Circuits & Numbers

Kevin Walsh
CS 3410, Spring 2010
Computer Science
Cornell University

See: P&H Chapter 2.4, 2.5, 3.2, C.5
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”
5 Essential Components?

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

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
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
Counting
• 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

i0  0
i1  1
i2  2
i3  3
i4  4
i5  5
i6  6
i7  7

3-bit encoder
(8-to-3)

encoder
b0
b1
b2
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>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>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
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

0011110
+ 011100

0111000
Half Adder

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