + 2 * # operators - 1
E. All of them

Boolean Algebra:
y = e \cdot (a \cdot b + c \cdot d)
5 literals
4 operators

Based on CK Cheng – CSE140 Spr18
• Logic circuit vs. Boolean Algebra Expression

• Simplify the Boolean expression: Reduce the complexity of the circuit
Next class

• Designing Combinational circuits