lecture 4

Combinational logic  2

- ROM

- arithmetic circuits

- arithmetic logic unit (ALU)

January 20, 2016
Last lecture: truth tables, logic gates & circuits
Recall multiplexor (selector)

\[ Y = \overline{S} \cdot A + S \cdot B \]

We will use this several times later.
"Read-only Memory"
(leftover topic from last lecture)

Sometimes we can think of a circuit as a "hardwired" memory (read only).

Note: the order of the $A_1 A_0$ variables matters.
Recall: binary arithmetic

\[
\begin{array}{c}
C_{n-1} \ldots C_2 C_1 C_0 \\
A_{n-1} \ldots A_2 A_1 A_0 \\
+ B_{n-1} \ldots B_2 B_1 B_0 \\
\hline
S_{n-1} \ldots S_2 S_1 S_0
\end{array}
\]

Notes:

- Co = 0

- A, B could represent signed or unsigned numbers
Let's build an "adder" circuit.

\[
\begin{array}{cccc}
C_{n-1} & \ldots & C_2 & C_1 \\
A_{n-1} & \ldots & A_2 & A_1 & A_0 \\
B_{n-1} & \ldots & B_2 & B_1 & B_0 \\
\hline
S_{n-1} & \ldots & S_2 & S_1 & S_0
\end{array}
\]

\[
\begin{array}{c|cc}
A_0 & B_0 & S_0 & C_1 \\
\hline
0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1
\end{array}
\]

\[
S_0 = \\
C_1 = 
\]
Half Adder

\[ S = A \oplus B \]

\[ C = A \cdot B \]
full adder

<table>
<thead>
<tr>
<th>$A_k$</th>
<th>$B_k$</th>
<th>$C_k$</th>
<th>$S_k$</th>
<th>$C_{k+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Ripple Adder

If $n = 32$, then we can have a long delay as carries propagate through the circuit. We'll return to this later.
As I mentioned before, the interpretation of the S bit string depends on whether the A and B bit strings are signed or unsigned.

However, the full adder circuit does not depend on whether A and B are signed or unsigned.
Overflow

We still might want to know if we have "overflowed":

e.g. - if the sum of two positive numbers yields a negative

- if the sum of two negative numbers yields a positive

How can we detect these two cases? (see Exercises 2)
TODO TODAY

- encoder
- decoder
- n-bit multiplexor
- fast adder
- ALU
Encoder

<table>
<thead>
<tr>
<th>$A_{m-1} \ldots A_3 A_2 A_1 A_0$</th>
<th>$Y_n \ldots Y_0$</th>
</tr>
</thead>
</table>

... many bits  

... code (few bits)
Encoder  Example 1

Panel with five buttons

<table>
<thead>
<tr>
<th>b_4</th>
<th>b_3</th>
<th>b_2</th>
<th>b_1</th>
<th>b_0</th>
<th>Y_2</th>
<th>Y_1</th>
<th>Y_0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

This assumes only one button can be pressed at any time.
This allows two buttons to be pressed at the same time (and encodes the one with the highest index).
Encoder Example 2

panel with ten buttons

display e.g. on a digital watch, calculator, etc.
Each light $L_i$ is turned on (1) by some set of button presses.

Each button $b_k$ turns on (1) some set of lights.
code word (in this example, it specifies which output is 1)
2-bit multiplexor

Decoder

Notation:
More general example (2-bit multiplexor)

Selects from four n-bit inputs. For each Ai, Bi, Ci, Di, we replicate the circuit on the previous slide, but use the same decoder circuit.
We will next look at some examples of how multiplexors are used.
Recall the ripple adder.

The main problem is that it is slow.
How to speed up the adder?

Instead of one 32 bit adder, think of two 16 bit adders.

We can compute the result of each, in half the time. (However, if C16 = 1, then we have to wait for it to ripple through.)
Tradeoffs: we chop the time in half (almost, why?) but it increases the number of gates by more than 50% (why?). Note we can repeat this idea (recursion).
Subtraction

\[
A_{n-1} \ldots A_2 A_1 A_0
\]

\[
- B_{n-1} \ldots B_2 B_1 B_0
\]

\[
S_{n-1} \ldots S_2 S_1 S_0
\]

\[x - y = x + (-y)\]

Invert bits and add 1.
Invert bits and add 1.

When $B_{\text{invert}}$ is 1, this adds 1 by setting $C_0$ to 1.
\[ A_{n-1} \ldots A_2 A_1 A_0 \]
\[ B_{n-1} \ldots B_2 B_1 B_0 \]
\[ S_{n-1} \ldots S_2 S_1 S_0 \]

\[ n = 32 \]

See Exercises 2
Let's include a bitwise AND and OR.
Arithmetic Logic Unit (ALU)
Announcements

- 193 registered (172 seats in room)

- Quiz 1

The solutions, grading scheme, and grades were posted yesterday. You should know this b/c you should be subscribed to news on my courses. I will not be posting stuff on Facebook.

The TAs were instructed **not** to take off points for trivial errors if it was clear you understood what you were doing. If your Quiz was not graded according to this guideline, let me know by resubmitting it with a yellow sticky explaining the problem. **Please do not make such requests for marginal judgment calls:** similar to professional sports ("the ruling on the field stands"). And, I reserve the right to regrade the whole exam, if I disagree with you.