Calculator Lab Overview

Note: Slides Updated 10/8/12

Design and implement a simple calculator:

- **Input**: 2, 4 bit 2's complement numbers
- **Output:** Signed Decimal Number
- **Operations**
  - Addition
  - Subtraction
  - Absolute Value

Restrictions

- You will learn how to implement arithmetic operations at a binary level, so:
  - **MAY NOT** use Verilog Arithmetic Operators **+** and **–**
    - **Solution would be Trivial!!**
  - May only use **1**, N-bit adder
    - Use Example from Lab 2
  - Top Level Design may be either Schematic or Verilog
  - May use continuous “assign” operators
    - AND **&**, OR **|**, XOR **^** , 1’s complement **~**, , “?” operator, etc.
  - May use any form of schematic entry

Review: Addition With Boolean Operators

Half-Adder

x	y	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

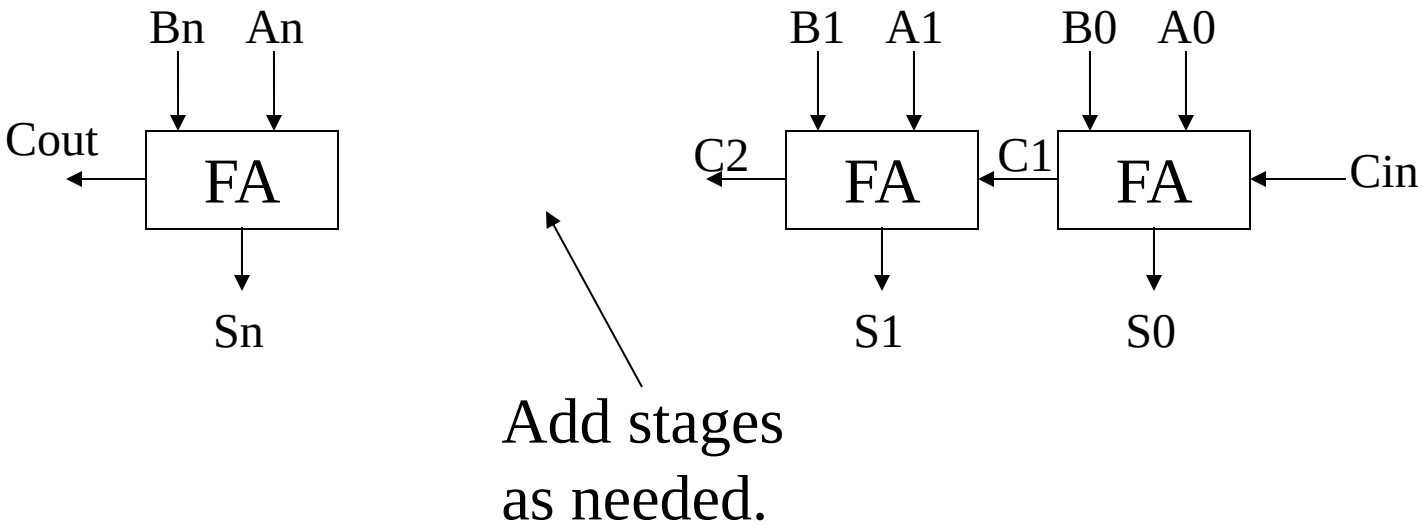
Sum = (~X & Y) | (X & ~Y) = X ^ Y

Carry = X&Y

Can be implemented with simple Boolean operations!!

Review: Adding Sets

- Half-Adders→Full Adders→N Bit Ripple Carry adders (Lab 2).



Subtraction with Boolean Operators

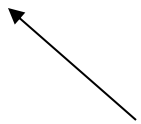
- Perform 2's complement of argument to be subtracted then add:

  For example, 3 – 1 → 3 + (-1) = 2

  0b011 3

  0b111 -1

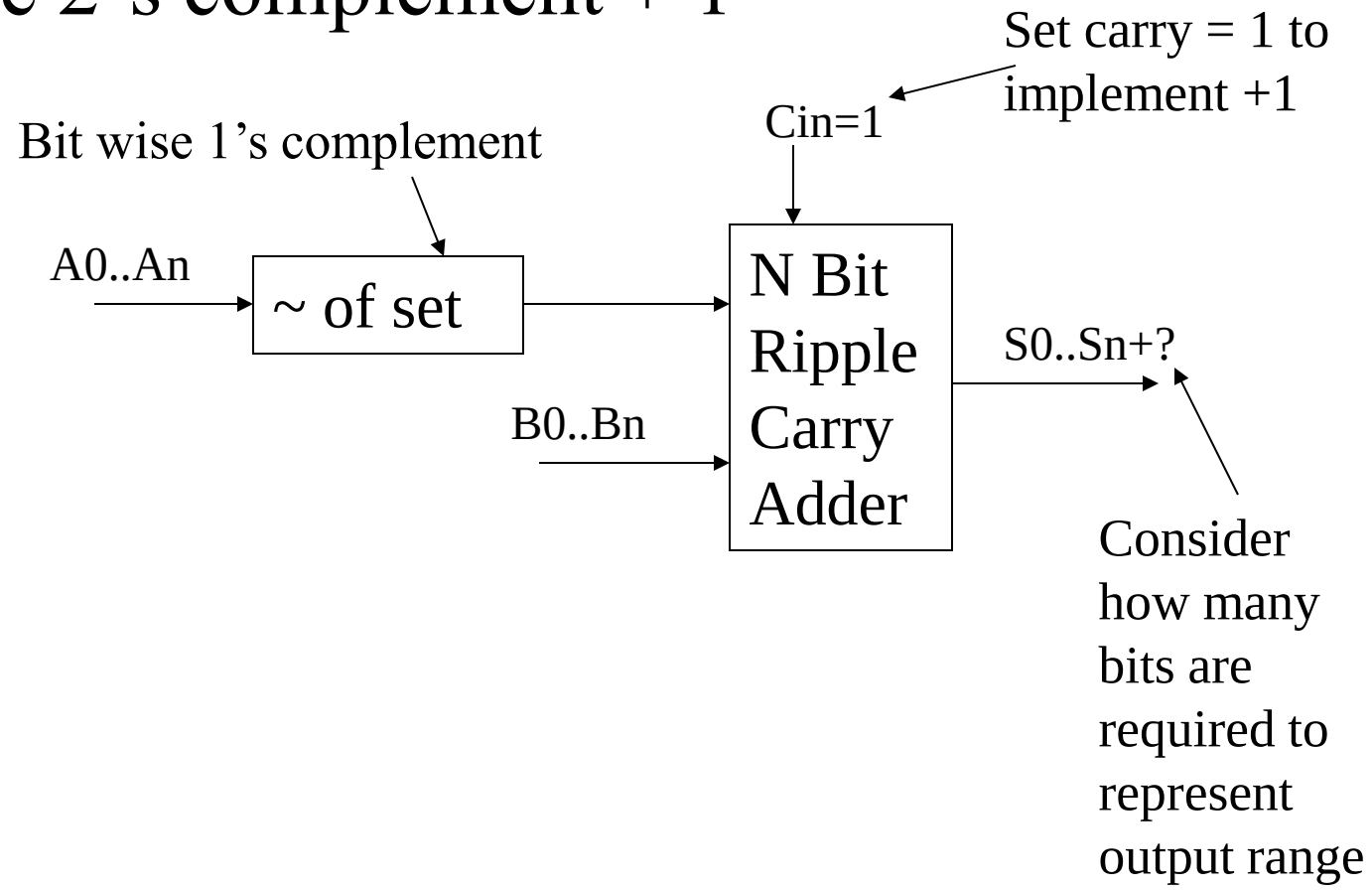
  0b1010 2 (3 bit)

  Carry to 4th bit position must be ignored.

Implementing 2's Complement

- Take 2's complement + 1



Absolute Value

Implement absolute value by testing for negative value and then subtracting.

For example,

if (arg = = neg) arg = 0 – arg

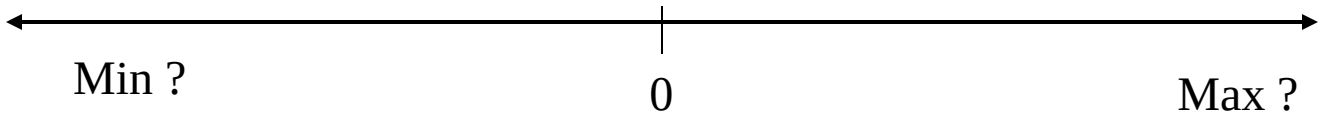
else arg = arg.

How can you tell arg is negative?

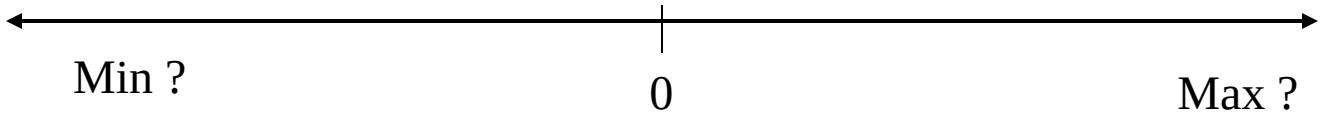
How many bits do you need?

Consider a simple, **3 bit** calculator.

What is the range of input values?



What is the range of output values?



How many bits are required to represent the output range?

Sign Extension

Recall the example used earlier: 3 – 1 = 2

0b011

0b111

0b1010

If 4 bits are required to represent the output, the result is not 2 but 0b1010 = -6!!!

This problem can be fixed by sign extending the input values to match the size of the output.

0b0011

0b1111

0b10010

The 4 bit result is now correct and the 5^{th} bit is ignored.

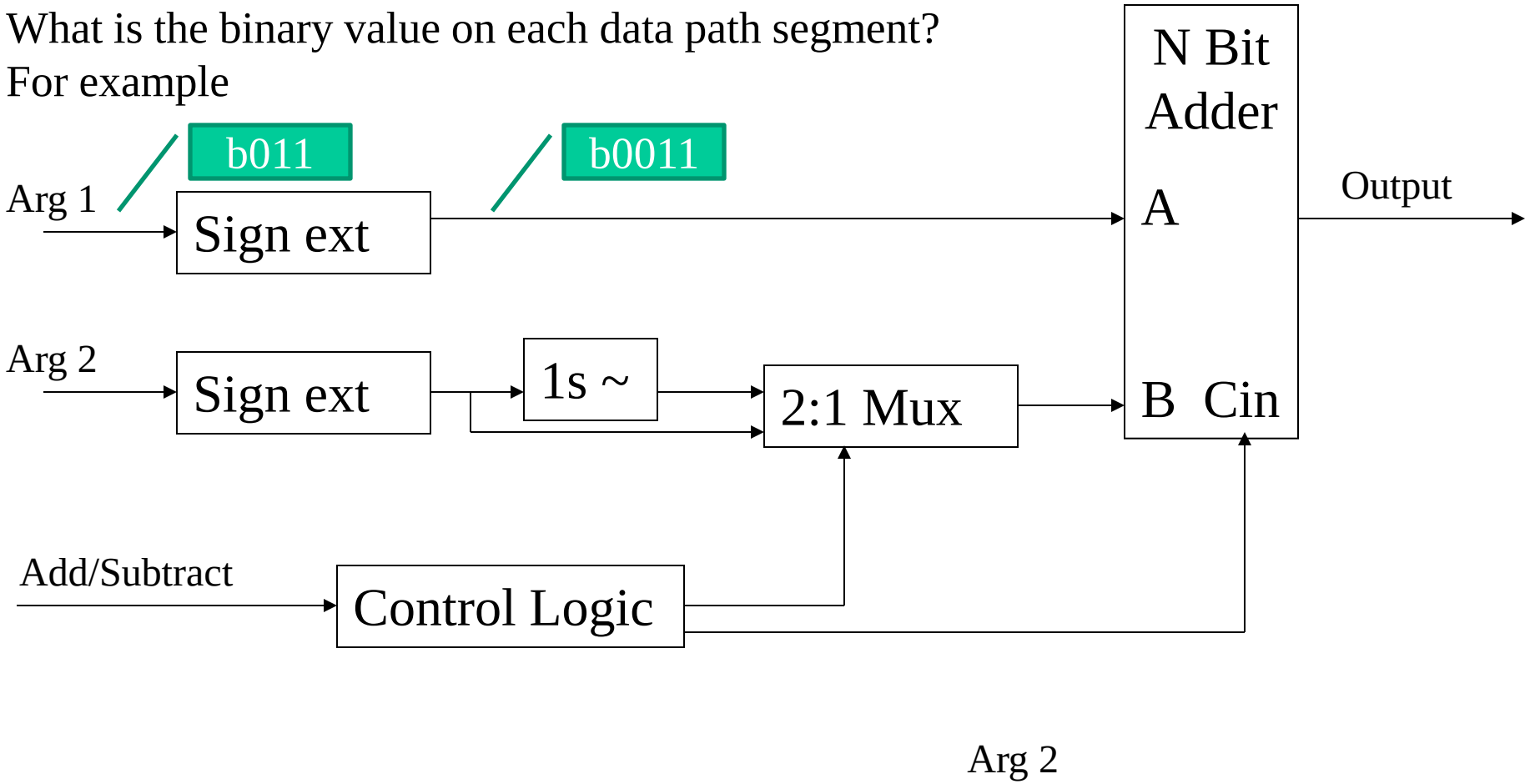
High Level Functional Organization to Implement Addition and Subtraction

Additional functionality will be required to implement absolute value

Consider 3 – 1 = 2

What is the binary value on each data path segment?

For example



Binary to Seven Segment Encoder

```
module Bto7seg(A, B, S);
        input [3:0] A; // 4 bit input
        output [6:0] B; // Seven segment output
        output [6:0] S; // Sign output
        wire [6:0] B;
        wire [6:0] S;
        `define n0 7'b1000000 // Each of these values will display the corresponding
        `define n1 7'b1111001 // digit on an active low 7 segment LED display
        `define n2 7'b0100100
        `define n3 you complete
        `define n4
        `define n5
        `define n6
        `define n7
        `define n8
        `define P 7'b1111111
        `define N 7'b1111110
```

Binary to Seven Segment Encoder, cont

```
assign B = (A == 4'b0000) ? `n0 : // Use the ? operator to construct a truth table
           (A == 4'b0001) ? `n1 : // For example, if A = 1, then B = n1 else
           (A == 4'b0010) ? `n2 : // if A = 2, then B = n2 else
           (A == 4'b0011) ? `n3 : // etc.
           (A == 4'b0100) ? `n4 :
           (A == 4'b0101) ? `n5 :
           (A == 4'b0110) ? `n6 :
           (A == 4'b0111) ? `n7 :
           (A == 4'b1111) ? `n1 :
           (A == 4'b1110) ? `n2 :
           (A == 4'b1101) ? `n3 :
           (A == 4'b1100) ? `n4 :
           (A == 4'b1011) ? `n5 :
           (A == 4'b1010) ? `n6 :
           (A == 4'b1001) ? `n7 : `n8
assign S = (A == 4'b0000) ? `P: // sign bit
           (A == 4'b0001) ? `P :
           (A == 4'b0010) ? `P :
           (A == 4'b0011) ? `P :
           (A == 4'b0100) ? `P :
           (A == 4'b0101) ? `P :
           (A == 4'b0110) ? `P :
           (A == 4'b0111) ? `P :
           (A == 4'b1111) ? `N :
           (A == 4'b1110) ? `N :
           (A == 4'b1101) ? `N :
           (A == 4'b1100) ? `N :
           (A == 4'b1011) ? `N :
           (A == 4'b1010) ? `N :
           (A == 4'b1001) ? `N : N
endmodule
```

A few more points

- Unlike Previous Pre-Lab Assignments, this Pre-Lab is a Partial Design Solution
- Need to Implement Absolute Value and Binary to 7-Segment Decimal Encoding for In-Lab
- Post-Lab Assignment Requires some Simulations