Introduction to Digital Design

Week 2: Switches, Transistors, Logic Gates

Yao Zheng
Assistant Professor
University of Hawai'i at Mānoa
Department of Electrical Engineering

Overview

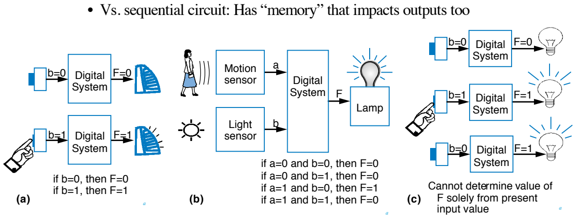
- Combinational circuit
  - Circuit outputs depend solely on the present combination of the circuit inputs.
- Switches
  - Electronic switches are the basis of binary digital circuits.
  - Three parts: source, output, and control.
- CMOS Transistors
  - Miniaturized switches.
  - NMOS, PMOS.
- Boolean Logic Gates
  - Boolean algebra primer: Boolean expression
  - Basic logic gates: NOT, AND, OR

2

2.1

Introduction

- Let's learn to design digital circuits, starting with a simple form of circuit:
  - ***Combinational circuit***
    - Outputs depend solely on the *present combination of the circuit inputs' values*
    - Vs. sequential circuit: Has "memory" that impacts outputs too



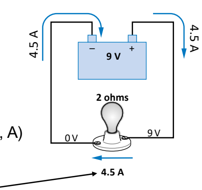
3

Note: Slides with animation are denoted with a small red "a" near the animated items

2.2

Switches

- Electronic switches are the basis of binary digital circuits
  - Electrical terminology
    - ***Voltage***: Difference in electric potential between two points (volts, V)
      - Analogous to water pressure
    - ***Resistance***: Tendency of wire to resist current flow (ohms, Ω)
      - Analogous to water pipe diameter
    - ***Current***: Flow of charged particles (amps, A)
      - Analogous to water flow
    - V = I * R (Ohm's Law)
      - 9 V = I * 2 ohms
      - I = 4.5 A

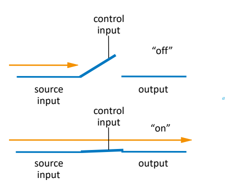


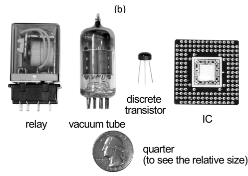
If a 9V potential difference is applied across a 2 ohm resistor, then 4.5 A of current will flow.

4

Switches

- A switch has three parts
  - Source input, and output
    - Current tries to flow from source input to output
  - Control input
    - Voltage that controls whether that current can flow
- The amazing shrinking switch
  - 1930s: Relays
  - 1940s: Vacuum tubes
  - 1950s: Discrete transistor
  - 1960s: Integrated circuits (ICs)
    - Initially just a few transistors on IC
    - Then tens, hundreds, thousands...

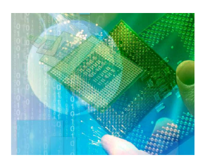


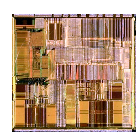


5

Moore's Law

- IC capacity doubling about every 18 months for several decades
  - Known as "Moore's Law" after Gordon Moore, co-founder of Intel
    - Predicted in 1965 predicted that components per IC would double roughly every year or so
  - Book cover depicts related phenomena
    - For a particular number of transistors, the IC area shrinks by half every 18 months
      - Consider how much shrinking occurs in just 10 years (try drawing it)
      - Enables incredibly powerful computation in incredibly tiny devices
  - Today's ICs hold *billions* of transistors
    - The first Pentium processor (early 1990s) needed only 3 million





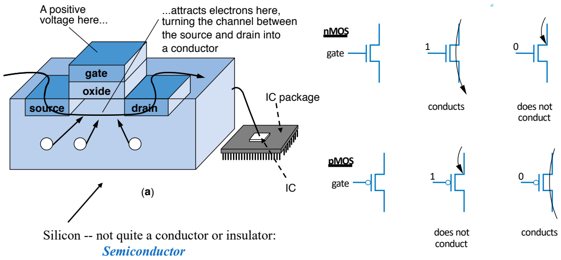
An Intel Pentium processor IC having millions of transistors

6

2.3

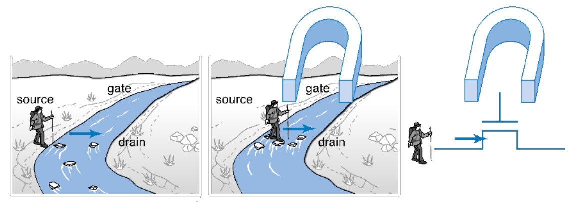
The CMOS Transistor

- CMOS transistor
  - Basic switch in modern ICs



7

CMOS Transistor Analogy



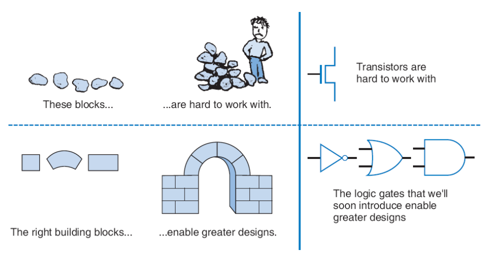
8

2.4

Boolean Logic Gates

Building Blocks for Digital Circuits

(Because Switches are Hard to Work With)



- "Logic gates" are better digital circuit building blocks than switches (transistors)
  - Why?...

9

Boolean Algebra and its Relation to Digital Circuits

- To understand the benefits of "logic gates" vs. switches, we should first understand Boolean algebra
- "Traditional" algebra
  - Variables represent real numbers (x, y)
  - Operators operate on variables, return real numbers (2.5*x + y - 3)
- ***Boolean Algebra***
  - Variables represent 0 or 1 only
  - Operators return 0 or 1 only
  - Basic operators
    - AND: *a AND b* returns 1 only when both a=1 and b=1
    - OR: *a OR b* returns 1 if either (or both) a=1 or b=1
    - NOT: *NOT a* returns the opposite of a (1 if a=0, 0 if a=1)

a	b	AND
0	0	0
0	1	0
1	0	0
1	1	1

a	b	OR
0	0	0
0	1	1
1	0	1
1	1	1

a	NOT
0	1
1	0

10

Boolean Algebra and its Relation to Digital Circuits

- Developed mid-1800's by George Boole to formalize human thought
  - Ex: "I'll go to lunch if Mary goes OR John goes, AND Sally does not go."
    - Let F represent my going to lunch (1 means I go, 0 I don't go)
    - Likewise, m for Mary going, j for John, and s for Sally
    - Then **F = (m OR j) AND NOT(s)**
  - Nice features
    - Formally evaluate
      - m=1, j=0, s=1 --> F = (1 OR 0) AND NOT(1) = 1 AND 0 = **0**
    - Formally transform
      - F = (m and NOT(s)) OR (j and NOT(s))
        - » Looks different, but same function
        - » We'll show transformation techniques soon
    - Formally prove
      - Prove that if Sally goes to lunch (s=1), then I don't go (F=0)
      - F = (m OR j) AND NOT(1) = (m OR j) AND 0 = 0

a	b	AND
0	0	0
0	1	0
1	0	0
1	1	1

a	b	OR
0	0	0
0	1	1
1	0	1
1	1	1

a	NOT
0	1
1	0

11

Evaluating Boolean Equations

- Evaluate the Boolean equation **F = (a AND b) OR (c AND d)** for the given values of variables a, b, c, and d:
  - Q1: a=1, b=1, c=1, d=0.
    - Answer: F = (1 AND 1) OR (1 AND 0) = 1 OR 0 = 1.
  - Q2: a=0, b=1, c=0, d=1.
    - Answer: F = (0 AND 1) OR (0 AND 1) = 0 OR 0 = 0.
  - Q3: a=1, b=1, c=1, d=1.
    - Answer: F = (1 AND 1) OR (1 AND 1) = 1 OR 1 = 1.

a	b	AND
0	0	0
0	1	0
1	0	0
1	1	1

a	b	OR
0	0	0
0	1	1
1	0	1
1	1	1

a	NOT
0	1
1	0

12

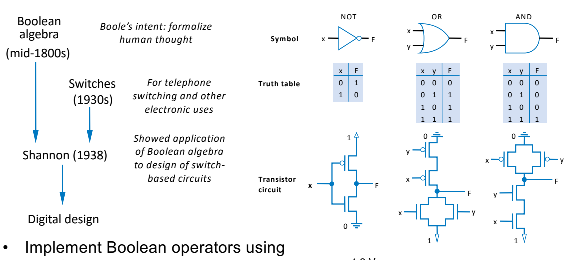
Converting to Boolean Equations

- Convert the following English statements to a Boolean equation
  - Q1. a is 1 and b is 1.
    - Answer: F = a AND b
  - Q2. either of a or b is 1.
    - Answer: F = a OR b
  - Q3. a is 1 and b is 0.
    - Answer: F = a AND NOT(b)
  - Q4. a is not 0.
    - Answer:
      - (a) Option 1: F = NOT(NOT(a))
      - (b) Option 2: F = a

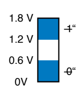
Converting to Boolean Equations

- Q1. A fire sprinkler system should spray water if high heat is sensed and the system is set to enabled.
  - Answer: Let Boolean variable h represent “high heat is sensed,” e represent “enabled,” and F represent “spraying water.” Then an equation is: F = h AND e.
- Q2. A car alarm should sound if the alarm is enabled, and either the car is shaken or the door is opened.
  - Answer: Let a represent “alarm is enabled,” s represent “car is shaken,” d represent “door is opened,” and F represent “alarm sounds.” Then an equation is: F = a AND (s OR d).
  - (a) Alternatively, assuming that our door sensor d represents “door is closed” instead of open (meaning d=1 when the door is closed, 0 when open), we obtain the following equation: F = a AND (s OR NOT(d)).

Relating Boolean Algebra to Digital Design



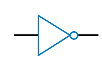
- Implement Boolean operators using transistors
  - Call those implementations ***logic gates***.
  - ***Lets us build circuits by doing math*** -- powerful concept



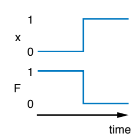
1 and 0 each actually corresponds to a voltage range

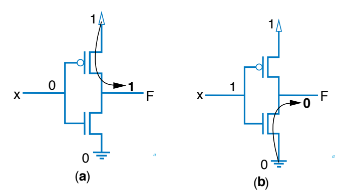
Next slides show how these circuits work. Note: The above OR/AND implementations are inefficient; we'll show why, and show better ones, later.

NOT gate



x	F
0	1
1	0

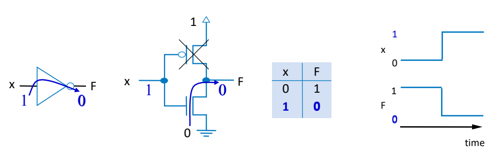




When the input is 0

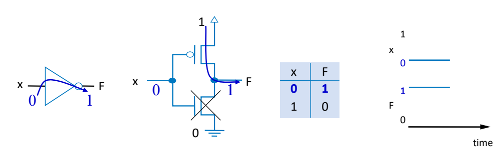
When the input is 1

Relating a Logic Gate Symbol, Circuit, Truth Table, and Timing Diagram



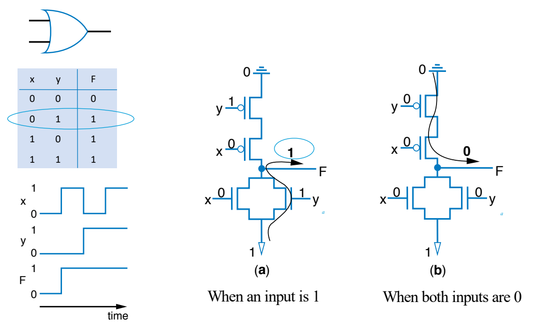
Setting x to 1 causes F to be 0

Relating a Logic Gate Symbol, Circuit, Truth Table, and Timing Diagram



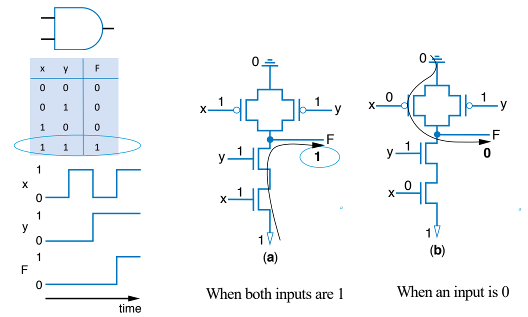
Setting x to 0 causes F to be 1

OR gate



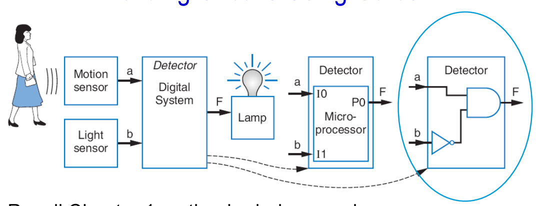
19

AND gate



20

Building Circuits Using Gates



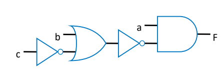
- Recall Chapter 1 motion-in-dark example
  - Turn on lamp (F=1) when motion sensed (a=1) and no light (b=0)
  - F = a AND NOT(b)
  - Build using logic gates, AND and NOT, as shown
  - We just built our first digital circuit!

21

Example: Converting a Boolean Equation to a Circuit of Logic Gates

Start from the output, work back towards the inputs

- Q: Convert the following equation to logic gates:
  F = a AND NOT(b OR NOT(c))



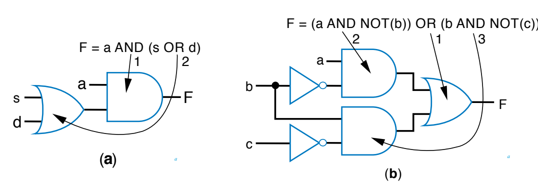
22

Example: Converting a Boolean Equation to a Circuit of Logic Gates

Video

23

More examples



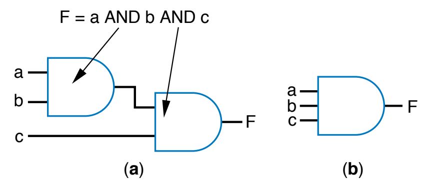
Start from the output, work back towards the inputs

24

More examples

Video

Using gates with more than 2 inputs

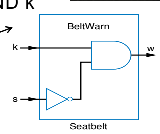


Can think of as AND(a,b,c)

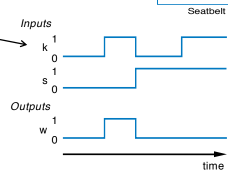
Example: Seat Belt Warning Light System

- Design circuit for warning light
- Sensors
  - s=1: seat belt fastened
  - k=1: key inserted
- Capture Boolean equation
  - seat belt not fastened, and key inserted
- Convert equation to circuit

w = NOT(s) AND k



- *Timing diagram* illustrates circuit behavior
  - We set inputs to any values
  - Output set according to circuit



Example: Seat Belt Warning Light System

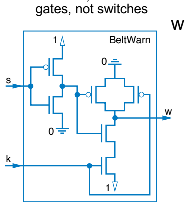
Video

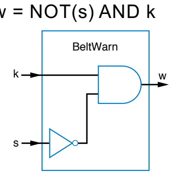
Gates vs. switches

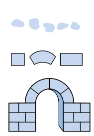
Notice

- Boolean algebra enables easy capture as equation and conversion to circuit
  - How design with switches?
  - Of course, logic gates are built from switches, but we think at level of logic gates, not switches

w = NOT(s) AND k

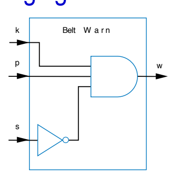




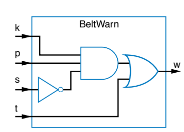


More examples: Seat belt warning light extensions

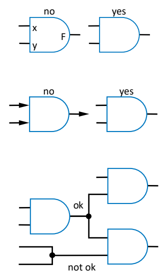
- Only illuminate warning light if person is in the seat (p=1), and seat belt not fastened and key inserted
- w = p AND NOT(s) AND k



- Given t=1 for 5 seconds after key inserted. Turn on warning light when t=1 (to check that warning lights are working)
- w = (p AND NOT(s) AND k) OR t



Some Gate-Based Circuit Drawing Conventions



Summary

- Combinational circuit
  - Circuit outputs depend solely on the present combination of the circuit inputs.
- Switches
  - Electronic switches are the basis of binary digital circuits.
  - Three parts: source, output, and control.
- CMOS Transistors
  - Miniaturized switches.
  - NMOS, PMOS.
- Boolean Logic Gates
  - Boolean algebra primer: Boolean expression
  - Basic logic gates: NOT, AND, OR