Basics of Logic Design: Boolean Algebra, Logic Gates

CPS 104

Lecture 8

Today's Lecture

Outline

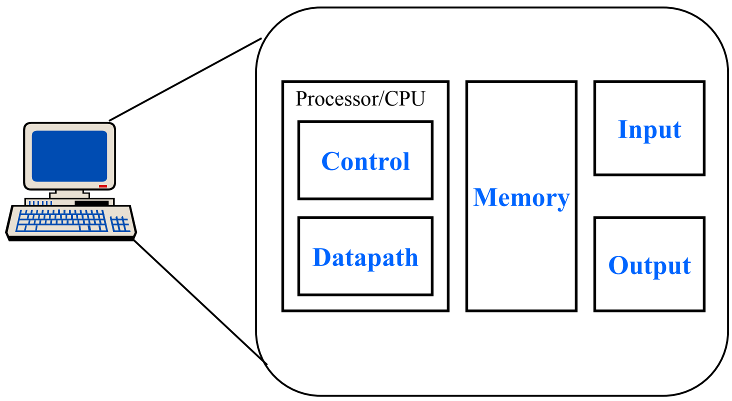
- Building the building blocks…
- Logic Design
  - Truth tables, Boolean functions, Gates and Circuits

Reading

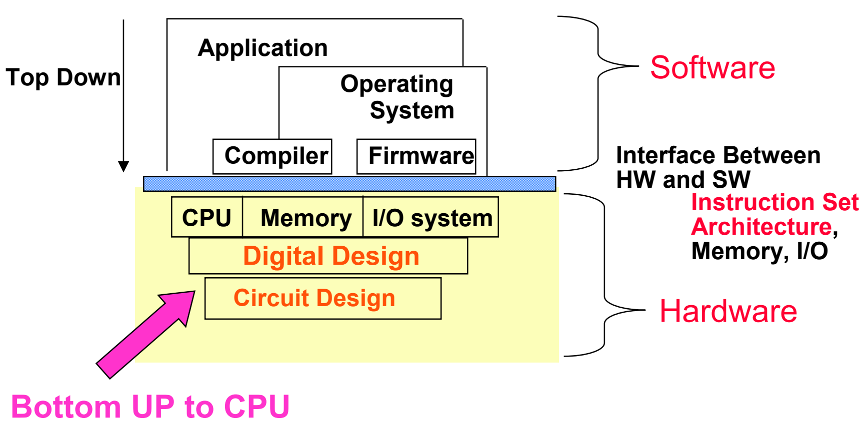
Appendix B, Chapter 4

The Big Picture

- **The Five Classic Components of a Computer**



What We've Done, Where We're Going



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

Boolean Algebra

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

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

Boolean Functions

- Example Boolean Functions: NOT, AND, OR, XOR, . . .

a	NOT(a)
0	1
1	0

a	b	AND(a,b)
0	0	0
0	1	0
1	0	0
1	1	1

a	b	OR(a,b)
0	0	0
0	1	1
1	0	1
1	1	1

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

a	b	XNOR(a,b)
0	0	1
0	1	0
1	0	0
1	1	1

a	b	NOR(a,b)
0	0	1
0	1	0
1	0	0
1	1	0

Boolean Functions and Expressions

- Boolean algebra notation: Use * for AND, + for OR, ~ for NOT.
  - Not is also written as A' and $\overline{A}$
- Using the above notation we can write Boolean expressions for functions

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

- We can evaluate the Boolean expression with all possible argument values to construct a truth table.
- What is truth table for F?

Boolean Function Simplification

- Boolean expressions can be simplified by using the following rules:
  - A*A = A
  - A* 0 = 0
  - A*1 = A
  - A*~A = 0

  - A+A = A
  - A+0 = A
  - A+1 = 1
  - A+~A = 1

  - A*B = B*A
  - A*(B+C) = (B+C)*A = A*B + A*C

Boolean Function Simplification

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

f_1 = ~a*~b*c + ~a*b*c + a*~b*c + a*b*c

f_2 = ~a*~b*~c + ~a*~b*c + a*b*~c + a*b*c

Simplify these functions...

Boolean Functions and Expressions

- **The Fundamental Theorem of Boolean Algebra: Every Boolean function can be written in disjunctive normal form as an OR of ANDs (Sum-of products) of it's arguments or their complements.**

  "Proof:" Write the truth table, construct sum-of-product from the table.

a	b	XNOR(a,b)
0	0	1
0	1	0
1	0	0
1	1	1

XNOR = (~a * ~b) + (a * b)

Boolean Functions and Expressions

- **Example-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

f_1 = ~a*~b*c + ~a*b*~c + a*~b*~c + a*b*c

f_2 = ~a*~b*~c + ~a*~b*c + a*b*~c + a*b*c

DeMorgan’s Laws

- ~(A+B) = ~A * ~B
- ~(A*B) = ~A + ~B

Example:

- ~C*~A*B + ~C*A*~B + C*A*B + C*~A*~B
- Use only XOR to represent this function

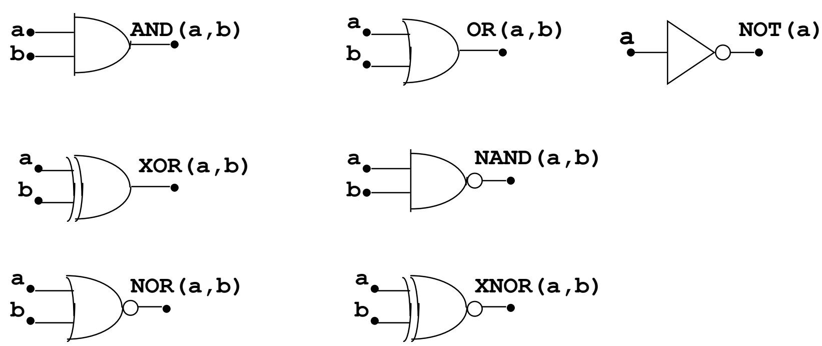
Applying the Theory

- Lots of good theory
- Can reason about complex boolean expressions
- Now we have to make it real…

Boolean Gates

- Gates are electronic devices that implement simple Boolean functions

Examples



Reality Check

- Basic 1 or 2 Input Boolean Gate 1- 4 Transistors

Pentium III

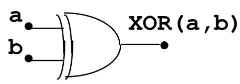
- Processor Core 9.5 Million Transistors
- Total: 28 Million Transistors

Pentium 4

- Total: 42 Million Transistors

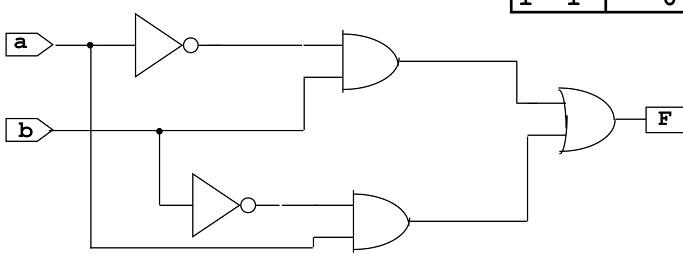
Boolean Functions, Gates and Circuits

- **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



Digital Design Examples

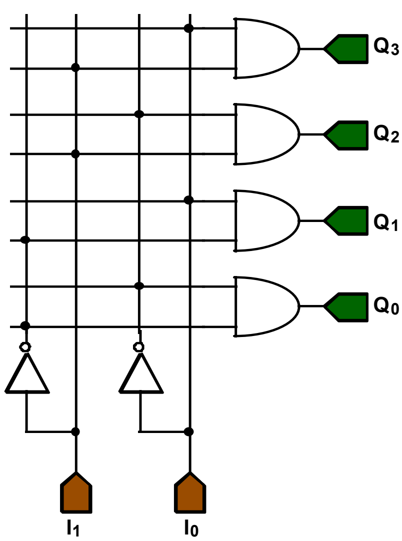
Input: 2 bits representing an unsigned number (n)
Output: n^2 as 4-bit unsigned binary number

Input: 2 bits representing an unsigned number (n)
Output: 3-n as unsigned binary number

Design Example

- Consider machine with 4 registers
- Given 2-bit input (register specifier, I_1, I_0)
- Want one of 4 output bits (O_3-O_0) to be 1
  - ➢ E.g., allows a single register to be accessed
- What is the circuit for this?

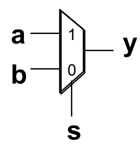
Circuit Example: Decoder



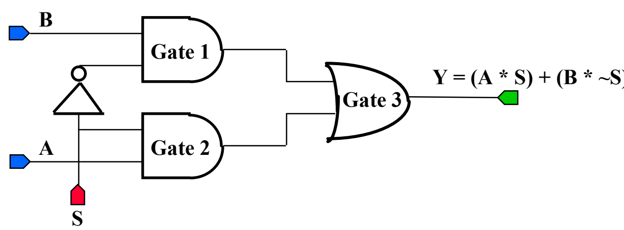
I_1	I_0	Q_0	Q_1	Q_2	Q_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Circuit Example: 2x1 MUX

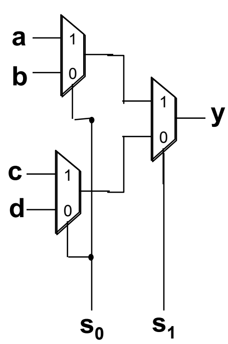
Multiplexor (MUX) selects from one of many inputs

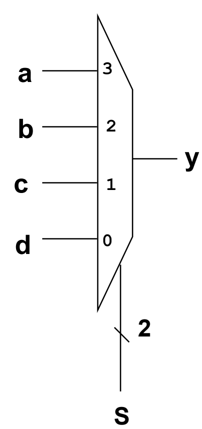


MUX(A, B, S) = (A * S) + (B * ~S)



Example 4x1 MUX





Arithmetic and Logical Operations in ISA

- What operations are there?
- How do we implement them?
  - Consider a 1-bit Adder

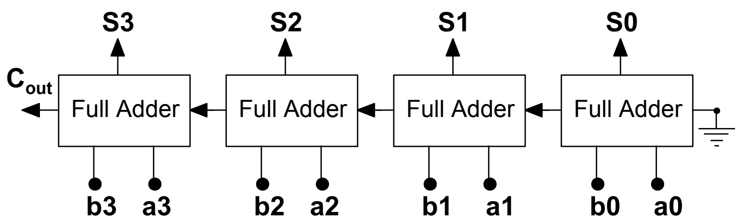
Summary

- Boolean Algebra & functions
- Logic gates (AND, OR, NOT, etc)
- Multiplexors

<u>Reading</u>

- Appendix B, Chapter 4

Example: 4-bit adder



Subtraction

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

Overflow

Example1:

0100000 (carries)

0110101_2 (= 53_{10})

$+0101010_2$ (= 42_{10})

1011111_2 (= -33_{10})

Example2:

1000000 (carries)

1010101_2 (= -43_{10})

$+1001010_2$ (= -54_{10})

0011111_2 (= 31_{10})

Example3:

1100000 (carries)

0110101_2 (= 53_{10})

$+1101010_2$ (= -22_{10})

0011111_2 (= 31_{10})

Example4:

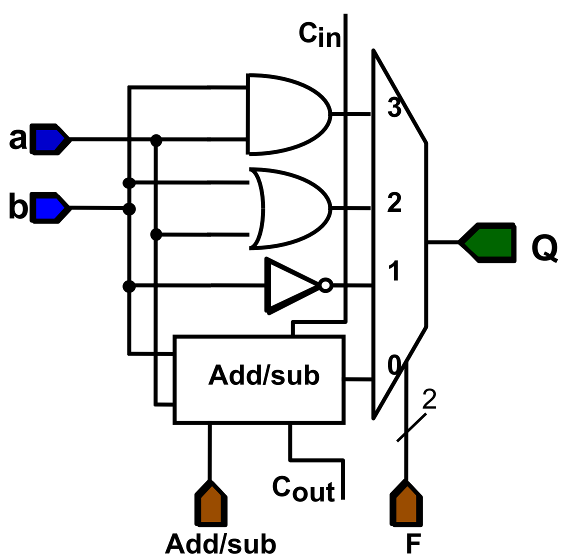
0000000 (carries)

0010101_2 (= 21_{10})

$+0101010_2$ (= 42_{10})

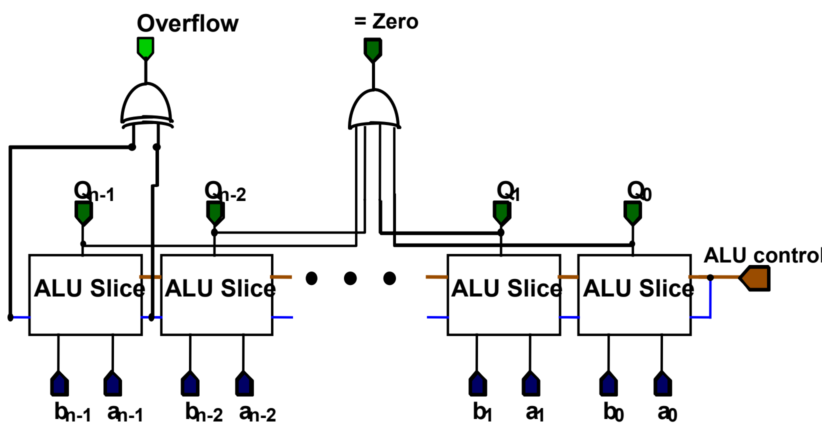
0111111_2 (= 63_{10})

ALU Slice



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

The ALU



Summary

- Boolean Algebra & functions
- Logic gates (AND, OR, NOT, etc)
- Multiplexors
- Adder
- Arithmetic Logic Unit (ALU)
- Reading
- Appendix B, Chapter 4