Mux Hierarchical Design (more inputs)

- How do we build a mux with 4 inputs?

A: \( S_0 \)
B: \( S_1 \)
C: Either
Mux Hierarchical Design (more inputs)

- How do we build a mux with 4 inputs?

Diagram:

- A: $S_0$
- B: $S_1$
- C: Either

Options:

- A
- B
- C
- D

- A: $S_0$
- B: $S_1$
- C: Either
Building an ALU (Part 2):

Happy Monday!

Exam 1 covers through today’s lecture + delay
2 parts: short answer & verilog
State – the central concept of computing

Computer can do 2 things
1) Store state
2) Manipulate state (Combine arithmetic and logical operations into one unit)
Today’s lecture

- We’ll finish the 32-bit ALU today!
  - 32-bit ALU specification

- Complete 1-bit ALU
- Assembling them to make 32-bit ALU
- Handling flags:
  - zero, negative, overflow
A specification for a 32-bit ALU

module alu32(out, overflow, zero, negative, A, B, control);
output [31:0] out;
output overflow, zero, negative;
input [31:0] A, B;
input [2:0] control;

Did overflow occur?
Is the output equal to zero?
Is the output negative?
Use a modular 1-bit ALU to build 32-bit ALU

- Previously we showed 1-bit adder/subtractor, 1-bit logic unit
  - Time to put them together.

```verilog
module alu1(out, carryout, A, B, carryin, control);
    output      out, carryout;
    input       A, B, carryin;
    input [2:0] control;
endmodule
```

<table>
<thead>
<tr>
<th>control</th>
<th>out_{i}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>undefined</td>
</tr>
<tr>
<td>1</td>
<td>undefined</td>
</tr>
<tr>
<td>2</td>
<td>A_i + B_i</td>
</tr>
<tr>
<td>3</td>
<td>A_i - B_i</td>
</tr>
<tr>
<td>4</td>
<td>A_i AND B_i</td>
</tr>
<tr>
<td>5</td>
<td>A_i OR B_i</td>
</tr>
<tr>
<td>6</td>
<td>A_i NOR B_i</td>
</tr>
<tr>
<td>7</td>
<td>A_i XOR B_i</td>
</tr>
</tbody>
</table>
Addition + Subtraction in one circuit (1-bit Arithmetic Unit)

- When Sub = 0, Y = B and Cin = 0. Result = A + B + 0 = A + B.
- When Sub = 1, Y = ~B and Cin = 1. Result = A + ~B + 1 = A – B.

Which parts belong in inside the 1-bit ALU?

A) the Full Adder,  B) the XOR gate,  C) Both,  D) Neither
Addition + Subtraction in one circuit (1-bit Arithmetic Unit)

- When Sub = 0, Y = B and Cin = 0. Result = A + B + 0 = A + B.
- When Sub = 1, Y = ~B and Cin = 1. Result = A + ~B + 1 = A − B.

What should we do with the full adder’s Cin input?
- A) Connect to Sub,  B) Connect to 1-bit ALU’s carryin
Addition + Subtraction in one circuit (1-bit Arithmetic Unit)

- When Sub = 0, Y = B and Cin = 0. Result = A + B + 0 = A + B.
- When Sub = 1, Y = ~B and Cin = 1. Result = A + ~B + 1 = A − B.

Where will the “Sub” signal come from?

<table>
<thead>
<tr>
<th>control</th>
<th>out=</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>undefined</td>
</tr>
<tr>
<td>1</td>
<td>undefined</td>
</tr>
<tr>
<td>2</td>
<td>A + B</td>
</tr>
<tr>
<td>3</td>
<td>A − B</td>
</tr>
</tbody>
</table>
Complete 1-bit ALU
Complete 1-bit Logic Unit

- What should the control inputs (R0, R1) connect to?
- How do we select between the adder and the logic unit?
- How do we control the selection?

<table>
<thead>
<tr>
<th>R1</th>
<th>R0</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$G_i = X_i Y_i$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$G_i = X_i + Y_i$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>$G_i = (X_i + Y_i)'$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$G_i = X_i \oplus Y_i$</td>
</tr>
</tbody>
</table>
Complete 1-bit ALU

1-bit ALU slice

A
B
carryin

control[0]
carryout

control[1]
carryout

control[2]
carryout

A
B
XOR

A
B
Full Adder
Sum
Cout

A
B
Logic Unit
out

R[0]
R[1]
Connecting 1-bit ALUs

$\rightarrow 32$-bit ALU

$A[31:0]$  $B[31:0]$  $\text{control}(2=0)$


$\text{cout}(31)$  $\text{cout}(2)$  $\text{cout}(1)$  $\text{cout}(0)$

$\text{carryOut}$
Flags (overflow, zero, **negative**)

- Let’s do negative first; negative evaluates to:
  - 1 when the output is negative, and
  - 0 when the output is positive or zero

- **Negative =**
  a) carryout [30]
  b) output [30]
  c) carryout [31]
  d) output [31]
  e) control [0]
Flags (overflow, zero, negative)

- zero evaluates to:
  - 1 when the output is equal to zero, else 0

- Zero =

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Zero?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>NOR</td>
</tr>
</tbody>
</table>

\[ \text{out}[31] \quad \text{out}[30] \quad \text{out}[29] \quad \text{out}[28] \quad \text{out}[27] \quad \text{out}[26] \quad \text{out}[25] \quad \text{out}[24] \quad \text{out}[23] \quad \text{out}[22] \quad \text{out}[21] \quad \text{out}[20] \quad \text{out}[19] \quad \text{out}[18] \quad \text{out}[17] \quad \text{out}[16] \quad \text{out}[15] \quad \text{out}[14] \quad \text{out}[13] \quad \text{out}[12] \quad \text{out}[11] \quad \text{out}[10] \quad \text{out}[9] \quad \text{out}[8] \quad \text{out}[7] \quad \text{out}[6] \quad \text{out}[5] \quad \text{out}[4] \quad \text{out}[3] \quad \text{out}[2] \quad \text{out}[1] \quad \text{out}[0] \]
Flags (overflow, zero, negative)

- Overflow (for 2’s complement) evaluates to:
  - 1 when the overflow occurred, else 0
  - adding two positive numbers yields a negative number
  - adding two negative numbers yields a positive number

- Consider the adder for the MSB:

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>C_in</th>
<th>C_out</th>
<th>S</th>
<th>Cin</th>
<th>Cout</th>
<th>overflow</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>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</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>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</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>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</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>1</td>
</tr>
</tbody>
</table>

- Overflow =

a) cin[31]  NOR cout[31]
b) cin[31]  AND cout[31]
c) cin[31]  OR cout[31]
d) cin[31]  XOR cout[31]
e) cin[31]  NAND cout[31]