Office Hours

HW1

CSUGLab
How to implement a desired function?

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>out</th>
<th>minterm</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>\overline{a} \overline{b} \overline{c}</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>\overline{a} \overline{b} \overline{c}</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>\overline{a} \overline{b} \overline{c}</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>\overline{a} \overline{b} \overline{c}</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>\overline{a} \overline{b} \overline{c}</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>\overline{a} \overline{b} \overline{c}</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>\overline{a} \overline{b} \overline{c}</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>\overline{a} \overline{b} \overline{c}</td>
</tr>
</tbody>
</table>

sum of products:  
• OR of all minterms where out=1

corollary: *any* combinational circuit *can be* implemented in two levels of logic (ignoring inverters)
How does one find the most efficient equation?
– Manipulate algebraically until...?
– Use Karnaugh maps (optimize visually)
– Use a software optimizer

For large circuits
– Decomposition & reuse of building blocks
• Voting Machine!
  – optical scan (thanks FL)

• Assume:
  – vote is recorded on paper by filling a circle
  – fixed number of choices
  – don’t worry about “invalids”

<table>
<thead>
<tr>
<th>Name</th>
<th>Votes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Al Franken</td>
<td></td>
</tr>
<tr>
<td>Bill Clinton</td>
<td></td>
</tr>
<tr>
<td>Condi Rice</td>
<td></td>
</tr>
<tr>
<td>Dick Cheney</td>
<td></td>
</tr>
<tr>
<td>Eliot Spitzer</td>
<td></td>
</tr>
<tr>
<td>Fred Upton</td>
<td></td>
</tr>
<tr>
<td>Lizard People</td>
<td>X</td>
</tr>
</tbody>
</table>

Write-in
Voting Machine Components

5 Essential Components?

- **Input**: paper with at exactly one mark
- **Datapath**: process current ballot
- **Output**: a number the supervisor can record
- **Memory & control**: none for now

Ballots

The 3410 optical scan vote counter reader machine
Photo-sensitive transistor

- photons replenish gate depletion region
- can distinguish dark and light spots on paper

• Use array of N sensors for voting machine input
7-Segment LED

- photons emitted when electrons fall into holes
detect N → 8
N might be large
• Routing wires is expensive

More efficient encoding?
• Base 10 - Decimal

• Just as easily use other bases
  – Base 2 - Binary
  – Base 8 - Octal
  – Base 16 - Hexadecimal
• Base conversion via repetitive division
  – Divide by base, write remainder, move left with quotient
• Base conversion via repetitive division
  – Divide by base, write remainder, move left with quotient
• Base conversion via repetitive division
  – Divide by base, write remainder, move left with quotient
Implementation . . .

- assume 8 choices, exactly one mark detected

3-bit encoder
(8-to-3)
Ballot Reading
7-Segment LED Decoder

3 inputs
- encode 0 – 7 in binary

7 outputs
- one for each LED
<table>
<thead>
<tr>
<th>b2</th>
<th>b1</th>
<th>b0</th>
<th>d6</th>
<th>d5</th>
<th>d4</th>
<th>d3</th>
<th>d2</th>
<th>d1</th>
<th>d0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

7 Segment LED Decoder Implementation
The 3410 optical scan vote counter reader machine
Addition works the same way regardless of base

- Add the digits in each position
- Propagate the carry

183
+ 254

001110
+ 011100

011100
Half Adder

- Adds two 1-bit numbers
- Computes 1-bit result and 1-bit carry

A \rightarrow B \rightarrow C \rightarrow R
Full Adder
• Adds three 1-bit numbers
• Computes 1-bit result and 1-bit carry
• Can be cascaded
4-Bit Full Adder

- Adds two 4-bit numbers and carry in
- Computes 4-bit result and carry out
- Can be cascaded
4-bit Adder

\[ \begin{align*}
& A_3 & B_3 \\
& A_2 & B_2 \\
& A_1 & B_1 \\
& A_0 & B_0 \\
\end{align*} \]

\[ \begin{align*}
& R_3 \\
& R_2 \\
& R_1 \\
& R_0 \\
\end{align*} \]
We can now implement any combinational (combinatorial) logic circuit

- Decompose large circuit into manageable blocks
  - Encoders, Decoders, Multiplexors, Adders, ...
- Design each block
  - Binary encoded numbers for compactness
- Can implement circuits using NAND or NOR gates
- Can implement gates using use P- and N-transistors