Basics of Logic Design
Arithmetic Logic Unit (ALU)

Computer Science 104

Today's Lecture

- Homework #3 Assigned Due Mar 4
- Project: form groups of 3 (preferred) by Mar 6, I will assign after that
- Project Specifications on Web, Due April 17 (written document and demonstration)
- Building the building blocks…

Outline

- Review
- Digital building blocks
- An Arithmetic Logic Unit (ALU)

Reading

Appendix C, Chapter 3

Review: Digital Design

- Logic Design, Switching Circuits, Digital Logic

Recall: Everything is built from transistors

- A transistor is a switch
- It is either on or off
- On or off can represent True or False

Given a bunch of bits (0 or 1)…

- Is this instruction a lw or a beq?
- What register do I read?
- How do I add two numbers?
- Need a method to reason about complex expressions

Review: Boolean Functions

- Boolean functions have arguments that take two values ({T,F} or {0,1}) and they return a single or a set of ({T,F} or {0,1}) value(s).
- Boolean functions can always be represented by a table called a "Truth Table"
- Example: F: {0,1}3 -> {0,1}2

a	b	c	f_1	f_2
0	0	0	0	1
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	0
1	1	0	0	1
1	1	1	1	1

Review: Boolean Functions and Expressions

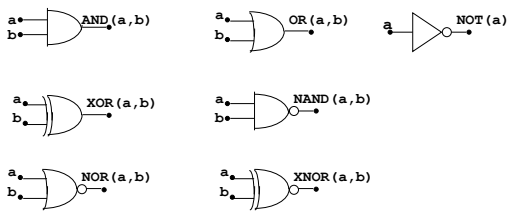
F(A, B, C) = (A * B) + (~A * C)

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Review: Boolean Gates

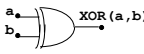
- Gates are electronics devices that implement simple Boolean functions

Examples



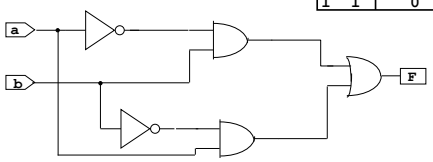
Review: Boolean Functions, Gates and Circuits

- Circuits are made from a network of gates. (function compositions).



F = ~a*b + ~b*a

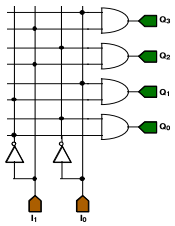
a	b	XOR(a,b)
0	0	0
0	1	1
1	0	1
1	1	0



Parity Example

- The parity code of a binary word counts the number of ones in a word. If there are an even number of ones the parity code is 0, if there are an odd number of ones the parity code is 1. For example, the parity of 0101 is 0, and the parity of 1101 is 1.
- Construct the truth table for a function that computes the parity of a four-bit word. Implement this function using AND, OR and NOT gates. (Note there are no constraints on the number of gate inputs.)

Circuit Example: Decoder

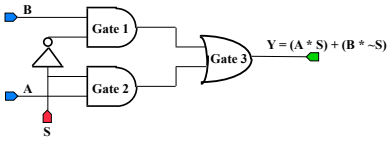


I_1 I_0	Q_0 Q_1 Q_2 Q_3
0 0	1 0 0 0
0 1	0 1 0 0
1 0	0 0 1 0
1 1	0 0 0 1

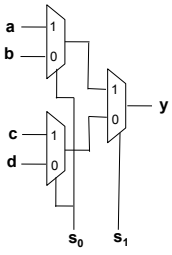
Circuit Example: 2x1 MUX

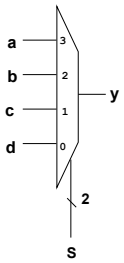
Multiplexor (MUX) selects one of many inputs

MUX(A, B, S) = (A * S) + (B * ~S)



Example 4x1 MUX





Arithmetic and Logical Operations in ISA

- What operations are there?
- Arithmetic Logic Unit (ALU)
  - Hardware that performs operations
  - Only one operation at a time
- How do we implement the operations?
  - Consider AND, OR, NOT, and ADD
  - Input is two bits, output…

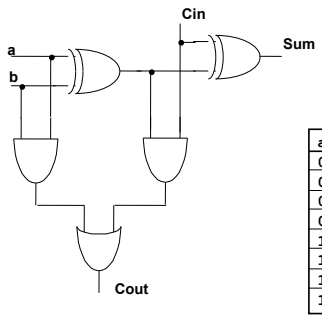
Truth Table for 1-bit Addition

```
  01101100
   01101101
  +00101100
   10011001
```

a	b	C_{in}	Sum	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

What is the circuit for Sum and for Cout?

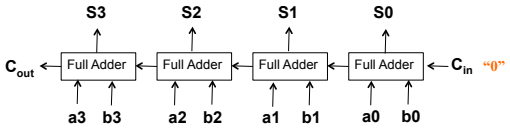
A 1-bit Full Adder



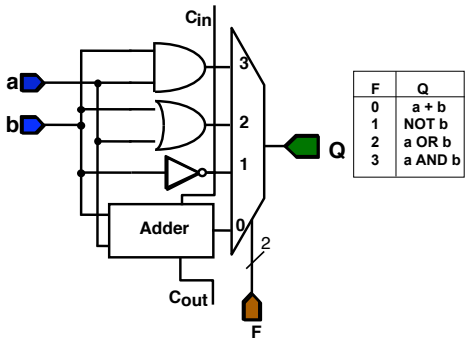

```
  01101100
   01101101
  +00101100
   10011001
```

a	b	C_{in}	Sum	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

Example: 4-bit adder



ALU Slice (Almost)

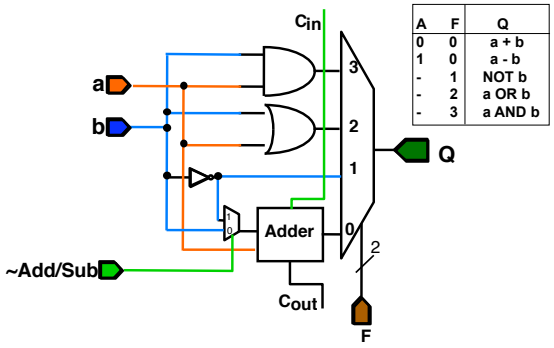


F	Q
0	a + b
1	NOT b
2	a OR b
3	a AND b

Subtraction

- How do we perform integer subtraction?
- What is the HW?

ALU Slice



A	F	Q
0	0	a + b
1	0	a - b
-	1	NOT b
-	2	a OR b
-	3	a AND b

Overflow

Example1:

0100000

0110101_2 $(= 53_{10})$
$+0101010_2$ $(= 42_{10})$
1011111_2 $(=-33_{10})$

Example2:

1000000

1010101_2 $(=-43_{10})$
$+1001010_2$ $(=-54_{10})$
0011111_2 $(= 31_{10})$

Example3:

1100000

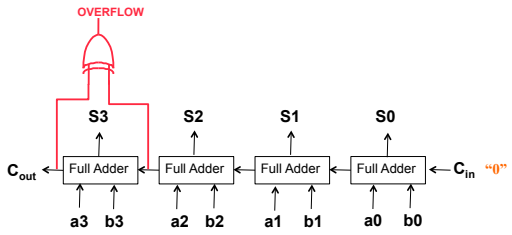
0110101_2 $(= 53_{10})$
$+1101010_2$ $(=-22_{10})$
0011111_2 $(= 31_{10})$

Example4:

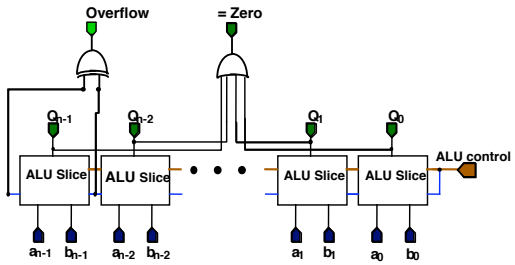
0000000

0010101_2 $(= 21_{10})$
$+0101010_2$ $(= 42_{10})$
0111111_2 $(= 63_{10})$

Overflow Detection for 4-bit adder

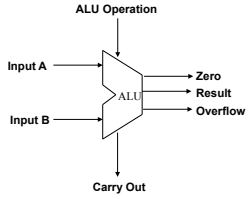


The ALU



Abstraction: The ALU

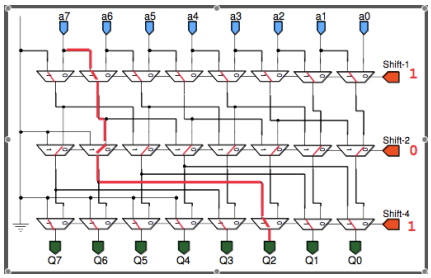
- General structure
- Two operand inputs
- Control inputs



The Shift Operation

- Consider an 8-bit machine
- How do I implement the shift operation?

Shifter



Summary thus far

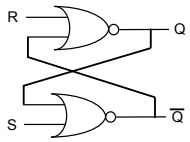
- Given Boolean function, generate a circuit that "realizes" the function.
- Constructed circuits that can add and subtract.
- The ALU: a circuit that can add, subtract, detect overflow, compare, and do bit-wise operations (AND, OR, NOT)
- Shifter

Next up: Storage Elements: Registers, Latches, Buses

Memory Elements

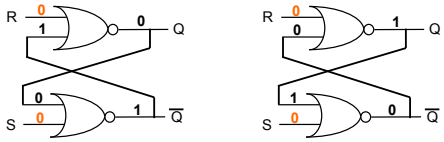
- All the circuit we looked at so far are combinational circuits: the output is a Boolean function of the inputs.
- We need circuits that can remember values. (registers)
- The output of the circuit is a function of the input AND a stored value (state) .
- Circuits with memory are called sequential circuits.

Set-Reset Latch



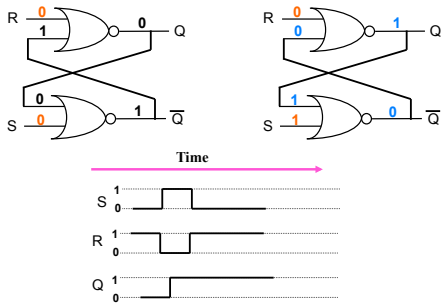
R	S	Q
0	0	Q
0	1	1
1	0	0
1	1	–

Set-Reset Latch (Continued)

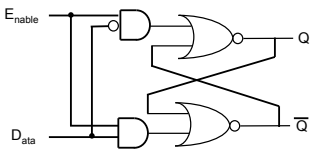


R	S	Q
0	0	Q
0	1	1
1	0	0
1	1	–

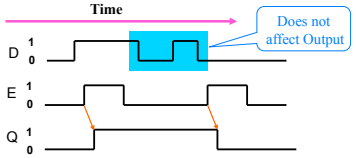
Set-Reset Latch (Continued)



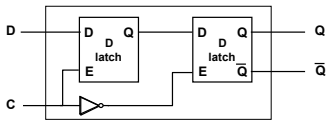
Data Latch (D Latch)



D	E	Q
0	1	0
1	1	1
–	0	Q

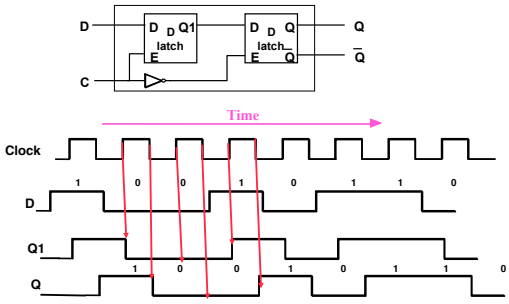


D Flip-Flop



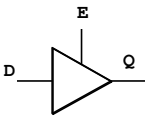
- On C ↑ D is transferred to the first D latch and the second is stable.
- On C ↓ the output of the first stage is transferred to the second (output), and the first stage is stable.

D Flip-Flop Timing



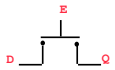
Tri-State Driver

- The Tri-State driver is like a (one directional) switch:
  - When the Enable is on (E=1) it transfers the input to the output.
  - When the Enable is off (E=0) it disconnects the output.



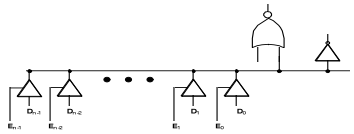
D	E	Q
0	1	0
1	1	1
-	0	Z

Z :- High Impedance

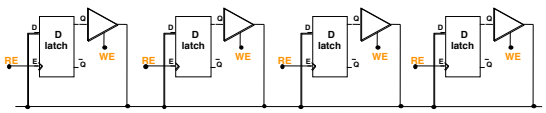


Bus Connections

- The Bus: Many to many connections.
- Mutual exclusion: At most one Enable is on!

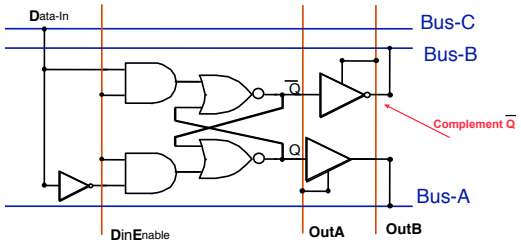


Register Cells on a bus



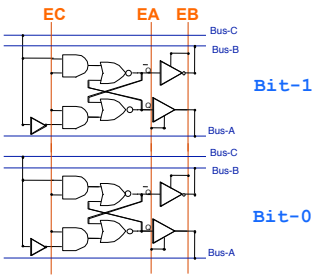
One can "source" and "sink" from any cell on the bus by activating the right controls (WE and RE).

3-Port Register Cell

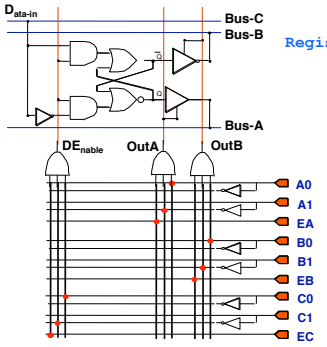


- Stores one bit of a register
- Can Read onto Bus-A & Bus-B and Write fromBus-C Simultaneously

3-Port Register File

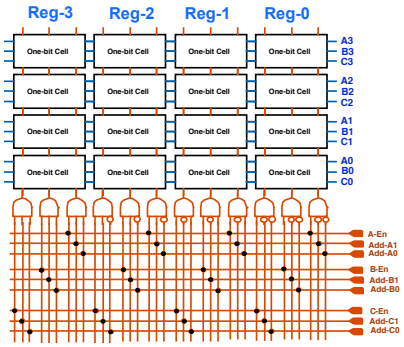


Address Decode Circuit



Register address: 01

Register File (Four 4-bit Registers)



Summary

- Given Boolean function, generate a circuit that "realize" the function.
- Constructed circuits that can add and subtract.
- The ALU: a circuit that can add, subtract, detect overflow, compare, and do bit-wise operations (AND, OR, NOT)
- Shifter
- Memory Elements: SR-Latch, D Latch, D Flip-Flop
- Tri-state drivers & Bus Communication
- Register Files
- Control Signals modify what circuit does with inputs
  - ALU, Shift, Register Read/Write