Logic Design (Part 3)
Combinational Logic Devices (Chapter 3 + Notes)

1

Digital Logic Circuits

- we can build the basic logic gates using transistors
- Can build any boolean function using these gates
  - Theory underlying design of Boolean functions ..Boolean Algebra
    - Optimize circuit using Karnaugh maps
- Power of abstraction….To build boolean functions, you can work with basic gates – no need to go down to the transistor level !!
- Use these gates as building blocks to build more complex combinational circuits
  - Combinational Logic Devices: Adder, Multiplier, Multiplexer, Decoder, …..
  - …any boolean function

2

Definition: Combinational and Sequential Logic Circuits

- A circuit is a collection of devices that are physically connected by wires
  - Combinational circuit
  - Sequential circuit
- In Combinational circuit the input determines output
- In sequential circuit, the input and the previous 'state' (previous values) determine output and next 'state'
  - Need to 'remember' previous value – need memory device
  - Need circuit to implement concept of storage

3

Recall our Goal….

- Design a machine that translates from natural language to electrons running around to solve the problem
  - We now have a device that controls how electrons run around
- Next: we want to build a computer
  - First step: Design a collection of logic devices that implement important functions that will be needed to build our computer
- S/W Analogy: When you write your software, you are using a collection of concepts, tools, IDEs and libraries
  - Each has been built, and tested, for you
  - All you have to do is combine them!

4

Combinational Logic Devices

- We saw how we can build the simple logic gates using transistors and build any boolean function using these gates
- Use these gates as building blocks to build more complex combinational circuits
  - Decoder: based on value of n-bit input control signal, select one of 2^N outputs
  - Multiplexer: based on value of N-bit input control signal, select one of 2^N inputs.
  - Adder: add two binary numbers
  - …any boolean function
- SW Analogy: We are building a library of functions
  - To design your solution, you can use any device in the library!

5

Three Devices we focus on…

- N-bit Adder
  - Can build Subtract using Adder
- Decoder
  - Decode a bit string
- Multiplexer
  - A channel selector
- Other useful combinational logic devices
  - Multipliers
  - Shifters (but may need storage)
  - Comparators (to compare two numbers)
  - …

6

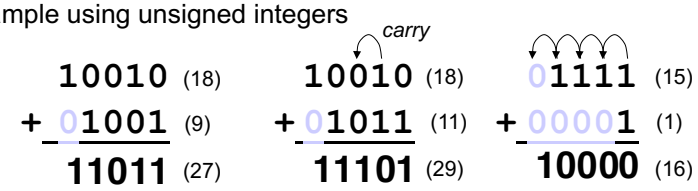
1. N-bit Adder

- Add two N-bit numbers, represented in 2's complement
- Algorithm (for now): add corresponding bit positions, starting with least significant position, and propagate the carry bit leftward.
  - In practice: there are faster algorithms
  - Big-Oh Analysis:
    - To add N bit numbers how 'far' will the carry propagate ?

7

Binary Addition

- Binary addition – just like base 10 (decimal) !
  - Add from right to left, propagating carry
  - Example using unsigned integers



Key Observation: We add one bit at a time
therefore, building block is a 1-bit adder
Use 1-bit adder to build N-bit adder!

8

1-bit Adder

- Two inputs A, B and Two outputs: S (sum) and Carry out (C)
- Truth table:

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- Problem?
- This works only for bit 0 where there is no Carry-in
  - Called a half adder
- In general, we can have a carry-in input, so 3 inputs are A,B,C_{in} (carry-in) and 2 outputs S, C_{out} (carry out)

9

Truth Table for Full-Adder

A	B	Carry In	Out	Carry Out
0	**0**	**0**	**0**	**0**
0	**0**	**1**	1	**0**
0	**1**	**0**	1	**0**
0	**1**	**1**	**0**	1
1	**0**	**0**	1	**0**
1	**0**	**1**	**0**	1
1	**1**	**0**	**0**	1
1	**1**	**1**	1	1

10

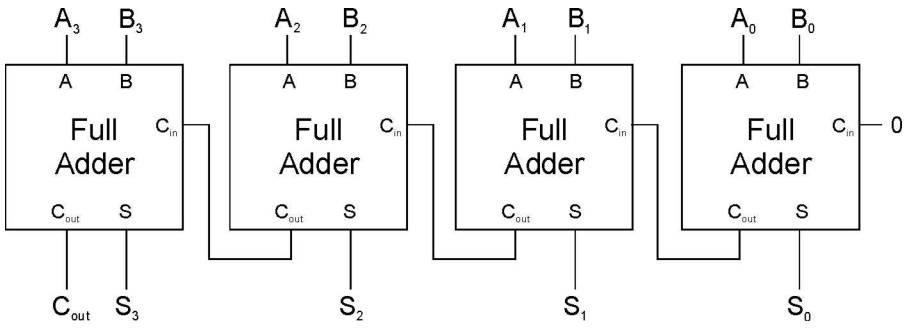
Truth Table for Full-Adder

A	B	Carry In	Out	Carry Out
0	**0**	**0**	**0**	**0**
0	**0**	**1**	1	**0**
0	**1**	**0**	1	**0**
0	**1**	**1**	**0**	1
1	**0**	**0**	1	**0**
1	**0**	**1**	**0**	1
1	**1**	**0**	**0**	1
1	**1**	**1**	1	1

11

N-bit Adder

- Use the building block of the full-adder to build N-bit adder
  - Need to connect carry-out to carry-in of next significant bit
- Example of 4-bit adder using 4 1-bit adders chained together (ripple carry adder)



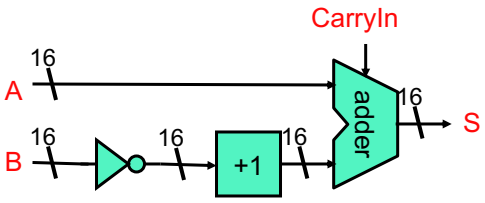
12

How about a "subtractor?"

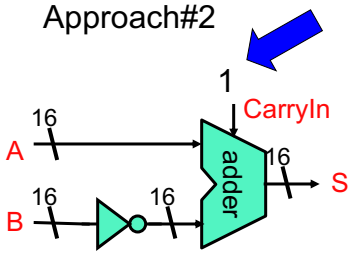
- Build a subtracter from out multi-bit adder
  - Calculate A – B = A + –B
  - Negate B
  - Recall –B = NOT(B) + 1

We "carry in" a 1
(no longer need incrementer)

Approach#1



Approach#2



Question: Can we build a single circuit that can do either (add or subtract)?

13

The Decoder

- Useful for recognizing a particular bit pattern of 0's and 1's
- Connection to Computer Organization:
  - Program consists of instructions -- coded in binary (0's and 1's)
  - We want to look at a bit string for the instruction and determine what the instruction is
    - Is it an ADD or a MULT or a GOTO or…..
    - each instruction is given a unique encoding & decoder looks at the encoding and determines which ONE of the instructions the code corresponds to (i.e, which instruction has to be executed)
- In S/W, a "case"/switch statement:
  - One of the cases will be evaluated depending on value of 'input'

14

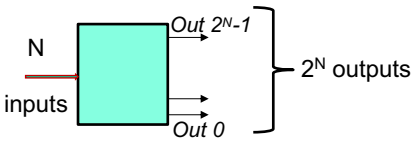
Switch (case) statement in C

```
int x;
….
switch(x) {
        case 0:        /* if x=0 call func Kevin */
                Kevin(); /* ex: Kevin does add */
                break;
        case 1:        /* if x=1 call func Graham */
                Graham();
                break;
        case 2:        /* if x=2 call func Sarah */
                Sarah(); /* ex: Sarah does AND */
                break;
        case 3:        /* if x=3 call func Linnea */
                Linnea();
                break;
        default: printf(”invalid value of x”\n);
                break;
}
..
```

15

N-2^N Decoder

- N inputs – these represent the binary encoding of the 2^N Outputs
  - Ex: if N=2, then 4 outputs 0,1,2,3, encoded to be ‘switched on’ when inputs are one of 00, 01, 10, 11 respectively
- Schematic:



16

Designing a Decoder: Truth table
4 output lines x0,x1,x2,x3 & 2 inputs a1,a0

From truth table, design circuit:

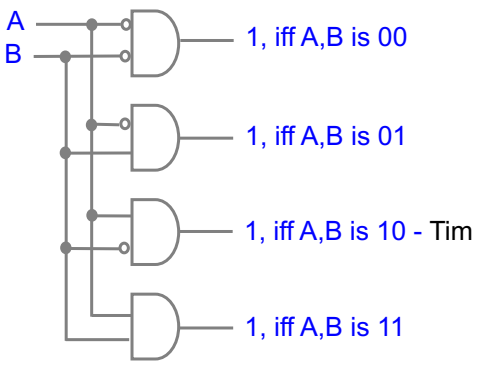
$x_0 = a_1'.a_0'$ *(i.e., (NOT a_1) AND (NOT a_0))*

$x_1 = a_1'.a_0$ $x_2 = a_1.a_0'$ $x_3 = a_1.a_0$

a_1	a_0	x_0	x_1	x_2	x_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

17

Decoder



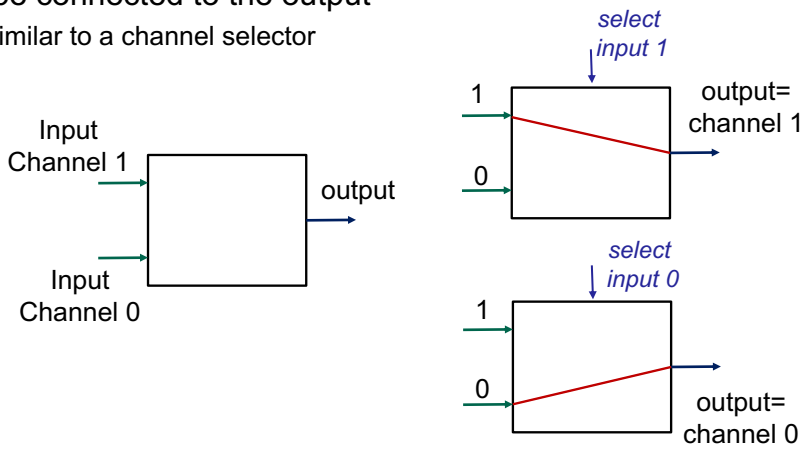
2-bit decoder
(4 input decoder)

- An n input decoder has 2^n outputs.
- $Output_i$ is 1 iff the binary value of the n-bit input is i.
- At any time, exactly one output is 1, all others are 0.

18

The Multiplexer - selector

- Multiplexer (MUX) is a device that selects one of the inputs to be connected to the output
  - Similar to a channel selector

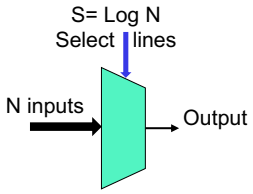


This is a 2-1 Multiplexer: Selects one of 2 inputs as the output

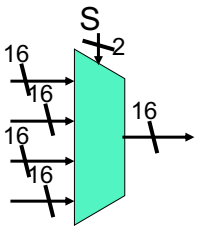
19

N-1 Multiplexer

- Multiplexer selects one of the N inputs as the output
  - It needs $\log_2 N$ 'select lines' to determine which of the N inputs is selected to appear at the output
  - Schematic of a MUX:



- Multi-bit muxes
  - Can switch an entire "bus" or group of signals
  - Switch n-bits with *n* muxes with the same select bits

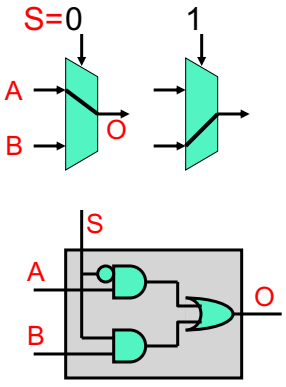


20

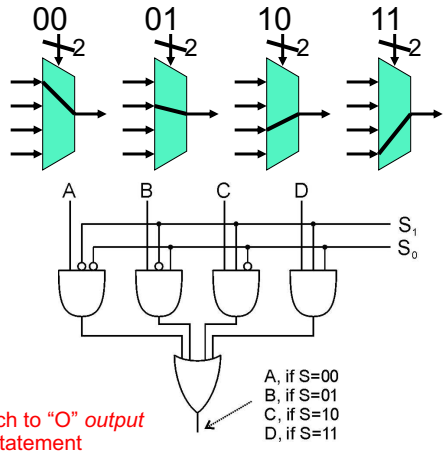
The Multiplexer (MUX) – 2-1 and 4-1 MUX

- Selector/Chooser of signals – *Imagine Switching Railroad Tracks*
  - Multi-way switch

2-to-1 Mux



4-to-1 Mux

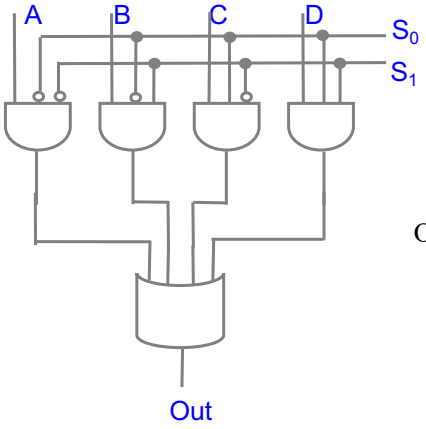


Input "S" *selects* A or B to attach to "O" *output*
Acts like an "IF/ELSE" statement

21

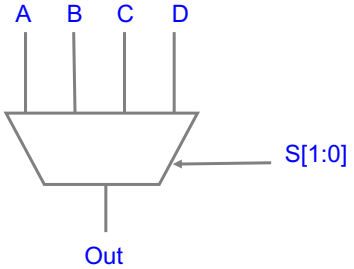
Designing 4-1 MUX Using Logic Gates

Multiplexer (MUX)



- In general, a MUX has
  - 2^n data inputs
  - n select (or control) lines
  - and 1 output.
- It behaves like a channel selector.

$$\text{Out} = A.\overline{S_0}.\overline{S_1} + B.\overline{S_0}.S_1 + C.S_0.\overline{S_1} + D.S_0.S_1$$

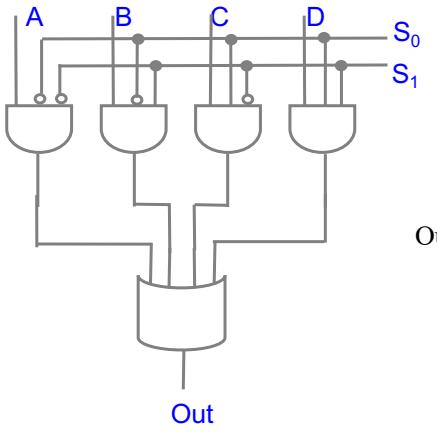


A 4-to-1 MUX:
Out takes the value of A,B, C or D
depending on the value of S (00, 01, 10, 11)
Out = A if S=00 Out = B if S = 01
Out = C if S=10 Out=D if S=11

22

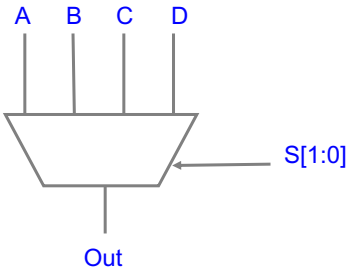
Designing 4-1 MUX Using Logic Gates

Multiplexer (MUX)



- In general, a MUX has
  - 2^n data inputs
  - n select (or control) lines
  - and 1 output.
- It behaves like a channel selector.

$$\text{Out} = A.\overline{S}_0.\overline{S}_1 + B.\overline{S}_0.S_1 + C.S_0.\overline{S}_1 + D.S_0.S_1$$

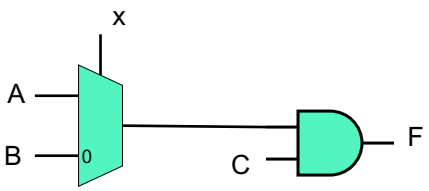


A 4-to-1 MUX:
Out takes the value of A,B, C or D depending on the value of S (00, 01, 10, 11)
Out = A if S=00 Out = B if S = 01
Out = C if S=10 Out=D if S=11

23

Example: MUX in a circuit

- Inputs A,,B,C and x (select signal); Output F
- Devices/Gates: 2-1 MUX, AND gate



- If x=0, output of MUX = B and F= B.C
- If x=1, output of MUX = A and F = A.C
- Can write F = xAC + x'BC

24

Combinational vs. Sequential

- Combinational Circuit
  - always gives the same output for a given set of inputs
    - ex: adder always generates sum and carry, regardless of previous inputs
- Sequential Circuit
  - stores information
  - output depends on stored information (state) plus input
    - so a given input might produce different outputs, depending on the stored information
  - *example: vending machine*
    - Current total increases when you insert coins
    - output depends on previous state
  - useful for building "memory" elements and "state machines"

25

Next . . Circuits with "memory"

- First we need to build a device that can store a bit
  - Using our current 'library' of gates
  - Building memory follows
- How to model sequential circuits/machines
  - Methodology for designing these machines: Finite state machine
  - Model as a directed graph
- How to we "synchronize" and "coordinate" the different pieces in the circuit….enter the CLOCK
- can we use a sequential circuit to "control" how computations take place in a processor ?
- Is a sequential circuit = Computer ?
  - Limitations of sequential machines..more in Foundations course

26

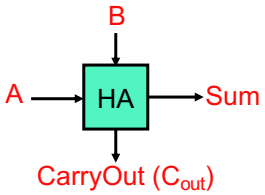
Appendix

27

Binary Arithmetic: Half Adder

- Logical Function: Half Adder, implement Carry Out:

A	B	Sum	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Half Adder's Logic Function:

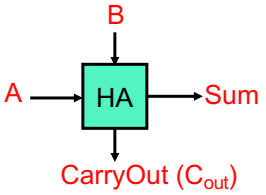
SUM=((A' AND B') OR (A AND B))'
C=((A' AND B') OR (A' AND B) OR (A AND B'))'

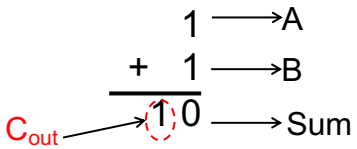
Realize though: C= (A AND B), this isn't always best way!

28

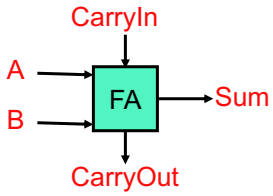
Addition: Full Adders

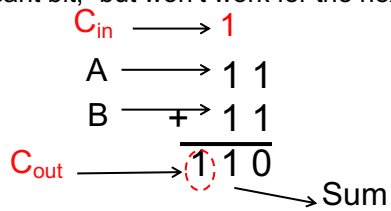
- There is a limit with the half adder
  - It can't implement multiple-bit addition





  - It works for "least significant bit," but won't work for the next





  - We need an adder that has 3 inputs and 2 outputs

3-29

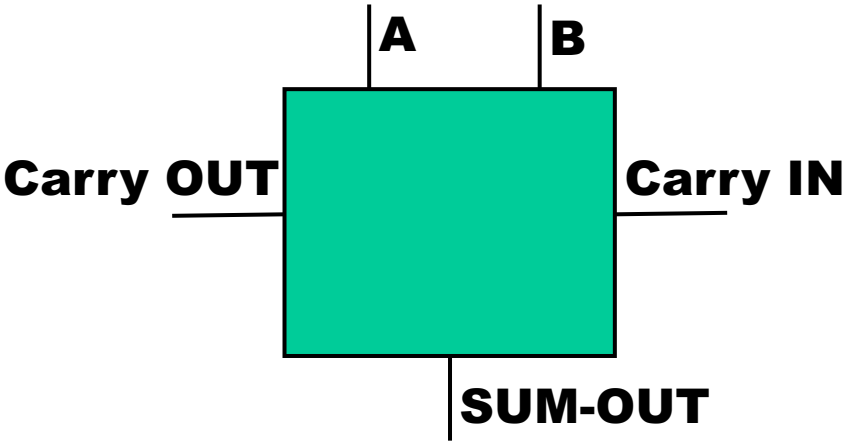
29

Truth Table

A	B	Carry In	Out	Carry Out
0	**0**	**0**	**0**	**0**
0	**0**	**1**	1	**0**
0	**1**	**0**	1	**0**
0	**1**	**1**	**0**	1
1	**0**	**0**	1	**0**
1	**0**	**1**	**0**	1
1	**1**	**0**	**0**	1
1	**1**	**1**	1	1

30

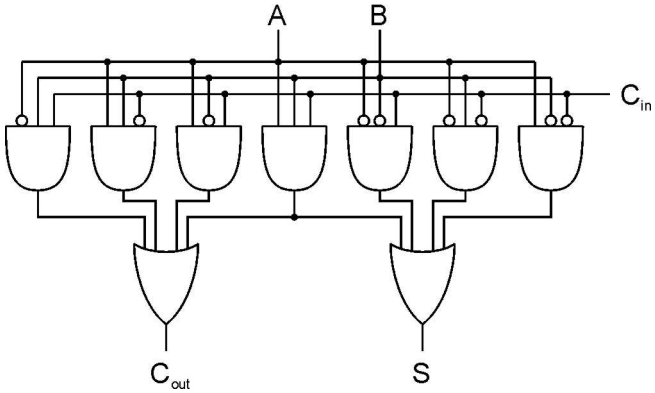
Full Adder



31

1-bit Full Adder

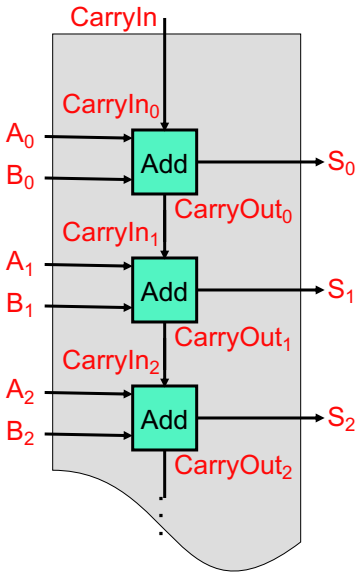
- Add two bits and carry-in, produce one-bit sum and carry-out.

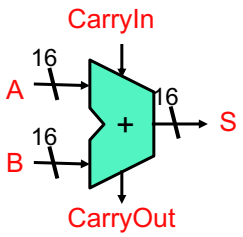


A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

32

N-bit Adder





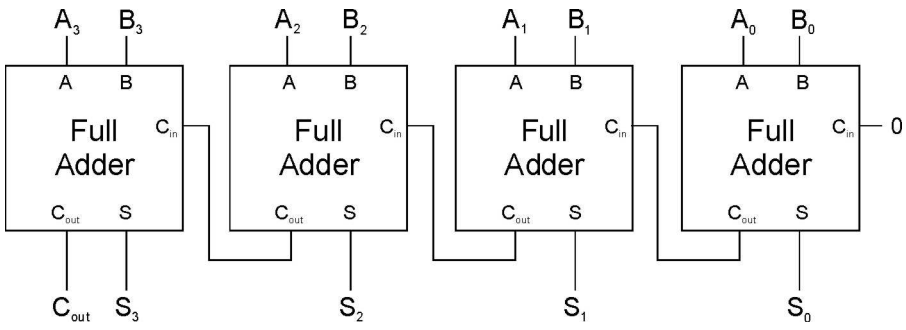
CarryOut: useful for detecting overflow

CarryIn: assumed to be zero if not present

3-33

33

Four-bit Adder



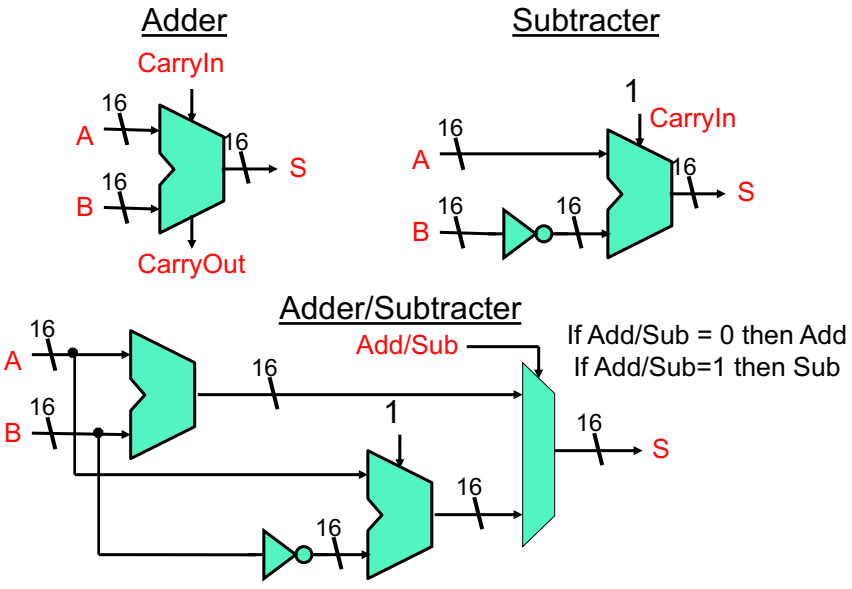
34

A multi function Arithmetic Unit

- In a CPU, we'd like to do BOTH addition and subtraction
  - Can we give the CPU the ability to choose between two pieces of hardware?
  - Yes!
- Using a MUX to build a multifunction ALU

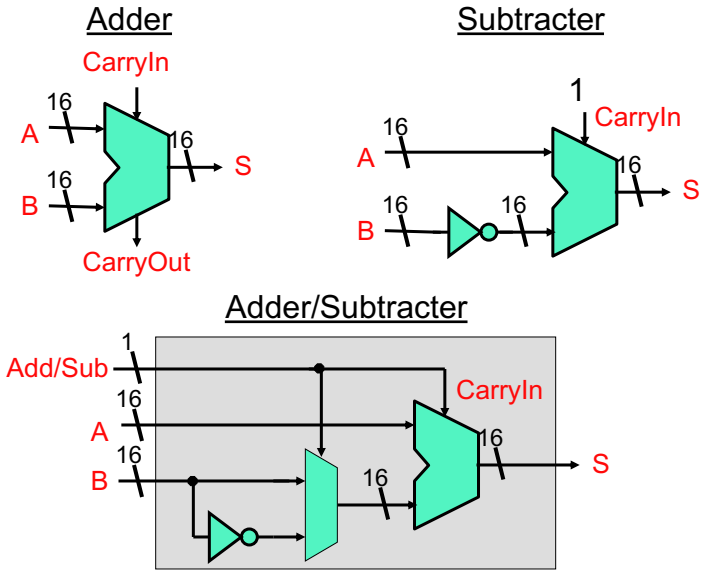
35

Building an ALU using MUX Adder/Subtracter - Approach #1



36

Adder/Subtracter - Approach #2 (Optimize HW)



37

Analysis of Circuits

- Download Example Circuits
  - In Cedar Logic: Set2.cdl
  - In Logisim: Set2.zip
    - Contains multiple files, titled Set2-Page1, Set2-Page2, etc.
    - Pages correspond to pages in the Set2.cdl file

38

Analysis of circuits in file: Set2

- Page 1: Check truth table and identify behavior
- Page 2: Part of the circuit looks identical to Page1, but there is an additional 'output.
  - Check truth table and identify behavior
- Page 3: This uses the circuit from Page2 and connects them (connects output of one circuit to input of next circuit).
  - Identify the function being implemented
- Page 4: If we treat 1-bit Adders as a Combinational logic device, then we can construct any N-bit adder using these devices
  - Bottom of the page has a schematic for a 4-bit adder
  - Input uses a Hex keyboard to input a 4-bit number
  - Uses a 7 segment LED display to display the 4-bit output as a hex number.
  - Analogy with software design: Construct solution which utilizes functions

39

Analysis of circuits in file: Set2

- Page 5: Uses an adder, but adds some logic operations to the input X before it is sent to the Adder: Therefore adding Y and f(X) to get output Z
  - Inputs use Hex keyboards to send in 4-bit numbers
  - Determine what is f(x) and then determine what function is being implemented.
- Page 6: Examples of Multiplexer and Decoder
- Page 7: Circuit uses a multiplexer and gates
  - As you set the different select lines to the multiplexer, determine what the output is
  - Determine the behavior of the circuit and its 'function'
- Page 8: Circuit uses Multiplexer, Adder
  - Check truth table and identify behavior
- Page 9: Uses Decoder and Multiplexer
  - Determine values of select lines of multiplexer based on output of decoder
- Page 10: Example using a comparator

40