

Lab 2: Robertson's Multiplication and Error Correction

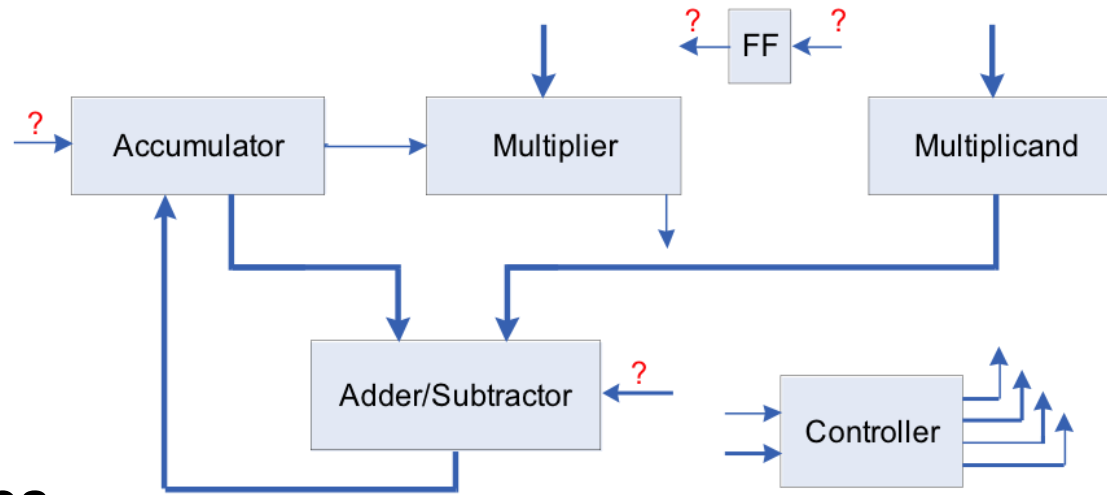
LogicWorks Implementation

- Part 1: Basic implementation with 25-bit Adder/Subtractor
- Part 2: Switch to 24-bit Adder/Subtractor with sign extending
- Part 3: Use the carry-out for the shift in value instead
- Part 4: Use the multiplicand's MSB for the shift in value
- Part 5: Fix issue with using multiplicand's MSB

Verilog Implementation

- Implement a Hamming Code detector and corrector in Verilog HDL.

2's Complement Robertson's Multiplication



Basic Idea

- Accumulate a partial sum in multiple steps.
- The rightmost bit of the multiplier is checked on every step to determine if the multiplicand should be added to the partial sum.
- Shift the partial sum and the multiplier one bit to the right on every step.
 - ✓ Make the partial sum line up with the multiplicand;
 - ✓ Always check the rightmost bit of multiplier, no need to check higher bits;
 - ✓ Lower bits of the partial sum shifted into the multiplier register for storage.
- Subtraction step at the end for negative multiplicands.

Multiplication Example

$$1010_2 * 1100_2$$

1010	multiplicand
<u>1100</u>	multiplier
0000000	
0000000	
11010	(add 1010, sign ext.)
<u>0110</u>	(sub 1010)
0011000	result

step1: accumulator multiplier multiplicand

 0000 → 1100 1010

 right shift

step2: 0000 → 0110 1010

 right shift

step3: 0000 → 0011 1010

accumulator = accumulator + multiplicand

1010 → 0011 1010

 right shift

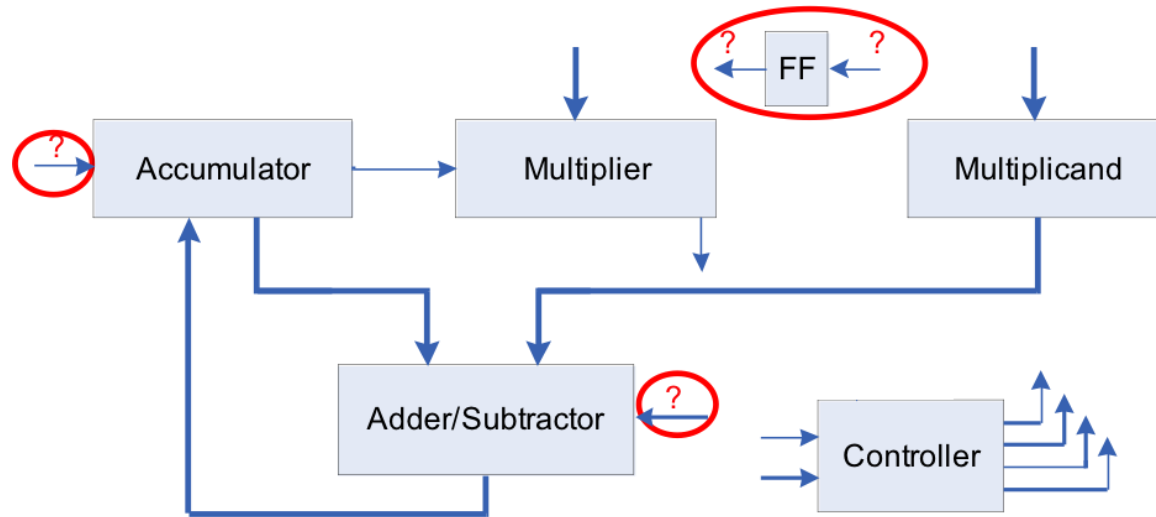
correction step: 1101 → 0001 1010

accumulator = accumulator - multiplicand

0011 → 0001 1010

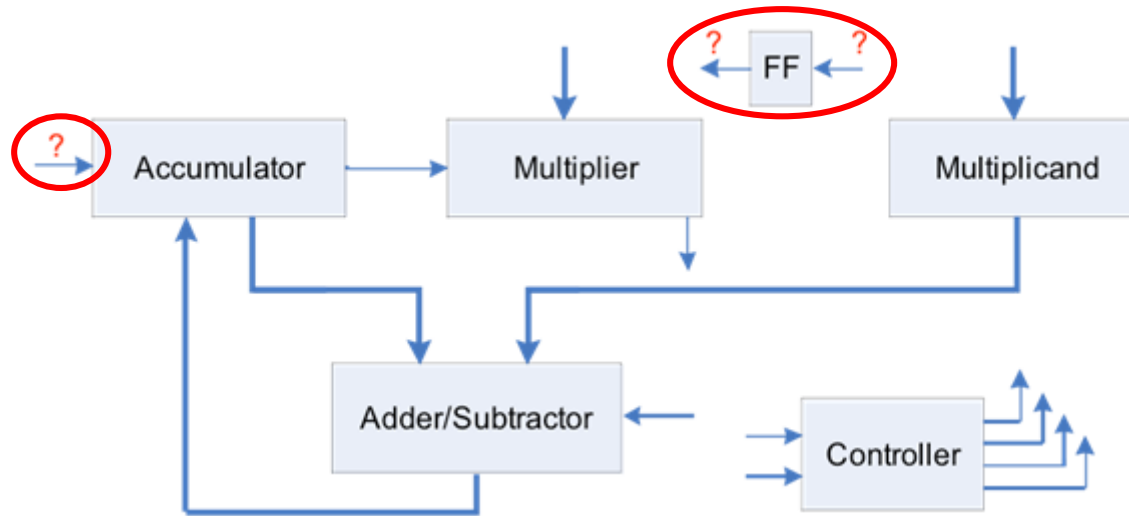
Result: 0011000

Overall Guidelines: Glue Logic



- The most significant bit of the accumulator during a shift
 - Different in each part of the lab
- The *A/S* signal for the adder/subtractor.
 - Add if $A/S = 0$; subtract if $A/S = 1$.

Overall Guidelines: Flip Flops



- Use **D Flip Flop** from LogicWorks library
- FF connections change in each part of the lab
 - Part 1 & 2: Sign extending
 - Part 3: Carry out of the partial sum
 - Part 4: MSB of the multiplicand
 - Part 5: Storing information about prior partial sums

LogicWorks Components

- **24-bit Shift Register:**
 - When LD is set, the parallel data input is loaded into the shift register;
 - When SH is set, the content of the register is shifted 1-bit right, with the most significant bit fed by S .
- **24-bit Adder-Subtractor:**
 - Output $Q(23:0) = A(23:0) \pm B(23:0)$
 - Output CO = Carry Out
- **25-bit Adder-Subtractor:**
 - Output $Q(24:0) = A(24:0) \pm B(24:0)$;
 - 24-bit inputs must be sign-extended to produce 25-bit inputs

LogicWorks Components

- **Control Unit:**
 - Handles the multi-cycle process by telling each device what to do on every cycle until the entire process is finished.
 - Every other device simply does what it's told on each cycle.
- **Your job: add necessary logic & connections.**

Verilog Implementation

- Given Modules:
 - HammingCode_tb.v – Fully specified, though you should add more test cases.
- Modules to Implement:
 - HammingCode.v – Your assignment is to fill this in.