Robot rides bike! ⇒
The PRIMER-V2 robot is capable of starting from a stopped position, start riding, follows a path specified by a controller, and can stop without falling! Very very cool…
Review

• Use this table and techniques we learned to transform from 1 to another
Today

• Data Multiplexors
• Arithmetic and Logic Unit
• Adder/Subtractor
Data Multiplexor (here 2-to-1, n-bit-wide)

“mux”
N instances of 1-bit-wide mux

How many rows in TT?

\[ c = \overline{s}a\overline{b} + \overline{s}ab + s\overline{a}b + sab \]
\[ = \overline{s}(a\overline{b} + ab) + s(\overline{a}b + ab) \]
\[ = \overline{s}(a(\overline{b} + b)) + s((\overline{a} + a)b) \]
\[ = \overline{s}(a(1) + s((1)b)) \]
\[ = \overline{s}a + sb \]
How do we build a 1-bit-wide mux?

\[ \overline{s}a + sb \]
4-to-1 Multiplexor?

How many rows in TT?

\[ e = \overline{s_1 s_0} a + \overline{s_1 s_0} b + s_1 \overline{s_0} c + s_1 s_0 d \]
Is there any other way to do it?

Hint: NCAA tourney!

Ans: Hierarchically!
Arithmetic and Logic Unit

• Most processors contain a special logic block called “Arithmetic and Logic Unit” (ALU)

• We’ll show you an easy one that does ADD, SUB, bitwise AND, bitwise OR

\[
\begin{align*}
\text{when } S=00, & \quad R=A+B \\
\text{when } S=01, & \quad R=A-B \\
\text{when } S=10, & \quad R=A \text{ AND } B \\
\text{when } S=11, & \quad R=A \text{ OR } B
\end{align*}
\]
Our simple ALU
Adder/Subtractor Design -- how?

• Truth-table, then determine canonical form, then minimize and implement as we’ve seen before

• Look at breaking the problem down into smaller pieces that we can cascade or hierarchically layer
Adder/Subtracter – One-bit adder LSB...

\[
\begin{array}{lll}
\text{a}_3 & \text{a}_2 & \text{a}_1 \\
\text{b}_3 & \text{b}_2 & \text{b}_1 \\
\hline
\text{s}_3 & \text{s}_2 & \text{s}_1 \\
\end{array}
\]

\[
\begin{array}{ll|ll}
a_0 & b_0 & s_0 & c_1 \\
0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 \\
\end{array}
\]

\[
\begin{align*}
s_0 &= \\
c_1 &=
\end{align*}
\]
Adder/Subtracter – One-bit adder (1/2)…

\[
\begin{array}{cccccc}
\text{a}_i & \text{b}_i & \text{c}_i & \text{s}_i & \text{c}_{i+1} \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 \\
1 & 1 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{align*}
\text{s}_i &= \\
\text{c}_{i+1} &=
\end{align*}
\]
Adder/Subtractor – One-bit adder (2/2)…

\[ s_i = \text{XOR}(a_i, b_i, c_i) \]
\[ c_{i+1} = \text{MAJ}(a_i, b_i, c_i) = a_i b_i + a_i c_i + b_i c_i \]
N 1-bit adders $\Rightarrow$ 1 N-bit adder

What about overflow?

Overflow = $c_n$?
What about overflow?

- Consider a 2-bit signed # & overflow:
  - \(10 = -2 + -2 \) or \(-1\)
  - \(11 = -1 + -2 \) only
  - \(00 = 0 \) NOTHING!
  - \(01 = 1 + 1 \) only

- Highest adder
  - \(C_1 = \) Carry-in = \(C_{in}\), \(C_2 = \) Carry-out = \(C_{out}\)
  - No \(C_{out}\) or \(C_{in}\) \(\Rightarrow\) NO overflow!

- \(C_{in}\), and \(C_{out}\) \(\Rightarrow\) NO overflow!

- \(C_{in}\), but no \(C_{out}\) \(\Rightarrow\) \(A,B\) both > 0, overflow!
- \(C_{out}\), but no \(C_{in}\) \(\Rightarrow\) \(A,B\) both < 0, overflow!
What about overflow?

- Consider a 2-bit signed # & overflow:
  
  \[
  \begin{align*}
  10 & = -2 \\
  11 & = -1 \\
  00 & = 0 \\
  01 & = 1
  \end{align*}
  \]

- Overflows when...
  
  - $C_{\text{in}}$, but no $C_{\text{out}} \Rightarrow A, B$ both $> 0$, overflow!
  
  - $C_{\text{out}}$, but no $C_{\text{in}} \Rightarrow A, B$ both $< 0$, overflow!

\[
\text{overflow} = C_n \ XOR \ C_{n-1}
\]
Extremely Clever Subtractor

XOR serves as conditional inverter!
Peer Instruction

1) Truth table for mux with 4-bits of signals has $2^4$ rows

2) We could cascade N 1-bit shifters to make 1 N-bit shifter for sll, srl
Peer Instruction Answer

1) Truth table for mux with 4-bits of signals controls 16 inputs, for a total of 20 inputs, so truth table is $2^{20}$ rows... **FALSE**

2) We could cascade N 1-bit shifters to make 1 N-bit shifter for sll, srl ... **TRUE**

---

1) Truth table for mux with 4-bits of signals is $2^4$ rows long

2) We could cascade N 1-bit shifters to make 1 N-bit shifter for sll, srl

<table>
<thead>
<tr>
<th></th>
<th>a) 12</th>
<th>b) FT</th>
<th>c) TF</th>
<th>d) TT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FF</td>
<td>FT</td>
<td>TF</td>
<td>TT</td>
</tr>
</tbody>
</table>
“And In conclusion…”

• Use muxes to select among input
  • $S$ input bits selects $2^S$ inputs
  • Each input can be $n$-bits wide, indep of $S$

• Can implement muxes hierarchically

• ALU can be implemented using a mux
  • Coupled with basic block elements

• N-bit adder-subtractor done using $N$ 1-bit adders with XOR gates on input
  • XOR serves as conditional inverter