Datapath Components

Module 7

UGTA - Corey Lancelotta

Key Takeaways

Logical Bit Shifting

Left & Right Shifts

Storage with Registers

Shift Registers & Parallel Load Registers

Manipulating Data

Arithmetic Operations & Comparators

Arithmetic Logic Unit

ALU Design & Function

BIT SHIFTING

SHIFT LEFT

1. Multiplying by base value (e.g., 2 for binary and 16 for hex) for each shift
2. LSB becomes 0
3. MSB is discarded

SHIFT RIGHT

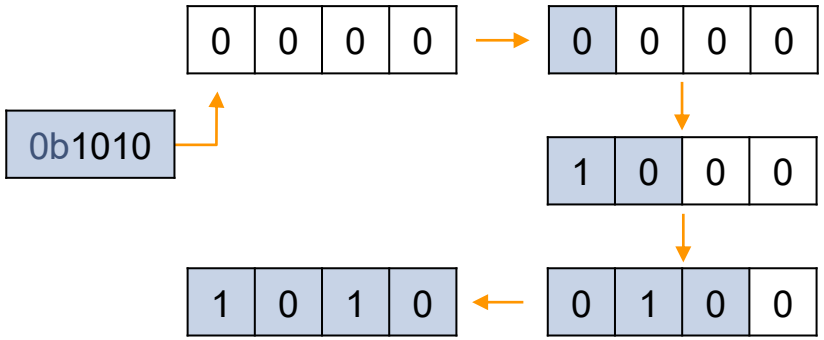
1. Dividing by base value (e.g., 2 for binary and 16 for hex) for each shift
2. MSB becomes 0
3. LSB is discarded

Important Note: Pay close attention to how many bits are being used to represent the number you are shifting. For example, decimal 10 as an 8-bit binary number has quite a few leading zeroes (0b00001010) These zeros are discarded when performing a bit shift to the left, not the "first" non-zero bit.

REGISTER TYPES

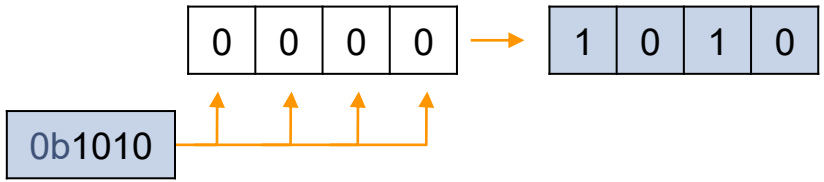
SHIFT REGISTER

1. Loads a single bit position from a single source
2. Values are shifted through the bit positions
3. Requires a serial signal (a stream of bits)



PARALLEL LOAD

1. Loads all bit positions at the same time
2. Each bit position has its own separate source
3. Loading the bit positions is synchronized by the clock



ADDERS

Full Adder

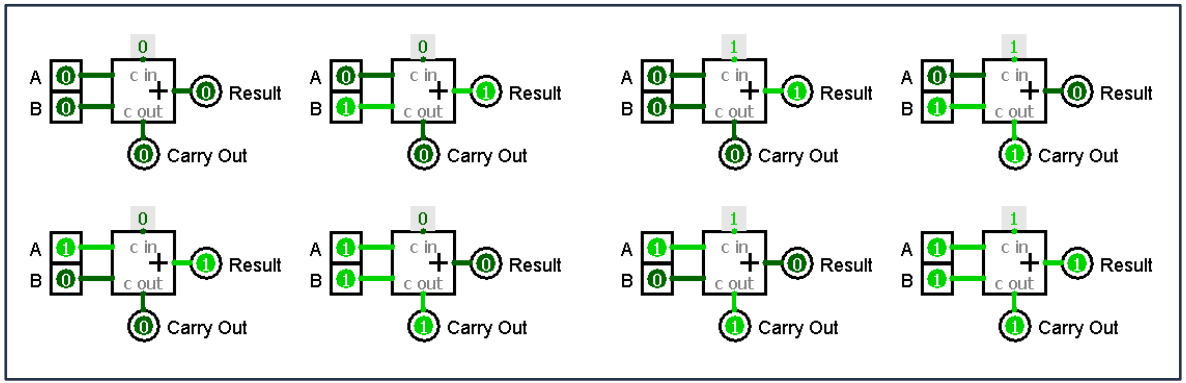
1. Two n-bit Number inputs
2. One 1-bit Carry input
3. Result output
4. Carry output

Half Adder

1. Two n-bit Number inputs
2. Result output
3. Carry output

SUBTRACTORS

1. Two n-bit Number inputs
2. One 1-bit Borrow input
3. Result output
4. Borrow output



Note: The Carry output takes a **value of 1**, when **Carry In + Input A + Input B exceeds the bit width of the adder** (in this case, ≥ 2)

MULTIPLIER & DIVIDER

1. Two n-bit Number inputs
2. One n-bit Carry input
3. Result output
4. Carry output

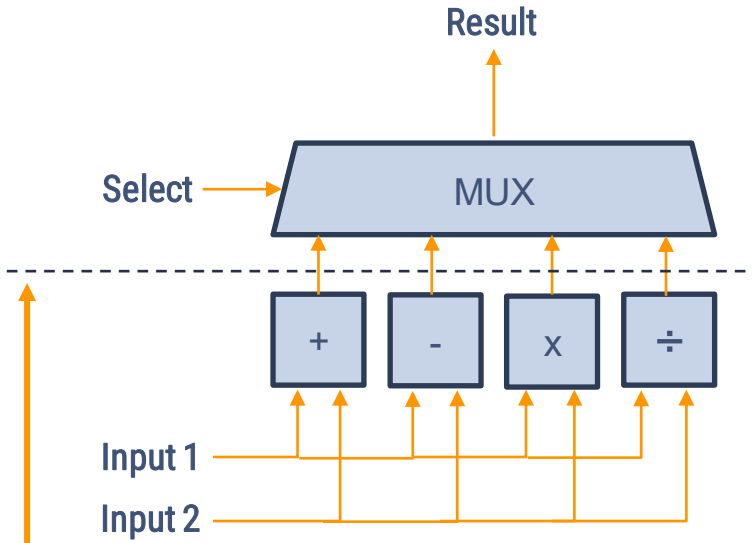
Note: Functions very similarly to the Adders and Subtractors

COMPARATORS

1. Two n-bit Number inputs
2. One 3-bit output (each bit is a Boolean result)
3. Each output bit represents a different comparison performed on the input numbers
   - One indicates if A < B
   - One indicates if A = B
   - One indicates if A < B

DETAILS

1. Combines multiple arithmetic operations into a single location
2. Allows for "operation selection"
3. Common designs use a Multiplexer to select which operation's result is passed out of the ALU
4. Any number of different operations can be added to the ALU
5. The ALUs considered in this course perform all operations, all the time. We simply select which result we want to use.

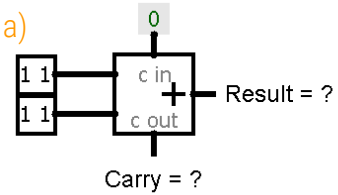


Note: The values at this point in the ALU are populated with the results of each arithmetic operations. We are simply deciding which result to pass out of the ALU.

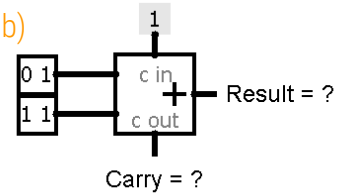
Practice Problems Questions

1. How many **bit shifts** (and in **which direction**) are needed to convert **decimal 25** to **0b11001000**?
2. How many **bit shifts** (and in **which direction**) are needed to convert **0b11101011** to a value less than **decimal 54**? What is the **binary** and **decimal result** of this operation?
3. Given the 10-bit number, **0b0101010111**, what is the result of **((0b0101010111 >> 2) << 3) >> 4** in **decimal** and **binary**?
4. What value does the **Carry** & **Result Outputs** have in the following **adder operations**?

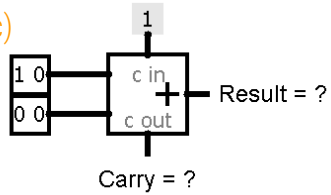
a)



b)

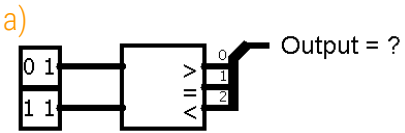


c)

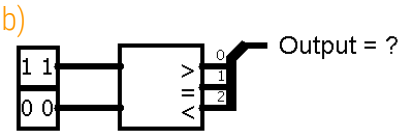


5. What is the **Output** of the following **comparator operations**?

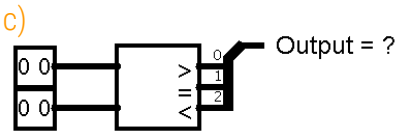
a)



b)



c)



Practice Problems Answers

1. The number 0b11001000 is decimal 200. Since 25 is less than 200, we want to shift left (multiplying by 2). **Shifting left 3 times** will yield 200.

2. The number 0b11101011 is decimal 235. We want a smaller number, so we start shifting to the right (dividing by 2). **Shifting right 3 times** will yield the number, **0b11101** = **decimal 29**.

3. 0b0101010111 >> 2 = 0b0001010101
   0b0001010101 << 3 = 0b1010101000
   0b1010101000 >> 4 = **0b0000101010 = decimal 42**.

4. a) carry = **1**, result = **10**
   b) carry = **1**, result = **01**
   c) carry = **0**, result = **11**

5. a) output = **001**
   b) output = **100**
   c) output = **010**