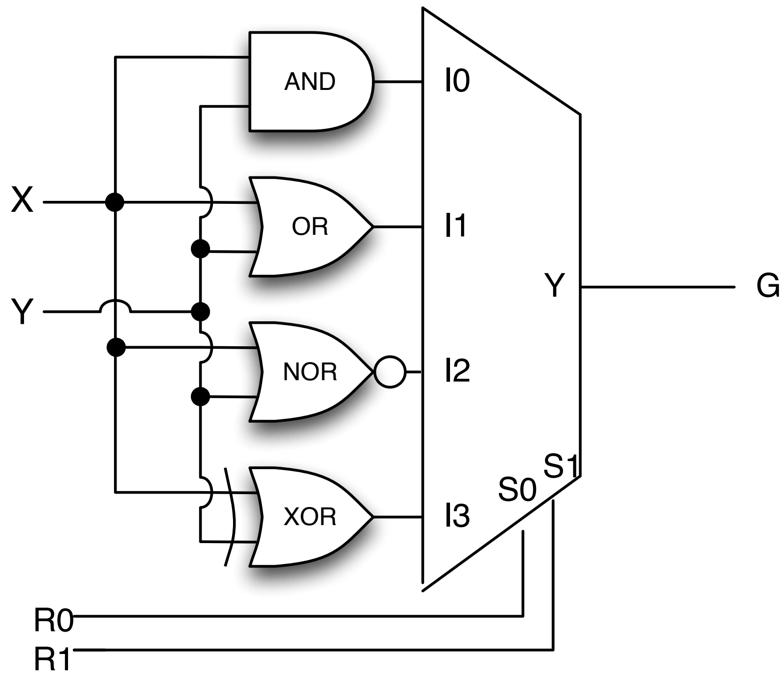
Building an ALU (Part 2):

Why NOR is more general than NOT:

- **We've changed the ALU design to include NOR over NOT**



x	NOT(x)
0	1
1	0

x	y	NOR(x,y)
0	0	1
0	1	0
1	0	0
1	1	0

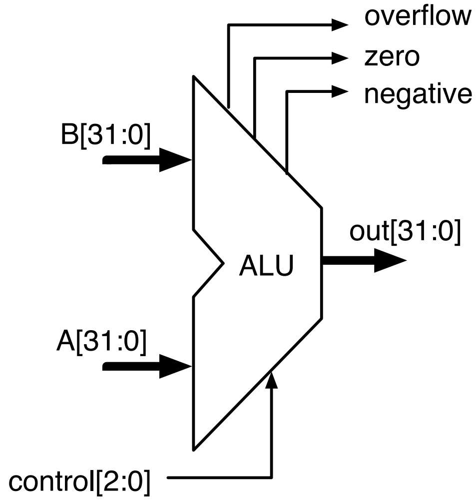
- **We can still implement NOT:**
  - NOT(X) is implemented by using NOR(X, Y) & setting Y=0

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

Building 32-bit ALU



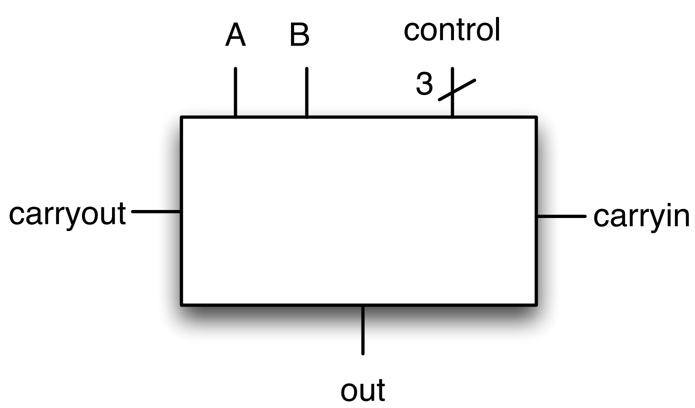
control	out =
0	undefined
1	undefined
2	A + B
3	A - B
4	A AND B
5	A OR B
6	A NOR B
7	A XOR B

```
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?

We want to create a 1-bit ALU

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

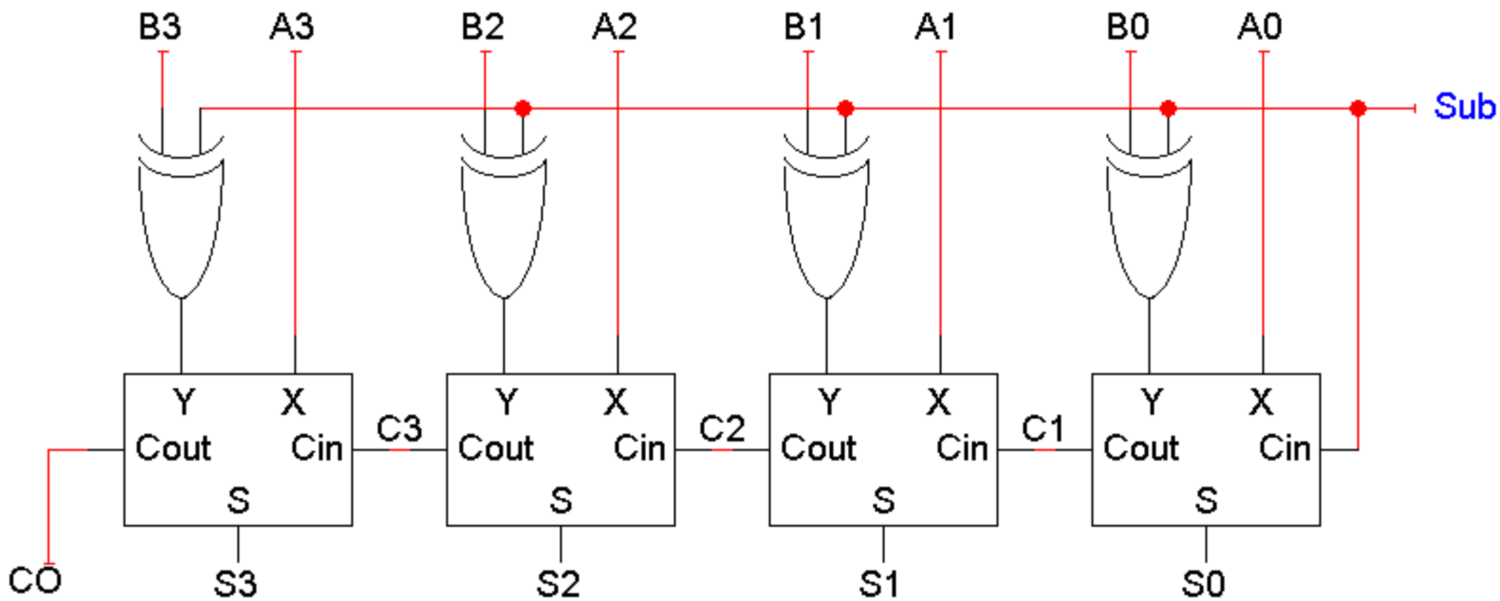


control	out_i =
0	undefined
1	undefined
2	$X_i + Y_i$
3	$X_i - Y_i$
4	X_i AND Y_i
5	X_i OR Y_i
6	X_i NOR Y_i
7	X_i XOR Y_i

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

Addition + Subtraction in one circuit

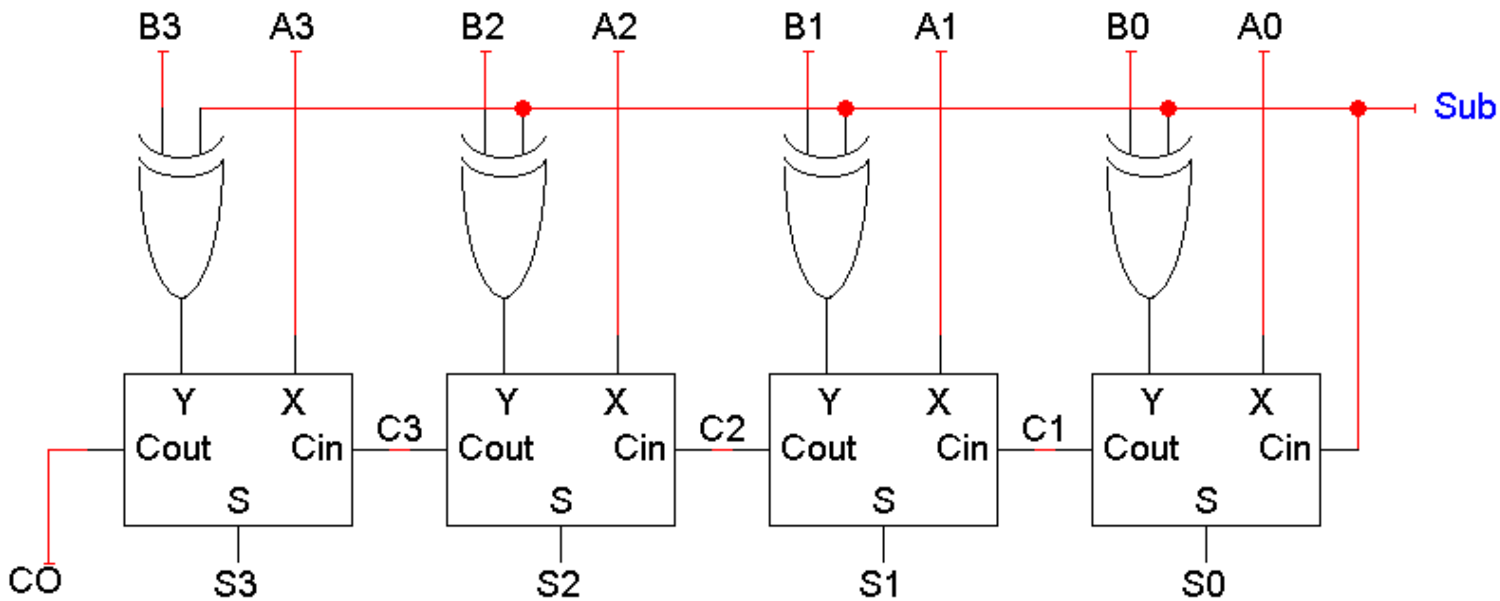
- When Sub = 0, Y = B and Cin = 0. Result = A + B + 0 = A + B.
- When Sub = 1, Y = ~B and Cin = 1. Result = A + ~B + 1 = A – B.



- **Which parts belong in inside the 1-bit ALU?**

Addition + Subtraction in one circuit

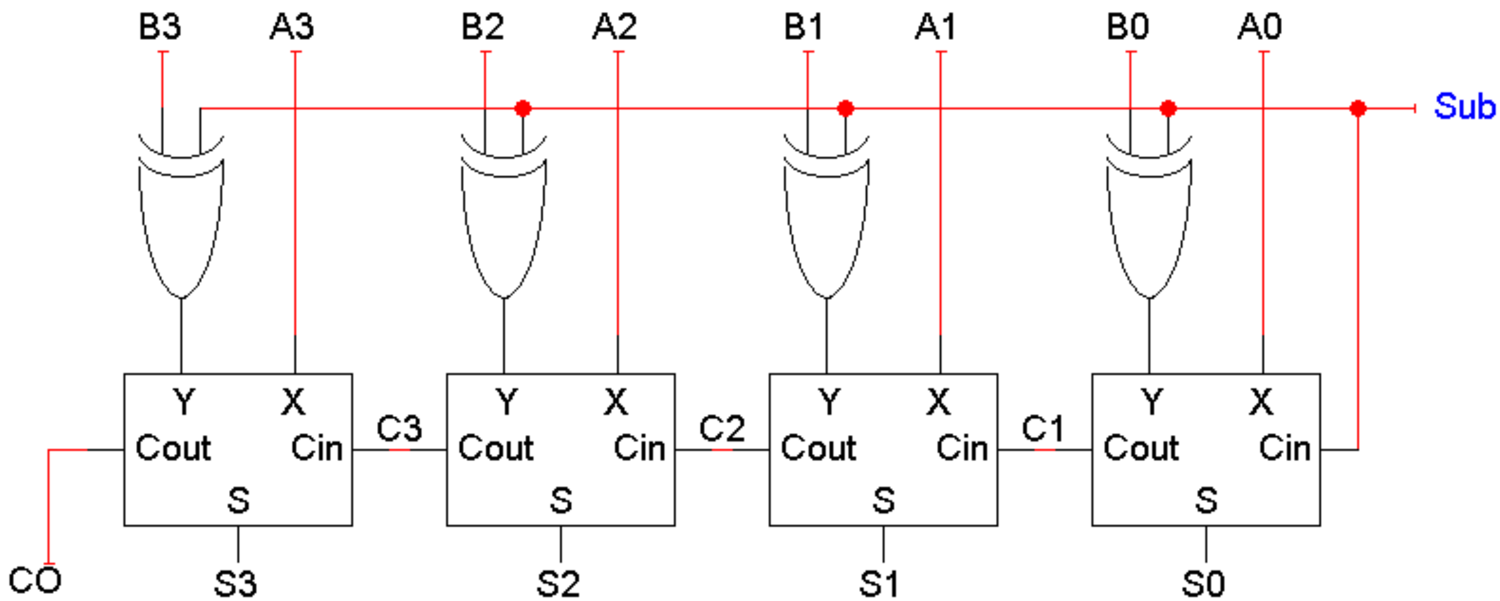
- When Sub = 0, Y = B and Cin = 0. Result = A + B + 0 = A + B.
- When Sub = 1, Y = ~B and Cin = 1. Result = A + ~B + 1 = A – B.



- **What should we do with the full adder's `Cin` input?**

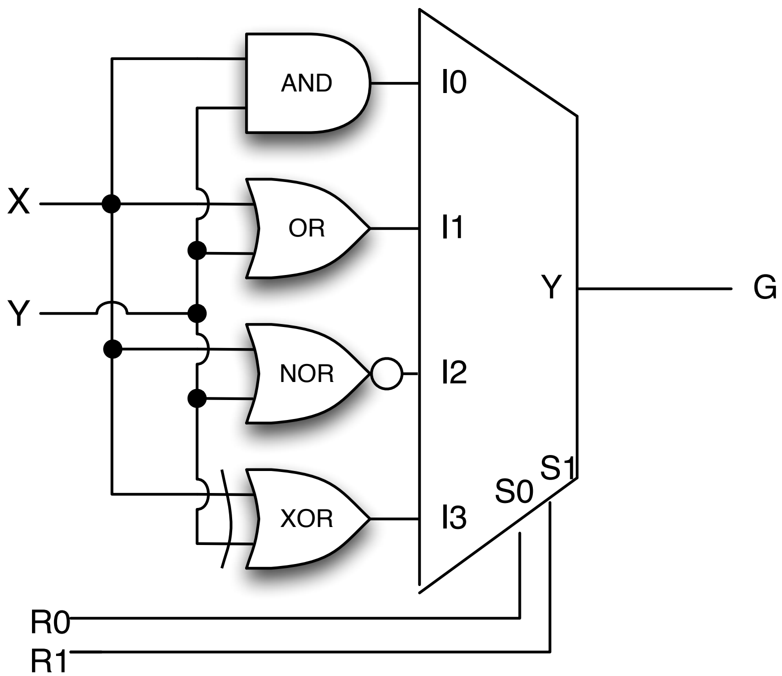
Addition + Subtraction in one circuit

- 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?**

Complete 1-bit Logic Unit

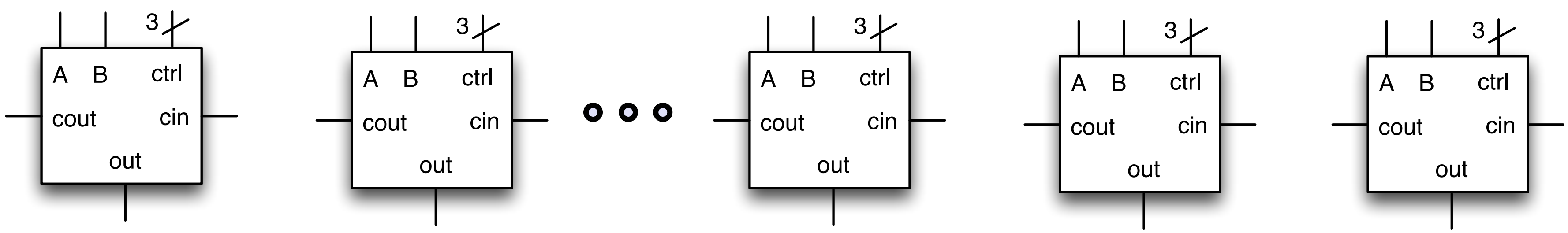


R_1	R_0	Output
0	0	$G_i = X_i Y_i$
0	1	$G_i = X_i + Y_i$
1	0	$G_i = (X_i + Y_i)'$
1	1	$G_i = X_i \oplus Y_i$

- 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?

Connecting 1-bit ALUs



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 =**

Flags (overflow, zero, negative)

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

- **Zero =**

Flags (overflow, zero, negative)

- **Overflow 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:**

X	Y	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

- **Overflow =**