Chapter 4: The Building Blocks: Binary Numbers, Boolean Logic, and Gates

Invitation to Computer Science,

C++ Version, Third & Fourth Edition

Spring 2008: Additions by S. Steinfadt

Objectives

In this chapter, you will learn about:

- The binary numbering system
- Boolean logic and gates
- Building computer circuits
- Control circuits

Introduction

- Chapter 4 focuses on hardware design (also called logic design)
    - How to represent and store information inside a computer
    - How to use the principles of symbolic logic to design gates
    - How to use gates to construct circuits that perform operations such as adding and comparing numbers, and fetching instructions

The Binary Numbering System

- A computer's internal storage techniques are different from the way people represent information in daily lives
- Information inside a digital computer is stored as a collection of binary data

Binary Representation of Numeric and Textual Information

- Binary numbering system
  - Base-2
  - Built from ones and zeros
  - Each position is a power of 2

    $1101 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$
- Decimal numbering system
  - Base-10
  - Each position is a power of 10

    $3052 = 3 \times 10^3 + 0 \times 10^2 + 5 \times 10^1 + 2 \times 10^0$

Figure 4.2
Binary-to-Decimal
Conversion Table

Binary	Decimal	Binary	Decimal
0	0	10000	16
1	1	10001	17
10	2	10010	18
11	3	10011	19
100	4	10100	20
101	5	10101	21
110	6	10110	22
111	7	10111	23
1000	8	11000	24
1001	9	11001	25
1010	10	11010	26
1011	11	11011	27
1100	12	11100	28
1101	13	11101	29
1110	14	11110	30
1111	15	11111	31

Binary Representation of Numeric and Textual Information (continued)

- Representing integers
  - Decimal integers are converted to binary integers
  - Given k bits, the largest unsigned integer is $2^k - 1$
    - Given 4 bits, the largest is $2^4-1 = 15$
  - Signed integers must also represent the sign (positive or negative) - ***Sign/Magnitude notation***

Binary Representation of Numeric and Textual Information (continued)

- Representing real numbers
  - Real numbers may be put into binary scientific notation: $a \times 2^b$ *(or* $\pm M \times B^{\pm E}$*)*
    - Example: 101.11×2^0
  - Number then normalized so that first significant digit is immediately to the right of the binary point
    - Example: $.10111 \times 2^3$
  - Mantissa and exponent then stored

Binary Representation of Numeric and Textual Information (continued)

- Characters are mapped onto binary numbers
  - ASCII code set
    - 8 bits per character; 256 character codes
  - UNICODE code set
    - 16 bits per character; 65,536 character codes
- Text strings are sequences of characters in some encoding

Binary Representation of Textual Information (cont'd)

ASCII
8 bits long

Decimal	Binary	Val.
48	00110000	0
49	00110001	1
50	00110010	2
51	00110011	3
52	00110100	4
53	00110101	5
54	00110110	6
55	00110111	7
56	00111000	8
57	00111001	9
58	00111010	:
59	00111011	;
60	00111100	<
61	00111101	=
62	00111110	>
63	00111111	?
64	01000000	@
65	01000001	A
66	01000010	B

Dec.	Unicode	Charac.
0x30	0x0030	[0]
0x31	0x0031	[1]
0x32	0x0032	[2]
0x33	0x0033	[3]
0x34	0x0034	[4]
0x35	0x0035	[5]
0x36	0x0036	[6]
0x37	0x0037	[7]
0x38	0x0038	[8]
0x39	0x0039	[9]
0x3A	0x003A	[:]
0x3B	0x003B	[;]
0x3C	0x003C	[<]
0x3D	0x003D	[=]
0x3E	0x003E	[>]
0x3F	0x003F	[?]
0x40	0x0040	[@]
0x41	0x0041	[A]
0x42	0x0042	[B]

Unicode
16 bits long

Partial listings only!

Binary Representation of Sound and Images

- Multimedia data is sampled to store a digital form, with or without detectable differences
- Representing sound data
  - Sound data must be digitized for storage in a computer
  - Digitizing means periodic sampling of amplitude values

Binary Representation of Sound and Images (continued)

- From samples, original sound may be approximated

- To improve the approximation:

  - Sample more frequently (*increase sampling rate)*

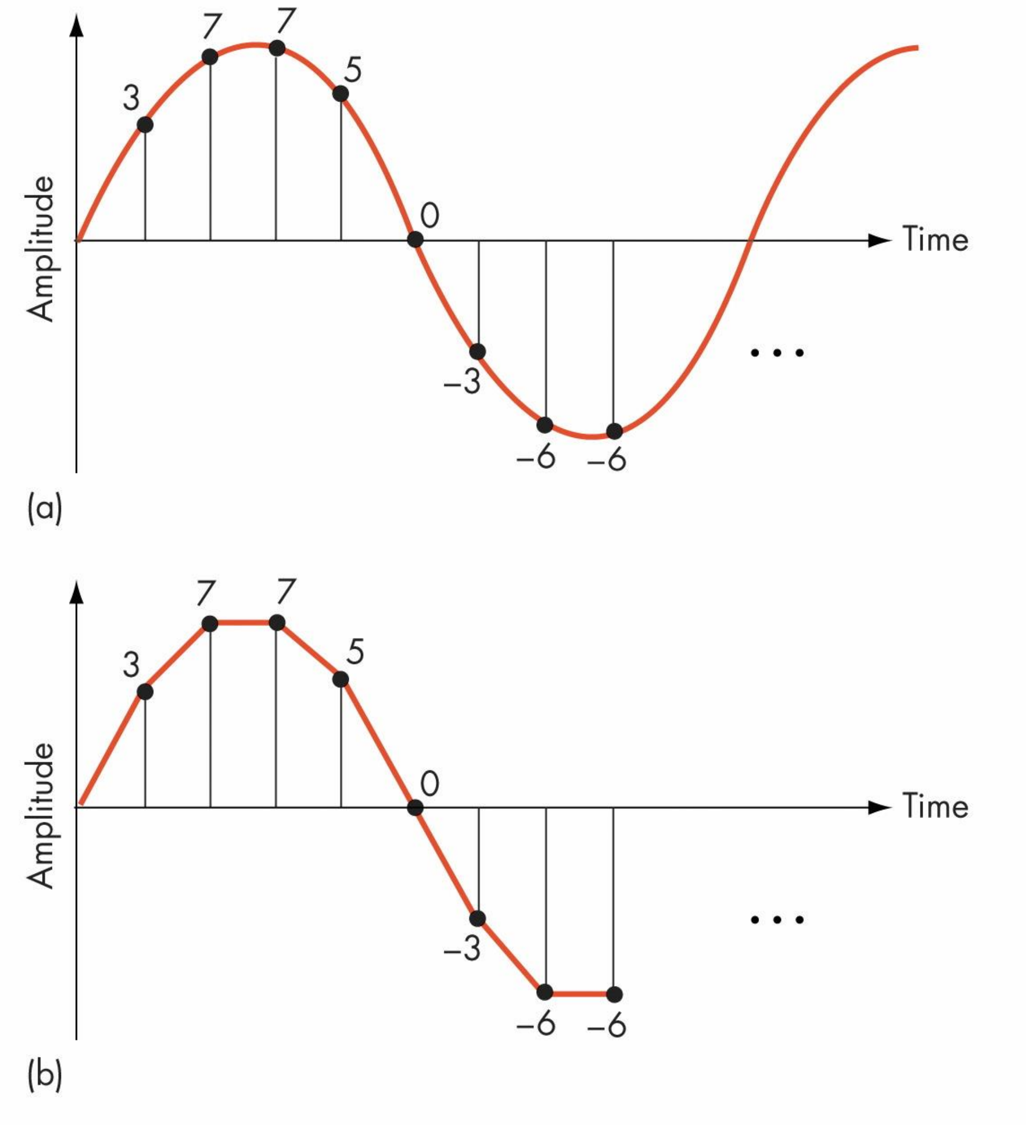
  - Use more bits for each sample value *(↑ bit depth)*

Figure 4.5

Digitization of an Analog Signal

(a) Sampling the Original Signal

(b) Recreating the Signal from the Sampled Values



Binary Representation of Sound (cont'd)

MP3 format discussed in text, AAC format here

AAC (Advanced Audio Coding) advantages over MP3

- Improved compression provides higher-quality results with smaller file sizes
- Higher resolution audio, yielding sampling rates up to 96 kHz
- Improved decoding efficiency, requiring less processing power for decode

- http://www.apple.com/quicktime/technologies/aac/

Binary Representation of Sound and Images (continued)

- Representing image data
  - Images are sampled by reading color and intensity values at even intervals across the image
  - Each sampled point is a pixel
  - Image quality depends on number of bits at each pixel

Binary Representation of Images (cont'd)

- Representing image data
  - Images are sampled by reading color and intensity values at even intervals across the image
  - Each sampled point is a pixel
  - Image quality depends on number of bits at each pixel
  - More image information: http://cat.xula.edu/tutorials/imaging/grayscale.php

The Reliability of Binary Representation

- Electronic devices are most reliable in a bistable environment
- Bistable environment
  - Distinguishing only two electronic states
    - Current flowing or not
    - Direction of flow
- Computers are bistable: hence binary representations

Binary Storage Devices

- Magnetic core
  - Historic device for computer memory
  - Tiny magnetized rings: flow of current sets the direction of magnetic field
  - Binary values 0 and 1 are represented using the direction of the magnetic field

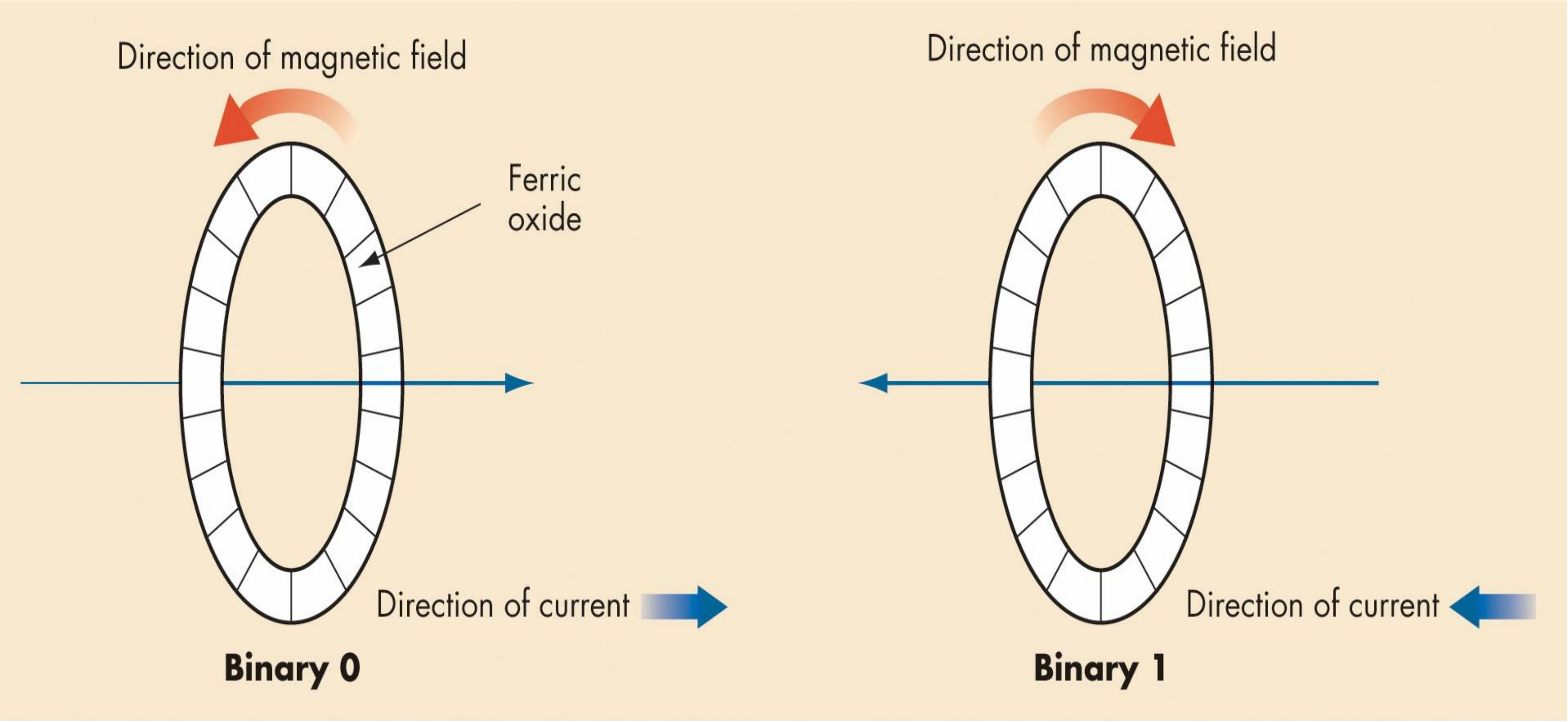


Figure 4.9
Using Magnetic Cores to Represent Binary Values

Binary Storage Devices (continued)

- Transistors
  - Solid-state switches: either permits or blocks current flow
  - A control input causes state change
  - Constructed from semiconductors

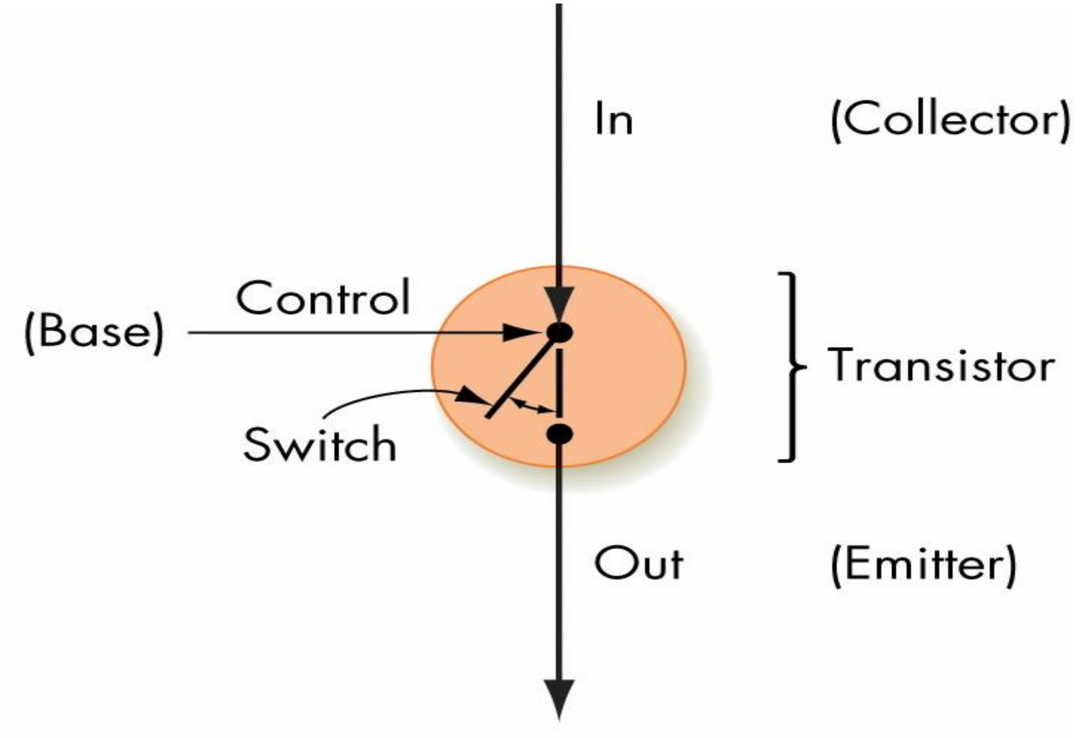


Figure 4.11

Simplified Model of a Transistor

Boolean Logic and Gates: Boolean Logic

- Boolean logic describes operations on true/false values
- True/false maps easily onto bistable environment
- Boolean logic operations on electronic signals may be built out of transistors and other electronic devices

Boolean Logic (continued)

- Boolean operations
  - a AND b
    - True only when a is true and b is true
  - a OR b
    - True when either a is true or b is true, or both are true
  - NOT a
    - True when a is false, and vice versa

Boolean Logic (continued)

- Boolean expressions
  - Constructed by combining together Boolean operations
    - Example: (a AND b) OR ((NOT b) AND (NOT a))
- Truth tables capture the output/value of a Boolean expression
  - A column for each input plus the output
  - A row for each combination of input values

Boolean Logic (continued)

- Example:

  (a AND b) OR ((NOT b) and (NOT a))

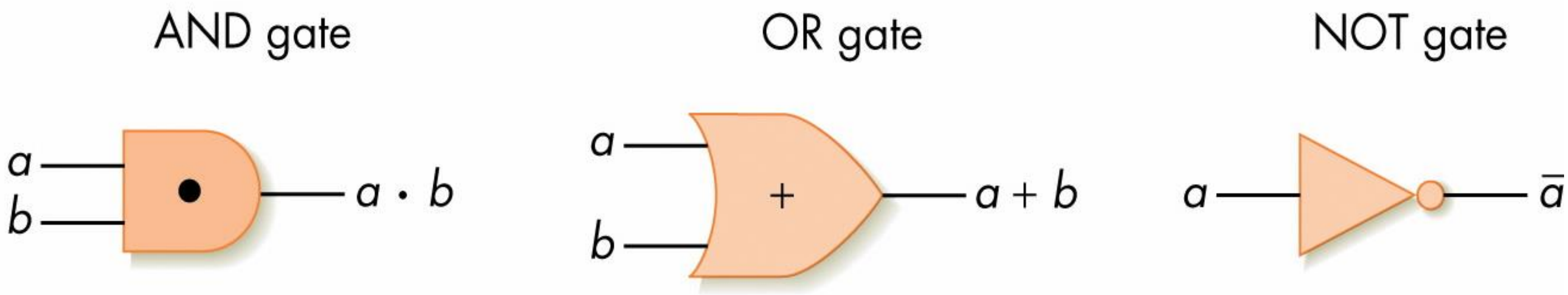
a	*b*	*Value*
0	0	1
0	1	0
1	0	0
1	1	1

Gates

- Gates
  - Hardware devices built from transistors to mimic Boolean logic
- AND gate
  - Two input lines, one output line
  - Outputs a 1 when both inputs are 1

Gates (continued)

- OR gate
  - Two input lines, one output line
  - Outputs a 1 when either input is 1
- NOT gate
  - One input line, one output line
  - Outputs a 1 when input is 0 and vice versa



a	b	a · b
0	0	0
0	1	0
1	0	0
1	1	1

a	b	a + b
0	0	0
0	1	1
1	0	1
1	1	1

a	ā
0	1
1	0

Figure 4.15

The Three Basic Gates and Their Symbols

Gates (continued)

- Abstraction in hardware design
  - Map hardware devices to Boolean logic
  - Design more complex devices in terms of logic, not electronics
  - Conversion from logic to hardware design may be automated

Building Computer Circuits: Introduction

- A circuit is a collection of logic gates:
  - Transforms a set of binary inputs into a set of binary outputs
  - Values of the outputs depend only on the current values of the inputs
- Combinational circuits have no cycles in them (no outputs feed back into their own inputs)

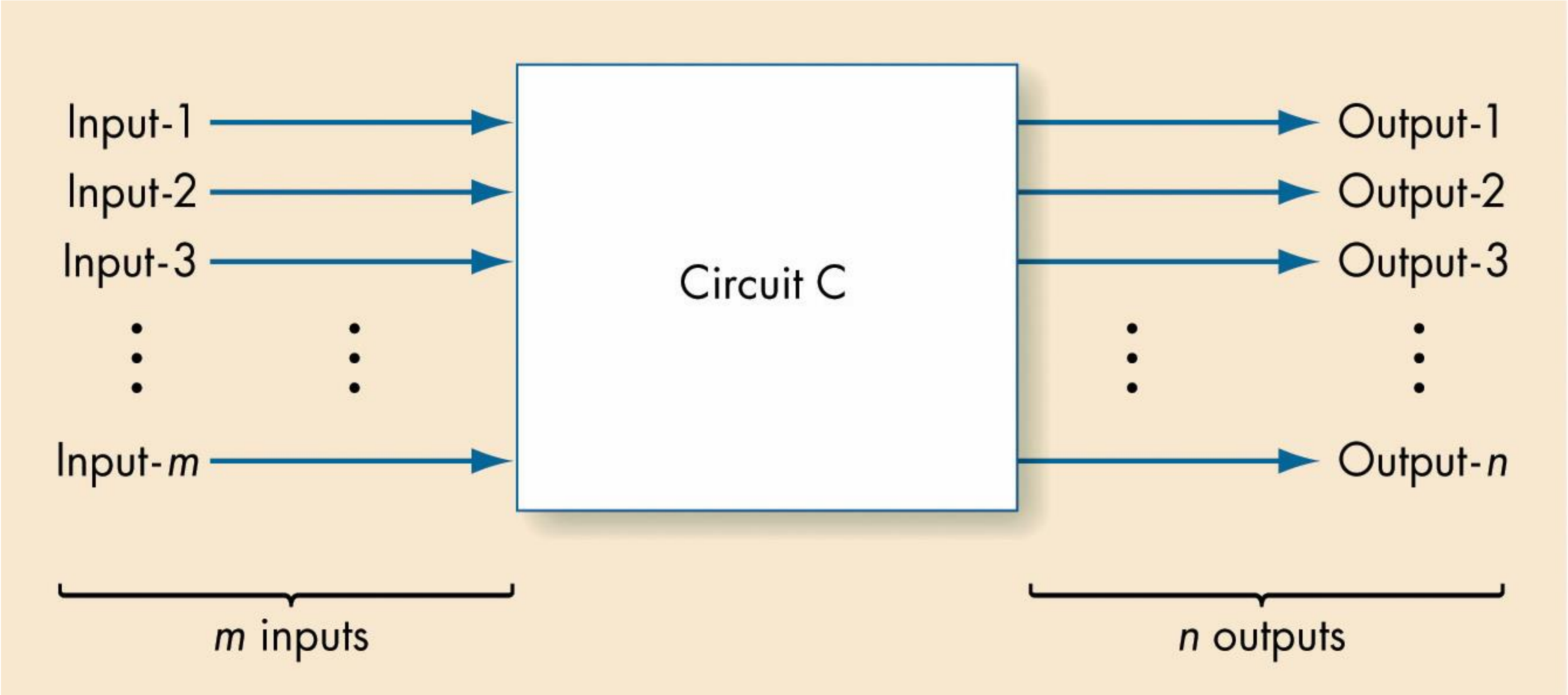


Figure 4.19

Diagram of a Typical Computer Circuit

A Circuit Construction Algorithm

- Sum-of-products algorithm is one way to design circuits:
  - Truth table to Boolean expression to gate layout

Sum-of-Products Algorithm for Constructing Circuits

1. Construct the truth table describing the behavior of the desired circuit
2. While there is still an output column in the truth table, do steps 3 through 6
3. Select an output column
4. Subexpression construction using AND and NOT gates
5. Subexpression combination using OR gates
6. Circuit diagram production
7. Done

Figure 4.21

The Sum-of-Products Circuit Construction Algorithm

A Circuit Construction Algorithm (continued)

- Sum-of-products algorithm
  - Truth table captures every input/output possible for circuit
  - Repeat process for each output line
    - Build a Boolean expression using AND and NOT for each 1 of the output line
    - Combine together all the expressions with ORs
    - Build circuit from whole Boolean expression

Examples Of Circuit Design And Construction

- Compare-for-equality circuit
- Addition circuit
- Both circuits can be built using the sum-of-products algorithm

A Compare-for-equality Circuit

- Compare-for-equality circuit
  - CE compares two unsigned binary integers for equality
  - Built by combining together 1-bit comparison circuits (1-CE)
  - Integers are equal if corresponding bits are equal (AND together 1-CD circuits for each pair of bits)

A Compare-for-equality Circuit (continued)

- 1-CE circuit truth table

a	*b*	*Output*
0	0	**1**
0	1	0
1	0	0
1	1	**1**

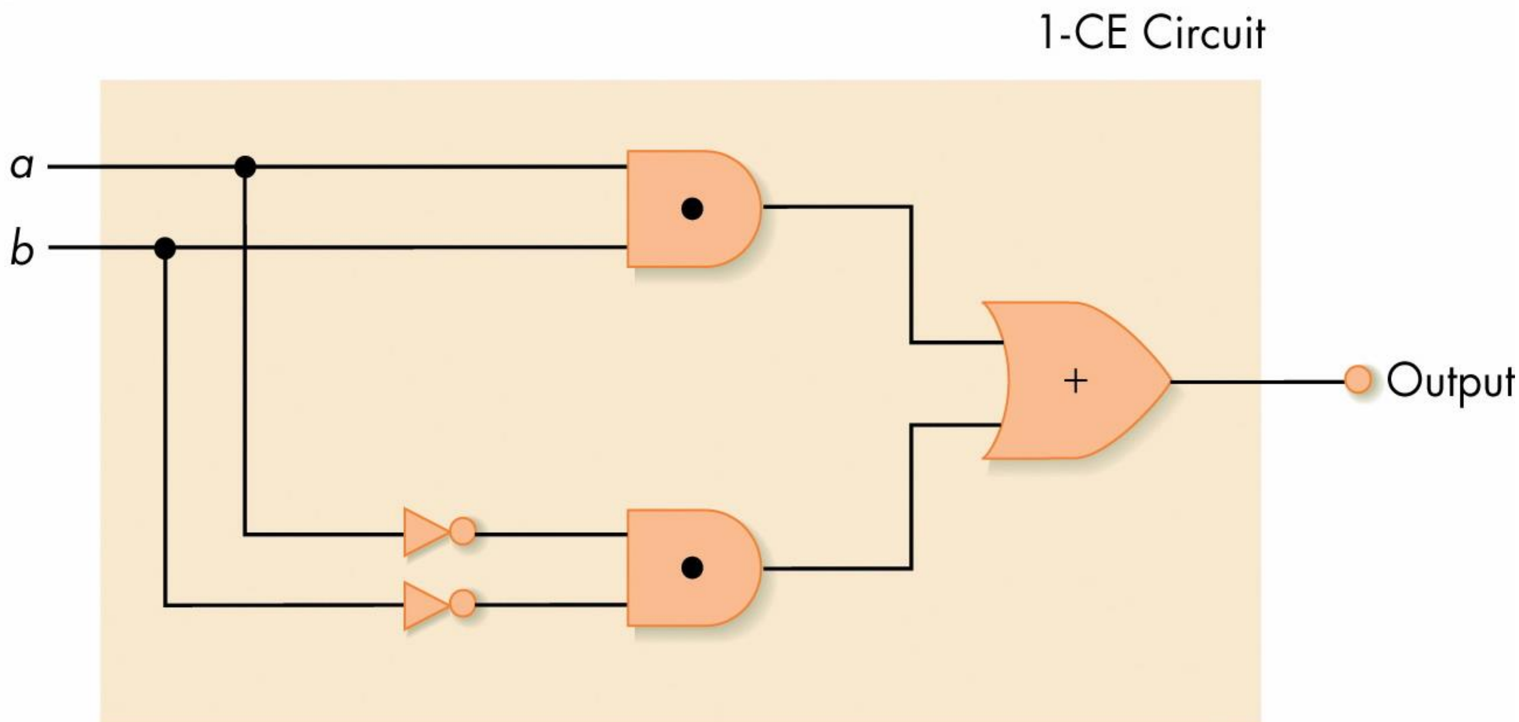
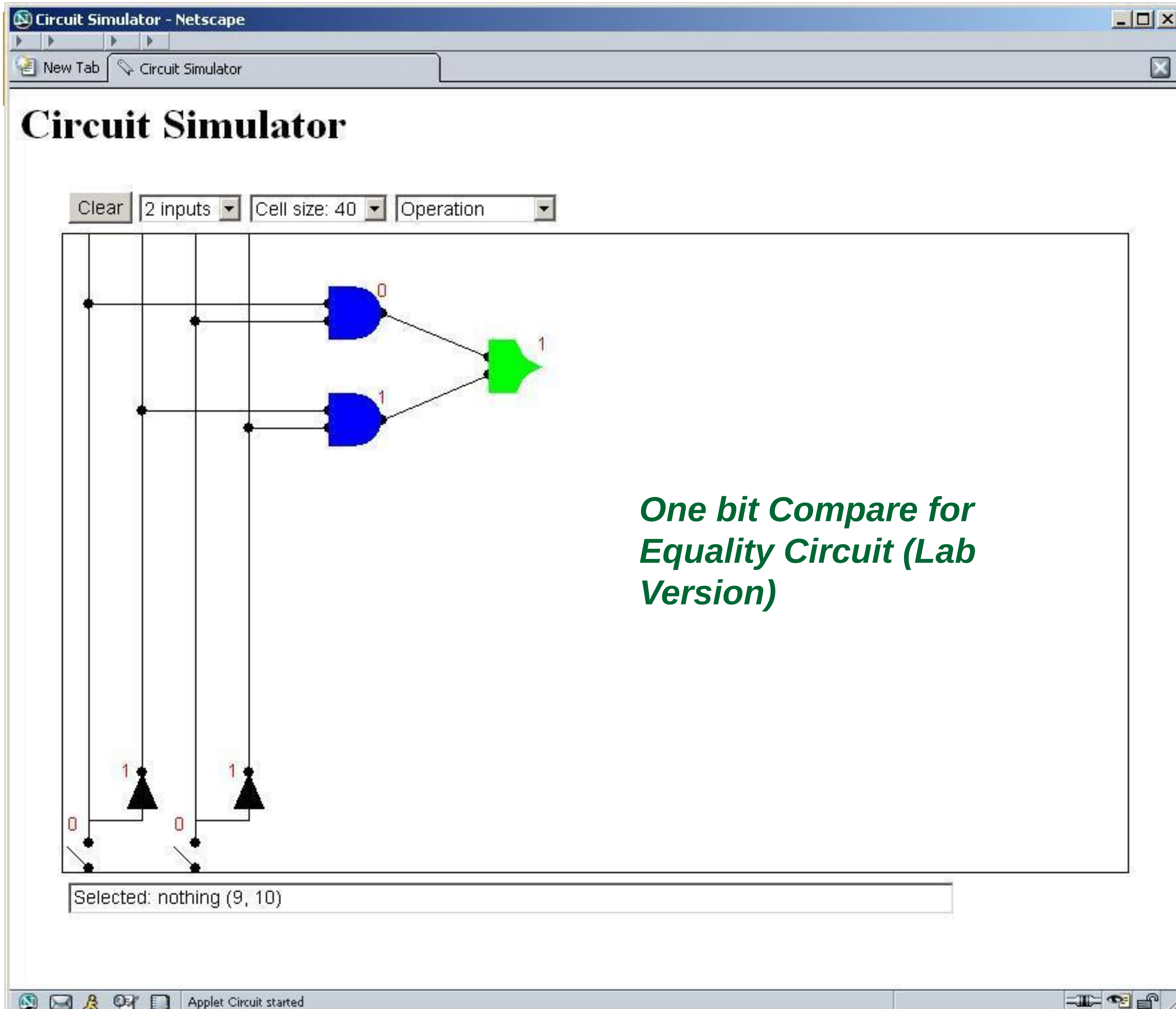


Figure 4.22

One-Bit Compare for Equality Circuit



One bit Compare for Equality Circuit (Lab Version)

A Compare-for-equality Circuit (continued)

- 1-CE Boolean expression
  - First case: (NOT a) AND (NOT b)
  - Second case: a AND b
  - Combined:

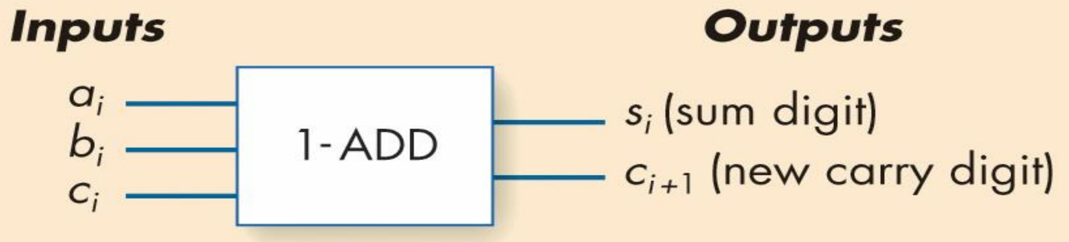
    ((NOT a) AND (NOT b)) OR (a AND b)

An Addition Circuit

- Addition circuit
  - Adds two unsigned binary integers, setting output bits and an overflow
  - Built from 1-bit adders (1-ADD)
  - Starting with rightmost bits, each pair produces
    - A value for that order
    - A carry bit for next place to the left

An Addition Circuit (continued)

- 1-ADD truth table
  - Input
    - One bit from each input integer
    - One carry bit (always zero for rightmost bit)
  - Output
    - One bit for output place value
    - One “carry” bit



Inputs			Outputs	
a_i	b_i	c_i	s_i	c_{i+1}
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

Figure 4.24

The 1-ADD Circuit and Truth Table

An Addition Circuit (continued)

- Building the full adder
  - Put rightmost bits into 1-ADD, with zero for the input carry
  - Send 1-ADD’s output value to output, and put its carry value as input to 1-ADD for next bits to left
  - Repeat process for all bits

Control Circuits

- Do not perform computations
- Choose order of operations or select among data values
- Major types of controls circuits
  - Multiplexors
    - Select one of inputs to send to output
  - Decoders
    - Sends a 1 on one output line, based on what input line indicates

Control Circuits (continued)

- Multiplexor form
  - 2^N regular input lines
  - N selector input lines
  - 1 output line
- Multiplexor purpose
  - Given a code number for some input, selects that input to pass along to its output
  - Used to choose the right input value to send to a computational circuit

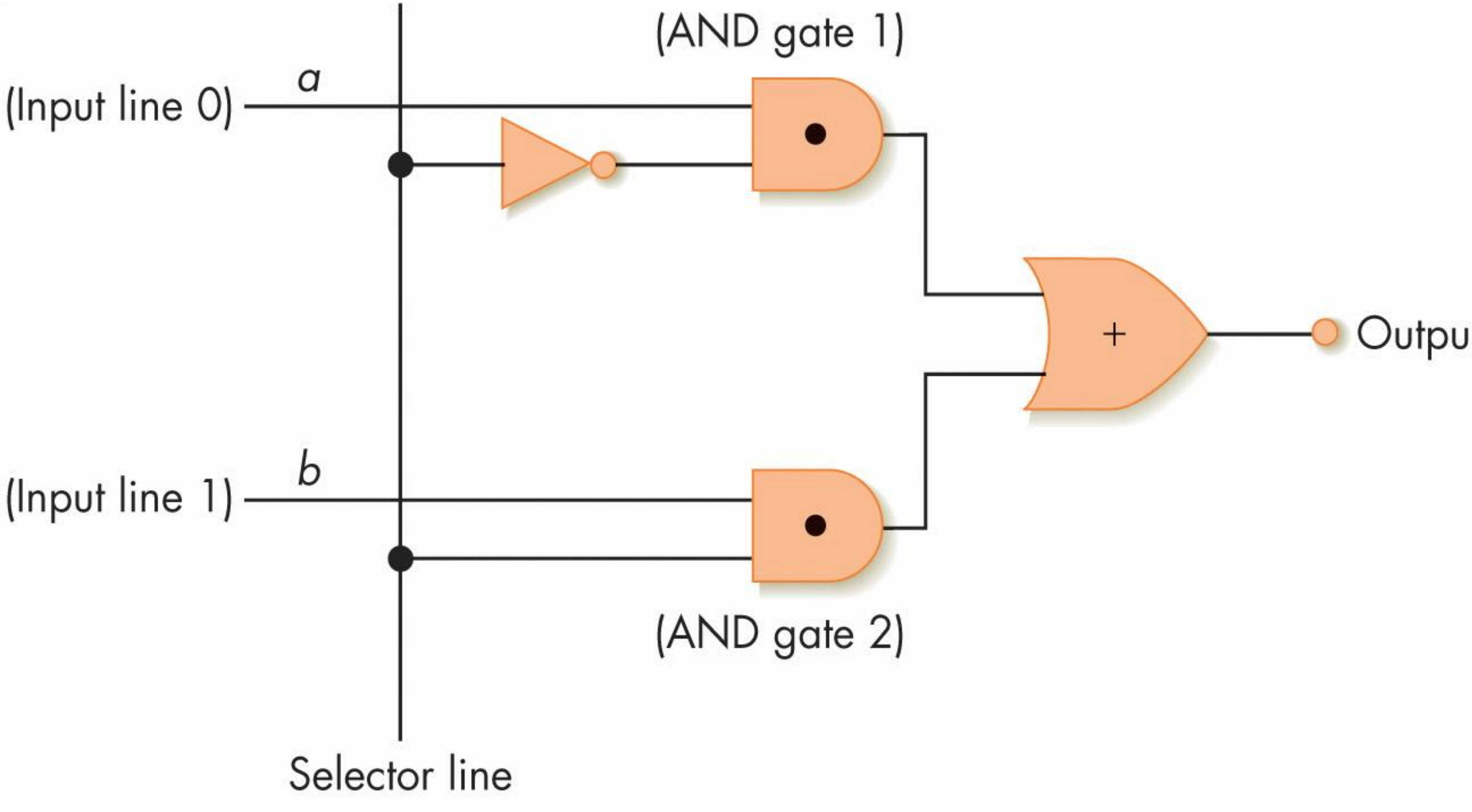


Figure 4.28

A Two-Input Multiplexor Circuit

Control Circuits (continued)

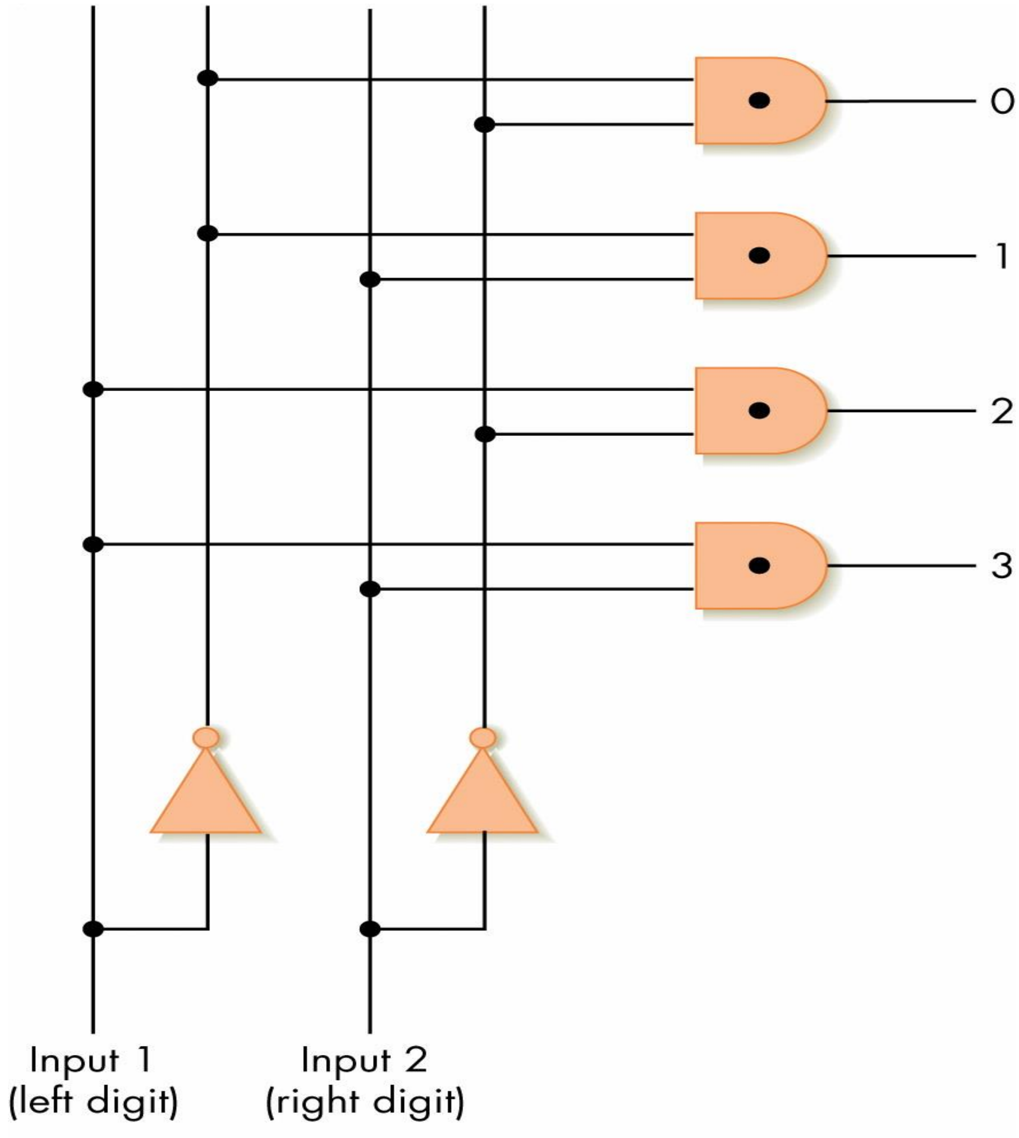
- Decoder
  - Form
    - N input lines
    - 2^N output lines
  - N input lines indicate a binary number, which is used to select one of the output lines
  - Selected output sends a 1, all others send 0

Control Circuits (continued)

- Decoder purpose
  - Given a number code for some operation, trigger just that operation to take place
  - Numbers might be codes for arithmetic: add, subtract, etc.
  - Decoder signals which operation takes place next

Figure 4.29

A 2-to-4 Decoder Circuit



Summary

- Digital computers use binary representations of data: numbers, text, multimedia
- Binary values create a bistable environment, making computers reliable
- Boolean logic maps easily onto electronic hardware
- Circuits are constructed using Boolean expressions as an abstraction
- Computational and control circuits may be built from Boolean gates