CS 61C: 
Great Ideas in Computer Architecture 
Switches, Transistors, Gates, and 
Boolean Logic 
Instructors: 
Krste Asanovic, Randy H. Katz 
http://inst.eecs.Berkeley.edu/~cs61c/fa12

Agenda

• Switching Networks, Transistors
• Gates and Truth Tables for Circuits
• Boolean Algebra
• (Logisim if there is time)
• And in Conclusion, ...

Hardware Design

• Next several weeks: how a modern processor is built, starting with basic elements as building blocks
• Why study hardware design?
  – Understand capabilities and limitations of hw in general and processors in particular
  – What processors can do fast and what they can’t do fast (avoid slow things if you want your code to run fast!)
  – Background for more in depth hw courses (CS 152)
  – Hard to know what will need for next 30 years
  – There is just so much you can do with standard processors: you may need to design own custom hw for extra performance
  – Even some commercial processors today have customizable hardware!

Synchronous Digital Systems

Hardware of a processor, such as the MIPS, is an example of a Synchronous Digital System

Synchronous:

• All operations coordinated by a central clock
  • “Heartbeat” of the system!

Digital:

• Represent all values by two discrete values
• Electrical signals are treated as 1’s and 0’s
  • 1 and 0 are complements of each other
• High /low voltage for true / false, 1 / 0
Switches: Basic Element of Physical Implementations

- Implementing a simple circuit (arrow shows action if wire changes to “1” or is asserted):

<table>
<thead>
<tr>
<th>A</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close switch (if A is “1” or asserted) and turn on light bulb (Z)</td>
<td></td>
</tr>
<tr>
<td>Open switch (if A is “0” or unasserted) and turn off light bulb (Z)</td>
<td></td>
</tr>
</tbody>
</table>

\[ Z = A \]

Switches (cont’d)

- Compose switches into more complex ones (Boolean functions):

\[ Z = A \text{ and } B \]
\[ Z = A \text{ or } B \]

Historical Note

- Early computer designers built ad hoc circuits from switches
- Began to notice common patterns in their work: ANDs, ORs, ...
- Master’s thesis (by Claude Shannon) made link between work and 19th Century Mathematician George Boole
  - Called it “Boolean” in his honor
- Could apply math to give theory to hardware design, minimization, ...

Transistors

- High voltage (\( V_{dd} \)) represents 1, or true
- Low voltage (0 volts or Ground) represents 0, or false
- Let threshold voltage (\( V_{th} \)) decide if a 0 or a 1
- If switches control whether voltages can propagate through a circuit, can build a computer
- Our switches: CMOS transistors

CMOS Transistor Networks

- Modern digital systems designed in CMOS
  - MOS: Metal-Oxide on Semiconductor
  - C for complementary: use pairs of normally-open and normally-closed switches
  - Used to be called COS-MOS for complementary-symmetry – MOS
- CMOS transistors act as voltage-controlled switches
  - Similar, though easier to work with, than relay switches from earlier era
  - Use energy primarily when switching

CMOS Transistors

- Three terminals: source, gate, and drain
  - Switch action: if voltage on gate terminal is (some amount) higher/lower than source terminal then conducting path established between drain and source terminals (switch is closed)

\[ \text{Gate} \]
\[ \text{Source} \]
\[ \text{Drain} \]

\[ n\text{-channel transistor} \]
open when voltage at Gate is low
when voltage (Gate) > voltage (Threshold)

\[ p\text{-channel transistor} \]
open when voltage at Gate is low
when voltage (Gate) > voltage (Threshold)

\[ \text{Gate} \]
\[ \text{Source} \]
\[ \text{Drain} \]

\[ \text{Gate} \]
\[ \text{Source} \]
\[ \text{Drain} \]

Note circle symbol to indicate “NOT” or “complement”

n-channel transistor
open when voltage at Gate is low
when voltage (Gate) > voltage (Threshold)

p-channel transistor
open when voltage at Gate is low
when voltage (Gate) > voltage (Threshold)

Historical Note:

- Early computer designers built ad hoc circuits from switches
- Began to notice common patterns in their work: ANDs, ORs, ...
- Master’s thesis (by Claude Shannon) made link between work and 19th Century Mathematician George Boole
  - Called it “Boolean” in his honor
- Could apply math to give theory to hardware design, minimization, ...

Transistors

- High voltage (\( V_{dd} \)) represents 1, or true
- Low voltage (0 volts or Ground) represents 0, or false
- Let threshold voltage (\( V_{th} \)) decide if a 0 or a 1
- If switches control whether voltages can propagate through a circuit, can build a computer
- Our switches: CMOS transistors

CMOS Transistor Networks

- Modern digital systems designed in CMOS
  - MOS: Metal-Oxide on Semiconductor
  - C for complementary: use pairs of normally-open and normally-closed switches
  - Used to be called COS-MOS for complementary-symmetry – MOS
- CMOS transistors act as voltage-controlled switches
  - Similar, though easier to work with, than relay switches from earlier era
  - Use energy primarily when switching

CMOS Transistors

- Three terminals: source, gate, and drain
  - Switch action: if voltage on gate terminal is (some amount) higher/lower than source terminal then conducting path established between drain and source terminals (switch is closed)

\[ \text{Gate} \]
\[ \text{Source} \]
\[ \text{Drain} \]
CMOS Circuit Rules

- Don’t pass weak values => Use Complementary Pairs
  - N-type transistors pass weak 1’s ($V_{dd} - V_{th}$)
  - N-type transistors pass strong 0’s (ground)
  - Use N-type transistors only to pass 0’s (N for negative)
- Converse for P-type transistors: Pass weak 0’s, strong 1’s ($V_{th} - V_{dd}$)
  - P-type transistors pass weak 0’s (
  - P-type transistors pass strong 1’s (ground)
  - Use P-type transistors only to pass 1’s (P for positive)
- Use pairs of N-type and P-type to get strong values
- Never leave a wire undriven
  - Make sure there’s always a path to $V_{dd}$ or gnd
- Never create a path from $V_{dd}$ to ground (ground)

EECS Grading Policy

- [http://www.eecs.berkeley.edu/Policies/ugrad.grading.shtml](http://www.eecs.berkeley.edu/Policies/ugrad.grading.shtml)
  "A typical GPA for courses in the lower division is 2.7. This GPA would result, for example, from 17% A’s, 50% B’s, 20% C’s, 10% D’s, and 3% F’s. A class whose GPA falls outside the range 2.5 - 2.9 should be considered atypical."
- Spring 2011: GPA 2.85
  - 24% A’s, 49% B’s, 18% C’s, 6% D’s, 3% F’s
- Job/Intern Interviews: They grill you with technical questions, so it’s what you say, not your GPA (CS61c gives you good stuff to say)

EECS Grading Policy

<table>
<thead>
<tr>
<th></th>
<th>Fall</th>
<th>Spring</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011</td>
<td>2.72</td>
<td>2.85</td>
</tr>
<tr>
<td>2010</td>
<td>2.81</td>
<td>2.81</td>
</tr>
<tr>
<td>2009</td>
<td>2.71</td>
<td>2.81</td>
</tr>
<tr>
<td>2008</td>
<td>2.95</td>
<td>2.74</td>
</tr>
<tr>
<td>2007</td>
<td>2.67</td>
<td>2.76</td>
</tr>
</tbody>
</table>

Administrivia

- Regrade window opens today and closes by next Wednesday’s lecture
  - Best to return to your lab/discussion TA (parallel processing!)
- Coming labs and HWs: build up for Project #3
  - Last week’s lab: Cache blocking
  - This week’s lab: SIMD instruction set
  - This week’s HW: Cache blocking and SIMD practice
  - Next week’s lab: OpenMP thread programming

CS61c in the News

Computers meet Cameras

- [http://mlb.mlb.com/photos/gigapan/?gpId=a2be33bf9acd8730e8a086e8c1f26d8d&ps=1](http://mlb.mlb.com/photos/gigapan/?gpId=a2be33bf9acd8730e8a086e8c1f26d8d&ps=1)

Agenda

- Switching Networks, Transistors
- Gates and Truth Tables for Circuits
- Boolean Algebra
- (Logisim if there is time)
- And in Conclusion, ...

MOS Networks

- $p$-channel transistor
  - closed when voltage at Gate is low
  - opens when: $voltage(Gate) > voltage(Threshold)$
- $n$-channel transistor
  - open when voltage at Gate is low
  - closes when: $voltage(Gate) > voltage(Threshold)$

Called an **inverter** or **not gate**
MOS Networks

- **n-channel transistor**
  - open when voltage at Gate is low
  - closed when voltage(Gate) > voltage(Source) = +

- **p-channel transistor**
  - closed when voltage at Gate is low
  - opens when voltage(Gate) < voltage(Source) = -

What is the relationship between x and y?

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

Called an inverter or not gate

---

**P = \( \frac{1}{2} \) C V^2 f**

- Dynamic Energy (when switching) is proportional to Capacitance * Voltage^2
- Since pulse is 0 -> 1 -> 0 or 1 -> 0 -> 1, Energy of a single transition is proportional to \( \frac{1}{2} \) * Capacitance * Voltage^2
- Power is just energy per transition times frequency of transitions: proportional to \( \frac{1}{2} \) * Capacitance * Voltage^2 * Frequency

---

Two Input Networks

- **Student Roulette**

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>3v</td>
<td>0v</td>
<td>3v</td>
</tr>
<tr>
<td>0v</td>
<td>3v</td>
<td>0v</td>
</tr>
<tr>
<td>3v</td>
<td>0v</td>
<td>3v</td>
</tr>
</tbody>
</table>

Called NAND gate (NOT AND)

Called NOR gate (NOT OR)

---

Truth Tables

List outputs for all possible inputs

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>F(0,0,0,0)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>F(0,0,0,1)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>F(0,0,1,0)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>F(0,0,1,1)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>F(0,1,0,0)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>F(0,1,0,1)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>F(0,1,1,0)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>F(0,1,1,1)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>F(1,0,0,0)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>F(1,0,0,1)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>F(1,0,1,0)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>F(1,0,1,1)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>F(1,1,0,0)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>F(1,1,0,1)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>F(1,1,1,0)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>F(1,1,1,1)</td>
</tr>
</tbody>
</table>
Truth Table Example #1:
y = F(a, b): 1 iff a ≠ 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>0</td>
</tr>
</tbody>
</table>

Truth Table Example #2:
2-bit Adder

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C2</th>
<th>C1</th>
<th>C0</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>B0</td>
<td>+</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

How Many Rows?

Truth Table Example #3:
32-bit Unsigned Adder

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>000 ... 0</td>
<td>000 ... 0</td>
<td>000 ... 00</td>
</tr>
<tr>
<td>000 ... 0</td>
<td>000 ... 1</td>
<td>000 ... 01</td>
</tr>
<tr>
<td>111 ... 1</td>
<td>111 ... 1</td>
<td>111 ... 10</td>
</tr>
</tbody>
</table>

Truth Table Example #4:
3-input Majority Circuit

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

This is called Sum of Products form;
Just another way to represent the TT as a logical expression

More simplified forms (fewer gates and wires)

Combinational Logic Symbols

- Common combinational logic systems have standard symbols called logic gates
  - Buffer, NOT
    - A → z
  - AND, NAND
    - A ⋅ B → z
  - OR, NOR
    - A ⊕ B → z

Easy to implement with CMOS transistors (the switches we have available and use most)

Agenda

- Switching Networks, Transistors
- Gates and Truth Tables for Circuits
- Boolean Algebra
- (Logisim if there is time)
- And in Conclusion, ...
Boolean Algebra

• Use plus for OR
  – “logical sum”
• Use product for AND (a \& b or implied via \text{and})
  – “logical product”
• “Hat” to mean complement (NOT)
• Thus
  \[
  ab + a + \overline{c} = a \text{AND b OR a OR (NOT c)}
  \]

Laws of Boolean Algebra

\[
\begin{align*}
X \overline{X} &= 0 \\
X 0 &= 0 \\
X 1 &= X \\
X X &= X \\
Y \overline{Y} &= 0 \\
X (Y \overline{Y}) &= X \\
\overline{Y} X &= X \overline{Y} \\
\overline{X Y} &= \overline{X} + \overline{Y} \\
X + X &= X \\
X + 0 &= X \\
0 + X &= X \\
X + X &= X \\
X = 1 \\
Y = 0 \\
Z = 1 \\
X + Y + Z &= X + (Y + Z) \\
X Y Z &= (X Y) Z \\
(X + Y) + (X + Z) &= (X + Y) + (X + Z) \\
X Y Z &= X Y Z \\
X Y Z &= X Y Z \\
X Y Z &= X Y Z \\
X Y Z &= X Y Z
\end{align*}
\]

Complementarity

Laws of 0’s and 1’s Identities

Idempotent Laws Commutativity

Associativity Distribution

Uniting Theorem United Theorem v. 2

DeMorgan’s Law

Boolean Algebraic Simplification Example

\[
y = ab + a + c
\]

\[
\begin{align*}
ab + a + c &= a(b + 1) + c \\
&= a(b + 1 + c)
\end{align*}
\]

Agenda

• Switching Networks, Transistors
• Gates and Truth Tables for Circuits
• Boolean Algebra
• And in Conclusion, …
Agenda

- Switching Networks, Transistors
- Gates and Truth Tables for Circuits
- Boolean Algebra
- (Logisim if there is time)
- And in Conclusion, ...

Logisim

- Free schematic capture/logic simulation program in Java
  - "A graphical tool for designing and simulating logic circuits"
  - Search and download version 2.7.1, online tutorial
    - ozark.hendrix.edu/~burch/logisim/
- Drawing interface based on toolbar
  - Color-coded wires aid in simulating and debugging a circuit
  - Wiring tool draws horizontal and vertical wires, automatically connecting to components and to other wires.
- Circuit layouts used as "subcircuits" of other circuits, allowing hierarchical circuit design
- Included circuit components: inputs and outputs, gates, multiplexers, arithmetic circuits, flip-flops, RAM memory

Logisim Wires

- Blue wires: value at that point is "unknown"
- Gray wires: not connected to anything
- OK when in process of building a circuit
- When finished => wires not be blue or gray
- If connected, all wires should be green
  - Bright green a 1
  - Dark green a 0

Common Mistakes in Logisim

- Connecting wires together
- Using input for output
- Connecting to edge without connecting to actual input
  - Unexpected direction of input

And in Conclusion, ...

- Real world voltages are analog, but are quantized to represent logic 0 and logic 1
- Transistors are just switches, combined to form gates: AND, OR, NOT, NAND, NOR
- Truth table can be mapped to gates for combinational logic design
- Boolean algebra allows minimization of gates