CS61C : Machine Structures

Lecture 4.1.1

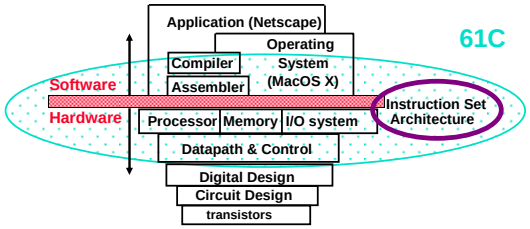
Logic Gates and Combinational Logic

2004-07-12

Kurt Meinz

inst.eecs.berkeley.edu/~cs61c

What are "Machine Structures"?



Coordination of many *levels of abstraction*

We'll investigate lower abstraction layers! (contract between HW & SW)

Below the Program

- **High-level language program (in C)**

```
swap  int v[], int k){
      int temp;
      temp = v[k];
      v[k] = v[k+1];
      v[k+1] = temp;
}
```

C compiler

- **Assembly language program (for MIPS)**

```
swap: sll   $2, $5, 2
      add   $2, $4,$2
      lw    $15, 0($2)
      lw    $16, 4($2)
      sw    $16, 0($2)
      sw    $15, 4($2)
      jr    $31
```

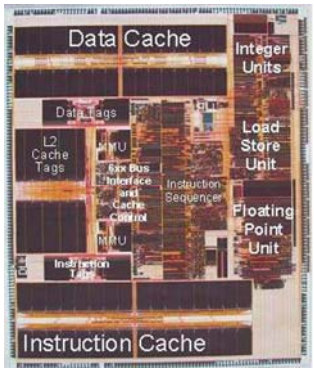
assembler

- **Machine (object) code (for MIPS)**

```
000000 00000 00101 0001000010000000
000000 00100 00010 0001000000100000 . . .
```

?

Physical Hardware - PowerPC 750



Digital Design Basics (1/2)

- **Next 4 weeks: we'll study how a modern processor is built starting with basic logic elements as building blocks.**
- **Why study logic design?**
  - **Understand what processors can do fast and what they can't do fast (avoid slow things if you want your code to run fast!)**
  - **Background for more detailed hardware courses (CS 150, CS 152)**

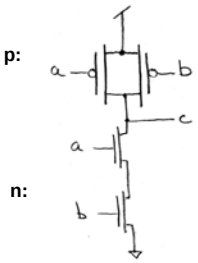
Digital Design Basics (2/2)

- **ISA is very important abstraction layer**
  - **Contract between HW and SW**
  - **Can you peek across abstraction?**
  - **Can you depend "across abstraction"?**
- **Voltages are analog, quantized to 0/1**
- **Circuit delays are fact of life**
- **Two types**
  - **Stateless Combinational Logic (&,|,~)**
  - **State circuits (e.g., registers)**

Outline

- Transistors
- Logic Gates
- Combinational Logic
- Boolean Algebra

Transistors (1/3)



CMOSFET Transistors:

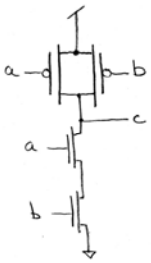
* Physically exist!

* Voltages are quantized

* Only 2 Types:
  - P-channel:
    0 on gate -> pull up (1)
  - N-channel:
    1 on gate -> pull down (0)

* Undriven otherwise.

Transistors (2/3)



CMOSFET Transistors:

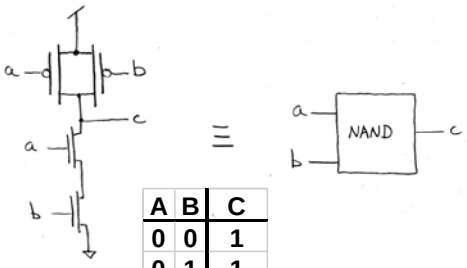
* have delay and require power

* can be combined to perform logical operations and maintain state.

  - logical operations will be our starting point for digital design

  - state tomorrow

Transistors (3/3): CMOS ➔ Nand



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

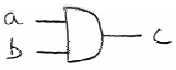
Logic Gates (1/4)

- Transistors are too low level
  - Good for measuring performance, power.
  - Bad for logical design / analysis

- Gates are collections of transistors wired in a certain way
  - Can represent and reason about gates with truth tables and Boolean algebra
  - Assume know truth tables and Boolean algebra from a math or circuits course.
  - Section B.2 in the textbook has a review

Logic Gates (2/4)

AND

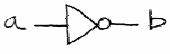


ab	c
00	0
01	0
10	0
11	1

OR

ab	c
00	0
01	1
10	1
11	1

NOT

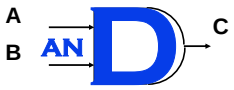


a	b
0	1
1	0

Logic Gates (3/4)

AND Gate

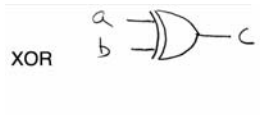
Symbol



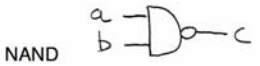
Definition

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

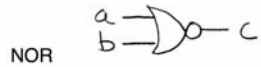
Logic Gates (4/4)



ab	c
00	0
01	1
10	1
11	0

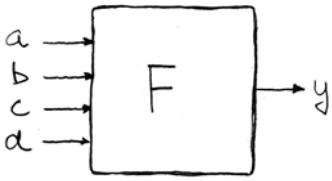


ab	c
00	1
01	1
10	1
11	0



ab	c
00	1
01	0
10	0
11	0

Truth Tables (1/6)

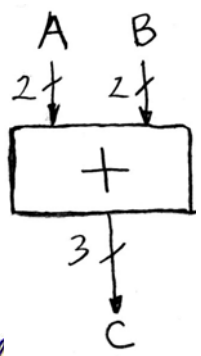


a	b	c	d	y
0	0	0	0	F(0,0,0,0)
0	0	0	1	F(0,0,0,1)
0	0	1	0	F(0,0,1,0)
0	0	1	1	F(0,0,1,1)
0	1	0	0	F(0,1,0,0)
0	1	0	1	F(0,1,0,1)
0	1	1	0	F(0,1,1,0)
0	1	1	1	F(0,1,1,1)
1	0	0	0	F(1,0,0,0)
1	0	0	1	F(1,0,0,1)
1	0	1	0	F(1,0,1,0)
1	0	1	1	F(1,0,1,1)
1	1	0	0	F(1,1,0,0)
1	1	0	1	F(1,1,0,1)
1	1	1	0	F(1,1,1,0)
1	1	1	1	F(1,1,1,1)

TT (2/6) Ex #1: 1 iff one (not both) a,b=1

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

TT (3/6): Example #2: 2-bit adder



A a_1a_0	B b_1b_0	C $c_2c_1c_0$
00	00	000
00	01	001
00	10	010
00	11	011
01	00	001
01	01	010
01	10	011
01	11	100
10	00	010
10	01	011
10	10	100
10	11	101
11	00	011
11	01	100
11	10	101
11	11	110

TT (4/6): Ex #3: 32-bit unsigned adder

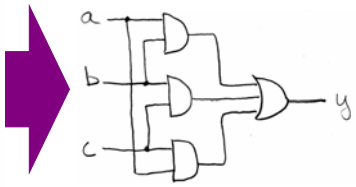
A	B	C
000 ... 0	000 ... 0	000 ... 00
000 ... 0	000 ... 1	000 ... 01
.	.	.
.	.	.
.	.	.
111 ... 1	111 ... 1	111 ... 10

TT (5/6): Conversion: 3-input majority

a	b	c	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

TT (6/6): Conversion: 3-input majority

a	b	c	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



Combinational Logic (1/2)

A ***combinational*** logic block is one in which the output is a function only of its current input.

- Combinational logic cannot have memory.
- Everything we've seen so far is CL
- CL will have delay (f(transistors))
  - More later.

Representations of CL Circuits (2/2)…

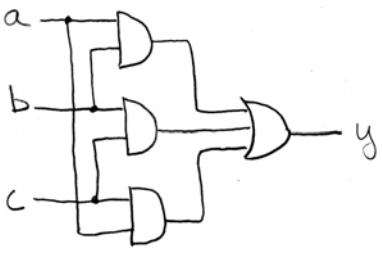
- Logic Gates ✓
- Truth Tables ✓
- Boolean Algebra ? ? ?

Boolean Algebra (1/7)

- George Boole, 19th Century mathematician
- Developed a mathematical system (algebra) involving logic, later known as "Boolean Algebra"
- Primitive functions: AND, OR and NOT
- The power of BA is there's a one-to-one correspondence between circuits made up of AND, OR and NOT gates and equations in BA

+ means OR, • means AND, $\bar{x}$ means NOT

BA (2/7): e.g., for majority fun…



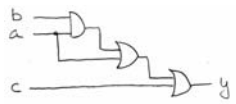
$y = a \cdot b + a \cdot c + b \cdot c$

$y = ab + ac + bc$

BA (3/7):Laws of Boolean Algebra

$x \cdot \overline{x} = 0$	$x + \overline{x} = 1$	complementarity
$x \cdot 0 = 0$	$x + 1 = 1$	laws of 0's and 1's
$x \cdot 1 = x$	$x + 0 = x$	identities
$x \cdot x = x$	$x + x = x$	idempotent law
$x \cdot y = y \cdot x$	$x + y = y + x$	commutativity
$(xy)z = x(yz)$	$(x + y) + z = x + (y + z)$	associativity
$x(y + z) = xy + xz$	$x + yz = (x + y)(x + z)$	distribution
$xy + x = x$	$(x + y)x = x$	uniting theorem
$\overline{x \cdot y} = \overline{x} + \overline{y}$	$\overline{(x + y)} = \overline{x} \cdot \overline{y}$	DeMorgan's Law

BA (4/7): Circuit & Algebraic Simplification



original circuit

$y = ((ab) + a) + c$

equation derived from original circuit

$= ab + a + c$
$= a(b + 1) + c$
$= a(1) + c$
$= a + c$

algebraic simplification

simplified circuit

BA (5/7): Simplification Example

$$
\begin{aligned}
y &= ab + a + c \\
&= a(b + 1) + c && \textit{distribution, identity} \\
&= a(1) + c && \textit{law of 1's} \\
&= a + c && \textit{identity}
\end{aligned}
$$

BA (6/7): Canonical forms (1/2)

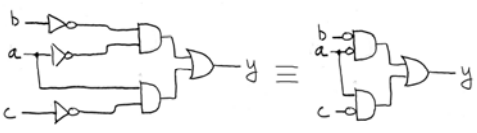
	abc	y
$\overline{a} \cdot \overline{b} \cdot \overline{c}$	000	1
$\overline{a} \cdot \overline{b} \cdot c$	001	1
	010	0
	011	0
$a \cdot \overline{b} \cdot \overline{c}$	100	1
	101	0
$a \cdot b \cdot \overline{c}$	110	1
	111	0

Sum-of-products (ORs of ANDs)

$$y = \overline{a}\overline{b}\overline{c} + \overline{a}\overline{b}c + a\overline{b}\overline{c} + ab\overline{c}$$

BA (7/7): Canonical forms (2/2)

$$
\begin{aligned}
y &= \overline{a}\overline{b}\overline{c} + \overline{a}\overline{b}c + a\overline{b}\overline{c} + ab\overline{c} \\
&= \overline{a}\overline{b}(\overline{c} + c) + a\overline{c}(\overline{b} + b) && \textit{distribution} \\
&= \overline{a}\overline{b}(1) + a\overline{c}(1) && \textit{complementarity} \\
&= \overline{a}\overline{b} + a\overline{c} && \textit{identity}
\end{aligned}
$$



"And In conclusion…"

- **Use this table and techniques we learned to transform from 1 to another**