Digital Circuit Analysis and Design with Simulink® Modeling and Introduction to CPLDs and FPGAs

Second Edition

Steven T. Karris

3 - Bit Up/Down Counter - Simulation Time: 100 sec
Digital Circuit Analysis and Design with Simulink® Modeling and Introduction to CPLDs & FPGAs
Second Edition

This text includes the following chapters and appendices:
• Common Number Systems and Conversions • Operations in Binary, Octal, and Hexadecimal Systems • Sign Magnitude and Floating Point Arithmetic • Binary Codes • Fundamentals of Boolean Algebra • Minterms and Maxterms • Combinational Logic Circuits • Sequential Logic Circuits • Memory Devices • Advanced Arithmetic and Logic Operations • Introduction to Field Programmable Devices • Introduction to the ABEL Hardware Description Language • Introduction to VHDL • Introduction to Verilog • Introduction to Boundary-Scan Architecture

Each chapter contains numerous practical applications. This is a design-oriented text.

Steven T. Karris is the president and founder of Orchard Publications. His undergraduate and graduate degrees are from Christian Brothers University, Memphis, Tennessee, and from Florida Institute of Technology, Melbourne, Florida. He is a registered professional engineer in California and Florida. He has over 30 years of professional engineering experience in industry. In addition, he has over 30 years of teaching experience as an adjunct professor at several educational institutions, the most recent at UC Berkeley, CA.

Orchard Publications
Visit us on the Internet
www.orchardpublications.com
or email us: info@orchardpublications.com


$74.95 U.S.A.
# Table of Contents

## 1 Common Number Systems and Conversions

1.1 Decimal, Binary, Octal, and Hexadecimal Systems .................................................. 1–1
1.2 Binary, Octal, and Hexadecimal to Decimal Conversions ........................................ 1–3
1.3 Decimal to Binary, Octal, and Hexadecimal Conversions .......................................... 1–3
1.4 Binary–Octal–Hexadecimal Conversions ................................................................. 1–7
1.5 Summary .................................................................................................................. 1–9
1.6 Exercises ............................................................................................................... 1–11
1.7 Solutions to End–of–Chapter Exercises ................................................................. 1–12

## 2 Operations in Binary, Octal, and Hexadecimal Systems

2.1 Binary System Operations ....................................................................................... 2–1
2.2 Octal System Operations ....................................................................................... 2–2
2.3 Hexadecimal System Operations ........................................................................... 2–5
2.4 Complements of Numbers ...................................................................................... 2–6
   2.4.1 Tens–Complement ......................................................................................... 2–7
   2.4.2 Nines–Complement ..................................................................................... 2–7
   2.4.3 Twos–Complement ...................................................................................... 2–8
   2.4.4 Ones–Complement ...................................................................................... 2–9
2.5 Subtraction with Tens– and Twos–Complements .................................................... 2–10
2.6 Subtraction with Nines– and Ones–Complements ..................................................... 2–11
2.7 Summary ................................................................................................................ 2–14
2.8 Exercises ............................................................................................................... 2–16
2.9 Solutions to End–of–Chapter Exercises ................................................................. 2–18

MATLAB Computations: Pages 2–4, 2–6, 2–19 through 2–21, 2–23

## 3 Sign Magnitude and Floating Point Arithmetic

3.1 Signed Magnitude of Binary Numbers ..................................................................... 3–1
3.2 Floating Point Arithmetic ....................................................................................... 3–2
   3.2.1 The IEEE Single Precision Floating Point Arithmetic .................................. 3–3
   3.2.2 The IEEE Double Precision Floating Point Arithmetic ................................. 3–7
3.3 Summary ................................................................................................................ 3–9
3.4 Exercises ............................................................................................................... 3–10
3.5 Solutions to End–of–Chapter Exercises ................................................................. 3–11

MATLAB Computations: Pages 3–1 through 3–2, 3–11
4 Binary Codes
4.1 Encoding ................................................................. 4–1
  4.1.1 Binary Coded Decimal (BCD) .............................. 4–1
  4.1.2 The Excess–3 Code ............................................. 4–2
  4.1.3 The 2*421 Code ................................................. 4–3
  4.1.4 The Gray Code .................................................. 4–4
4.3 The Extended Binary Coded Decimal Interchange Code (EBCDIC) ................. 4–8
4.4 Parity Bits ............................................................... 4–8
4.5 Error Detecting and Correcting Codes ........................................ 4–9
4.6 Cyclic Codes ............................................................. 4–9
4.7 Summary ................................................................. 4–14
4.8 Exercises ................................................................. 4–16
4.9 Solutions to End–of–Chapter Exercises .................................... 4–17

5 Fundamentals of Boolean Algebra
5.1 Basic Logic Operations ............................................. 5–1
5.2 Fundamentals of Boolean Algebra .............................. 5–1
  5.2.1 Postulates .......................................................... 5–1
  5.2.2 Theorems ............................................................ 5–2
5.3 Truth Tables .............................................................. 5–3
5.4 Summary ................................................................. 5–5
5.5 Exercises ................................................................. 5–7
5.6 Solutions to End–of–Chapter Exercises ............................... 5–8

6 Minterms and Maxterms
6.1 Minterms ............................................................... 6–1
6.2 Maxterms ............................................................... 6–2
6.3 Conversion from One Standard Form to Another ......................... 6–3
6.4 Properties of Minterms and Maxterms .............................. 6–4
6.5 Summary ................................................................. 6–9
6.6 Exercises ................................................................. 6–10
6.7 Solutions to End–of–Chapter Exercises ............................... 6–12

7 Combinational Logic Circuits
7.1 Implementation of Logic Diagrams from Boolean Expressions .................... 7–1
7.2 Obtaining Boolean Expressions from Logic Diagrams ............................ 7–10
7.3 Input and Output Waveforms ........................................... 7–11
7.4 Karnaugh Maps .......................................................... 7–13
7.4.1 K-map of Two Variables ................................................................. 7–14
7.4.2 K-map of Three Variables ............................................................... 7–15
7.4.3 K-map of Four Variables ................................................................. 7–15
7.4.4 General Procedures for Using a K-map of n Squares ..................... 7–17
7.4.5 Don’t Care Conditions ................................................................. 7–20
7.5 Design of Common Logic Circuits .................................................... 7–21
  7.5.1 Parity Generators/Checkers ......................................................... 7–22
  7.5.2 Digital Encoders ........................................................................ 7–27
  7.5.3 Decimal–to–BCD Encoder ............................................................ 7–31
  7.5.4 Digital Decoders ........................................................................ 7–37
  7.5.5 Equality Comparators ................................................................. 7–40
  7.5.6 Multiplexers and Demultiplexers .................................................. 7–44
  7.5.7 Arithmetic Adder and Subtractor Logic Circuits ......................... 7–52
7.6 Summary ....................................................................................... 7–63
7.7 Exercises ....................................................................................... 7–65
7.8 Solutions to End–of–Chapter Exercises .......................................... 7–68

Simulink Modeling: Pages 7–3, 7–12, 7–25, 7–29 through 7–30, 7–47, 7–50, 7–56 through 7–60

8 Sequential Logic Circuits

8.1 Introduction to Sequential Circuits .................................................... 8–1
8.2 Set–Reset (SR) Flip Flop ................................................................. 8–1
8.3 Data (D) Flip Flop ........................................................................ 8–4
8.4 JK Flip Flop .................................................................................. 8–5
8.5 Toggle (T) Flip Flop ...................................................................... 8–6
8.6 Flip Flop Triggering ...................................................................... 8–7
8.7 Edge–Triggered Flip Flops ............................................................... 8–8
8.8 Master / Slave Flip Flops ............................................................... 8–8
8.9 Conversion from One Type of Flip Flop to Another ......................... 8–11
8.10 Analysis of Synchronous Sequential Circuits ................................ 8–13
8.11 Design of Synchronous Counters .................................................. 8–23
8.12 Registers ................................................................................... 8–28
8.13 Ring Counters .......................................................................... 8–34
8.14 Ring Oscillators ......................................................................... 8–37
8.15 Summary ................................................................................... 8–39
8.16 Exercises ................................................................................... 8–42
8.17 Solutions to End–of–Chapter Exercises ....................................... 8–45

Simulink Modeling: Pages 8–19, 8–37
9 Memory Devices

9.1 Random-Access Memory (RAM) ................................................................. 9–1
9.2 Read-Only Memory (ROM) ........................................................................... 9–3
9.3 Programmable Read-Only Memory (PROM) ............................................... 9–7
9.4 Erasable Programmable Read-Only Memory (EPROM) ................................ 9–9
9.5 Electrically-Erasable Programmable Read-Only Memory (EEPROM) .......... 9–10
9.6 Flash Memory ............................................................................................. 9–10
9.7 Memory Sticks ............................................................................................ 9–10
9.8 Cache Memory ........................................................................................... 9–11
9.9 Virtual Memory ......................................................................................... 9–11
9.10 Scratch Pad Memory ................................................................................. 9–12
9.11 The Simulink Memory Block ........................................................................ 9–12
9.12 Summary .................................................................................................. 9–14
9.13 Exercises .................................................................................................. 9–16
9.14 Solutions to End-of-Chapter Exercises ..................................................... 9–17

Simulink Modeling: Pages 9–6, 9–12

10 Advanced Arithmetic and Logic Operations ........................................... 10–1

10.1 Computers Defined .................................................................................. 10–1
10.2 Basic Digital Computer System Organization and Operation ................ 10–2
10.3 Parallel Adder ......................................................................................... 10–4
10.4 Serial Adder ........................................................................................... 10–5
10.5 Overflow Conditions ............................................................................... 10–6
10.6 High-Speed Addition and Subtraction ..................................................... 10–9
10.7 Binary Multiplication ............................................................................... 10–10
10.8 Binary Division ....................................................................................... 10–13
10.9 Logic Operations of the ALU ................................................................. 10–14
10.10 Other ALU functions ............................................................................ 10–15
10.11 Logic and Bit Operations with Simulink Blocks .................................... 10–16
10.11.1 The Logical Operator Block ............................................................... 10–16
10.11.2 The Relational Operator Block .......................................................... 10–16
10.11.3 The Interval Test Block .................................................................... 10–17
10.11.4 The Interval Test Dynamic Block ....................................................... 10–18
10.11.5 The Combinatorial Logic Block ......................................................... 10–19
10.11.6 The Compare to Zero Block ............................................................... 10–24
10.11.7 The Compare to Constant Block ....................................................... 10–25
10.11.8 The Bit Set Block .............................................................................. 10–26
10.11.9 The Clear Bit Block ........................................................................... 10–27
10.11.10 The Bitwise Operator Block ............................................................. 10–28
# 11 Introduction to Field Programmable Devices

11.1 Programmable Logic Arrays (PLAs) ................................................................. 11–1
11.2 Programmable Array Logic (PAL) ................................................................. 11–5
11.3 Complex Programmable Logic Devices (CPLDs) 11–6
   11.3.1 The Altera MAX 7000 Family of CPLDs ............................................. 11–7
   11.3.2 The AMD Mach Family of CPLDs ......................................................... 11–13
   11.3.3 The Lattice Family of CPLDs .............................................................. 11–15
   11.3.4 Cypress Flash370 Family of CPLDs ....................................................... 11–16
   11.3.5 Xilinx XC9500 Family of CPLDs .......................................................... 11–21
   11.3.6 CPLD Applications .............................................................................. 11–31
11.4 Field Programmable Gate Arrays (FPGAs) ..................................................... 11–37
   11.4.1 SRAM-Based FPGA Architecture ...................................................... 11–38
   11.4.2 Xilinx FPGAs .................................................................................... 11–38
   11.4.3 Atmel FPGAs ................................................................................... 11–41
   11.4.4 Altera FPGAs .................................................................................. 11–42
   11.4.5 Lattice FPGAs .................................................................................. 11–43
   11.4.6 Antifuse-Based FPGAs ................................................................. 11–44
   11.4.7 Actel FPGAs .................................................................................... 11–44
   11.4.8 QuickLogic FPGAs ................................................................. 11–44
   11.4.9 FPGA Block Configuration – Xilinx FPGA Resources ................. 11–50
   11.5 The CPLD versus FPGA Trade–Off ......................................................... 11–59
   11.6 What is Next ....................................................................................... 11–59
   11.7 Summary ............................................................................................. 11–62
   11.8 Exercises ............................................................................................ 11–64
   11.9 Solutions to End–of–Chapter Exercises ................................................... 11–66

# A Introduction to MATLAB®

A.1 Command Window ................................................................................... A–1
A.2 Roots of Polynomials ............................................................................. A–3
A.3 Polynomial Construction from Known Roots .......................................... A–4
A.4 Evaluation of a Polynomial at Specified Values ....................................... A–5
A.5 Rational Polynomials ............................................................................ A–8
A.6 Using MATLAB to Make Plots .............................................................. A–9
A.7 Subplots ................................................................. A–18
A.8 Multiplication, Division and Exponentiation .......................... A–19
A.9 Script and Function Files ............................................. A–26
A.10 Display Formats ..................................................... A–31

MATLAB Computations: Entire Appendix A

**Introduction to Simulink®**

B.1 Simulink and its Relation to MATLAB .......................... B–1
B.2 Simulink Demos ...................................................... B–20

Simulink Modeling: Entire Appendix B

**Introduction to ABEL Hardware Description Language**

C.1 Introduction ........................................................... C–1
C.2 Basic Structure of an ABEL Source File ......................... C–1
C.3 Declarations ........................................................... C–3
C.4 Numbers ................................................................. C–5
C.5 Directives ................................................................. C–6
   C.5.1 The @alternate Directive ........................................ C–6
   C.5.2 The @radix Directive ............................................. C–7
   C.5.3 The @standard Directive ....................................... C–7
C.6 Sets ............................................................................ C–7
   C.6.1 Indexing or Accessing a Set .................................... C–8
   C.6.2 Set Operations .................................................... C–9
C.7 Operators ..................................................................... C–11
   C.7.1 Logical Operators ............................................... C–11
   C.7.2 Arithmetic Operators ........................................... C–12
   C.7.3 Relational Operators .......................................... C–12
   C.7.4 Assignment Operators ......................................... C–13
   C.7.5 Operator Priorities .............................................. C–13
C.8 Logic Description ..................................................... C–14
   C.8.1 Equations ........................................................ C–14
   C.8.2 Truth Tables ...................................................... C–15
   C.8.3 State Diagram .................................................... C–18
   C.8.4 Dot Extensions ................................................ C–21
C.9 Test Vectors ............................................................. C–22
C.10 Property Statements .................................................. C–23
C.11 Active–Low Declarations .......................................... C–23
### Introduction to VHDL

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>D.1 Introduction</td>
<td>D-1</td>
</tr>
<tr>
<td>D.2 The VHDL Design Approach</td>
<td>D-1</td>
</tr>
<tr>
<td>D.3 VHDL as a Programming Language</td>
<td>D-3</td>
</tr>
<tr>
<td>D.3.1 Elements</td>
<td>D-3</td>
</tr>
<tr>
<td>D.3.2 Comments</td>
<td>D-4</td>
</tr>
<tr>
<td>D.3.3 Identifiers</td>
<td>D-4</td>
</tr>
<tr>
<td>D.3.4 Literal Numbers</td>
<td>D-4</td>
</tr>
<tr>
<td>D.3.5 Literal Characters</td>
<td>D-5</td>
</tr>
<tr>
<td>D.3.6 Literal Strings</td>
<td>D-5</td>
</tr>
<tr>
<td>D.3.7 Bit Strings</td>
<td>D-5</td>
</tr>
<tr>
<td>D.3.8 Data Types</td>
<td>D-5</td>
</tr>
<tr>
<td>D.3.9 Integer Types</td>
<td>D-6</td>
</tr>
<tr>
<td>D.3.10 Physical Types</td>
<td>D-7</td>
</tr>
<tr>
<td>D.3.11 Floating Point Types</td>
<td>D-8</td>
</tr>
<tr>
<td>D.3.12 Enumeration Types</td>
<td>D-9</td>
</tr>
<tr>
<td>D.3.13 Arrays</td>
<td>D-9</td>
</tr>
<tr>
<td>D.3.14 Records</td>
<td>D-11</td>
</tr>
<tr>
<td>D.3.15 Subtypes</td>
<td>D-11</td>
</tr>
<tr>
<td>D.3.16 Object Declarations</td>
<td>D-12</td>
</tr>
<tr>
<td>D.3.17 Attributes</td>
<td>D-13</td>
</tr>
<tr>
<td>D.3.18 Expressions and Operators</td>
<td>D-14</td>
</tr>
<tr>
<td>D.3.19 Sequential Statements</td>
<td>D-15</td>
</tr>
<tr>
<td>D.3.20 Variable Assignments</td>
<td>D-15</td>
</tr>
<tr>
<td>D.3.21 If Statement</td>
<td>D-16</td>
</tr>
<tr>
<td>D.3.22 Case Statement</td>
<td>D-16</td>
</tr>
<tr>
<td>D.3.23 Loop Statements</td>
<td>D-17</td>
</tr>
<tr>
<td>D.3.24 Null Statement</td>
<td>D-19</td>
</tr>
<tr>
<td>D.3.25 Assertions</td>
<td>D-19</td>
</tr>
<tr>
<td>D.3.26 Subprograms and Packages</td>
<td>D-20</td>
</tr>
<tr>
<td>D.3.27 Procedures and Functions</td>
<td>D-20</td>
</tr>
<tr>
<td>D.3.28 Overloading</td>
<td>D-23</td>
</tr>
<tr>
<td>D.3.29 Package and Package Body Declarations</td>
<td>D-24</td>
</tr>
<tr>
<td>D.3.30 Package Use and Name Visibility</td>
<td>D-26</td>
</tr>
<tr>
<td>D.4 Structural Description</td>
<td>D-26</td>
</tr>
<tr>
<td>D.4.1 Entity Declarations</td>
<td>D-26</td>
</tr>
<tr>
<td>D.4.2 Architecture Declarations</td>
<td>D-29</td>
</tr>
<tr>
<td>D.4.3 Signal Declarations</td>
<td>D-30</td>
</tr>
<tr>
<td>D.4.4 Blocks</td>
<td>D-30</td>
</tr>
<tr>
<td>D.4.5 Component Declarations</td>
<td>D-32</td>
</tr>
<tr>
<td>D.4.6 Component Instantiation</td>
<td>D-33</td>
</tr>
</tbody>
</table>
### D.5 Behavioral Description

- D.5.1 Signal Assignment ......................................................... D–33
- D.5.2 Process and the Wait Statement ........................................ D–34
- D.5.3 Concurrent Signal Assignment Statements ....................... D–35
- D.5.4 Conditional Signal Assignment ......................................... D–36
- D.5.5 Selected Signal Assignment ............................................. D–38

### D.6 Organization ................................................................. D–41

- D.6.1 Design Units and Libraries ............................................. D–42
- D.6.2 Configurations ............................................................... D–43

### D.7 Design Example ............................................................ D–47

### E

**Introduction to Verilog**

- E.1 Description ....................................................................... E–1
- E.2 Verilog Applications ................................................................. E–2
- E.3 The Verilog Programming Language ........................................ E–2
- E.4 Lexical Conventions ............................................................... E–6
- E.5 Program Structure ............................................................... E–7
- E.6 Data Types ........................................................................ E–9
  - E.6.1 Physical Data Types ....................................................... E–9
  - E.6.2 Abstract Data Types ...................................................... E–11
- E.7 Operators .......................................................................... E–11
  - E.7.1 Binary Arithmetic Operators ........................................ E–11
  - E.7.2 Unary Arithmetic Operators .......................................... E–12
  - E.7.3 Relational Operators ................................................... E–12
  - E.7.4 Logical Operators ........................................................ E–12
  - E.7.5 Bitwise Operators ........................................................ E–13
  - E.7.6 Unary Reduction Operators .......................................... E–13
  - E.7.7 Other Operators .......................................................... E–14
  - E.7.8 Operator Precedence .................................................... E–14
- E.8 Control Statements ............................................................ E–15
  - E.8.1 Selection Statements ..................................................... E–15
  - E.8.2 Repetition Statements .................................................. E–16
- E.9 Other Statements .............................................................. E–17
  - E.9.1 Parameter Statements .................................................. E–17
  - E.9.2 Continuous Assignment Statements ............................. E–17
  - E.9.3 Blocking Assignment Statements ................................ E–17
  - E.9.4 Non-Blocking Assignment Statements ....................... E–18
- E.10 System Tasks ................................................................. E–19
- E.11 Functions ...................................................................... E–21
- E.12 Timing Control ............................................................... E–22
  - E.12.1 Delay Control ............................................................ E–22

---

digital circuit analysis and design with simulink modeling
and introduction to CPLDs and FPGAs, second edition

copyright © orchard publications
F

Introduction to Boundary Scan Architecture

F.1 The IEEE Standard 1149.1 ................................................................. F–1
F.2 Introduction ................................................................................. F–1
F.3 Boundary Scan Applications ......................................................... F–3
F.4 Board with Boundary-Scan Components ...................................... F–4
F.5 Field Service Boundary-Scan Applications ................................. F–5

References

R–1

Index

IN–1
Implementation of Logic Diagrams from Boolean Expressions

b.

\[
\overline{A \cdot B} = \overline{A} + \overline{B} = A \oplus B
\]

Figure 7.5. Implementation of the XOR function with NAND gates only

This implementation requires only two IC SN7400 devices.

Figure 7.5A below shows a Simulink* model for the implementation of the logic circuit of Figure 7.5. The logic gates AND, OR, NAND, NOR, XOR, and NOT can be selected by first dragging the AND block from Simulink’s Logic and Bit Operations Library, and from Function Block Parameters window, we can choose any of these six gates, and we can specify the number of inputs and the icon shape (rectangular or distinctive). For this model, we selected the NAND gate with two inputs and the distinctive icon shape.

![Simulink model for the logic circuit of Figure 7.5](image)

Figure 7.5A - Simulink model for the logic circuit of Figure 7.5

Example 7.4

Implement the Boolean expression \( D = A(\overline{B}C + BC) + \overline{A}(\overline{B}C + BC) \) with XOR gates only.

Solution:

The Boolean expression \( B\overline{C} + B\overline{C} \) represents the XOR operation and the expression \( \overline{B}\overline{C} + BC \) represents the XNOR operation. Since the XNOR operation is the complement of the XOR operation, we make the substitution

\[
\overline{B\overline{C}} + BC = \overline{B\overline{C}} + BC
\]

* For an introduction to Simulink, please refer to Introduction to Simulink with Engineering Applications, ISBN 0-9744239-7-1. A brief introduction is included in Appendix B of this text.
7.4 Karnaugh Maps

A Karnaugh map, henceforth referred to as K-map, is a matrix of squares. In general, a Boolean expression with \( n \) variables can be represented by a K-map of \( 2^n \) squares where each square represents a row of an equivalent truth table. A K-map provides a very powerful method of reducing Boolean expressions to their simplest forms.
Figure 7.49 shows Motorola's MC74HC147 decimal–to–BCD encoder pin assignment.
Chapter 8

Sequential Logic Circuits

This chapter is an introduction to sequential logic circuits. These are logic circuits where the output(s) depend not only on the input(s) but also on previous states of the output(s). It begins with a discussion of the different types of flip flops, and continues with the analysis and design of binary counters, registers, ring counters, and ring oscillators.

8.1 Introduction to Sequential Circuits

In the previous chapter a combinational logic circuit was defined as a logic circuit whose output(s) are strictly a function of its inputs. A sequential circuit is a logic circuit whose output(s) is a function of its input(s) and also its internal state(s). The (internal) state of a sequential logic circuit is either a logic 0 or a logic 1, and because of its ability to maintain a state, it is also called a memory circuit. Generally, sequential circuits have two outputs one of which is the complement of the other and multivibrators fall into this category. Sequential circuits may or may not include logic gates.

Flip flops, also known as bistable multivibrators, are electronic circuits with two stable outputs one of which is the complement of the other. The outputs will change only when directed by an input command.

There are 4 types of flip flops and these are listed below.
1. Set–Reset (SR) or Latch
2. D Type (Data or Delay)
3. JK
4. T (Toggle)

8.2 Set–Reset (SR) Flip Flop

Figure 8.21(a) shows a basic Set–Reset (SR) flip flop constructed with two NAND gates, and Figure 8.21(b) shows the symbol for the SR flip flop where S stand for Set and R for Reset.

* For a thorough discussion on multivibrator circuits, please refer to Electronic Devices and Amplifier Circuits, ISBN 0–9709511-7-5
Example 8.3

Describe the operation of the sequential circuit of Figure 8.23. As indicated, x is an input that can assume the values of 0 or 1.

Solution:

As with the previous example, we begin with the construction of a state table denoted as Table 8.9. Since the inputs $J_1$, $K_1$, $J_2$, and $K_2$ are also dependent on the external input $x$, to facilitate
the circuit analysis we divide the Flip Flop Input section of the State Table into two subsections, one for the condition for \( x = 0 \), and the other for the condition \( x = 1 \).

**TABLE 8.9 State Table for Example 8.3**

<table>
<thead>
<tr>
<th>Present State</th>
<th>Flip Flop Inputs</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x=0 )</td>
<td>( x=1 )</td>
<td>( x=0 )</td>
</tr>
<tr>
<td>( Q_1 ) ( Q_2 ) ( Q_3 )</td>
<td>( J_1 ) ( K_1 ) ( J_2 ) ( K_2 ) ( J_3 ) ( K_3 )</td>
<td>( Q_1 ) ( Q_2 ) ( Q_3 )</td>
</tr>
<tr>
<td>0 0 0 0 0 1 1</td>
<td>1 1 1 1 1 1</td>
<td>0 0 1 1 1 1</td>
</tr>
<tr>
<td>0 0 1 0 0 1 1 1 1</td>
<td>0 0 0 0 1 1</td>
<td>0 1 0 0 0 0</td>
</tr>
<tr>
<td>0 1 0 0 0 0 1 1</td>
<td>0 0 1 1 1 1</td>
<td>0 1 1 0 0 1</td>
</tr>
<tr>
<td>0 1 1 1 1 1 1</td>
<td>0 0 0 0 1 1</td>
<td>1 0 0 0 1 0</td>
</tr>
<tr>
<td>1 0 0 0 0 1 1</td>
<td>1 1 1 1 1 1</td>
<td>1 0 0 1 1 1</td>
</tr>
<tr>
<td>1 0 1 0 0 1 1</td>
<td>0 0 0 0 1 1</td>
<td>1 1 0 0 1 0</td>
</tr>
<tr>
<td>1 1 0 0 0 0 1 1</td>
<td>0 0 1 1 1 1</td>
<td>1 1 1 1 0 1</td>
</tr>
<tr>
<td>1 1 1 1 1 1</td>
<td>0 0 0 0 1 1</td>
<td>0 0 0 1 1 0</td>
</tr>
</tbody>
</table>

The Next State section is also divided into two subsections, one corresponding to the condition \( x = 0 \) and the other to the condition \( x = 1 \). From the circuit of Figure 8.23, we obtain the following expressions:

\[
J_1 = K_1 = xQ_2Q_3 + \overline{xQ_2Q_3}
\]

\[
J_2 = K_2 = xQ_3 + \overline{xQ_3}
\]

\[
J_3 = K_3 = 1
\]

(8.2)

From the state table above, we derive two state diagrams, one for \( x = 0 \), and the other for \( x = 1 \) shown in Figure 8.24.

**Figure 8.24. State Diagrams for Example 8.3**

The state diagrams of Figure 8.24 reveal that the given circuit is an octal counter that counts up from zero to one,... to seven and then repeats whenever \( X = 0 \), and counts down from seven to...
 Analysis of Synchronous Sequential Circuits

 six... to zero when \( X = 1 \). Thus, the circuit is an up/down counter in which the counting sequence is controlled by the external input \( X \). The given circuit is also provided with Set Direct and Reset Direct commands and thus it has the capabilities of being initially preset or cleared before the counting sequence begins.

The Simulink model for the 3-bit up/down counter of Figure 8.23 is shown in Figure 8.23A below. The D Flip-Flop and Clock blocks are in the Simulink Extras Toolbox, Flip-Flops library, and the NAND and NOT gates are in the Logic and Bit Operations Library. The D Flip-Flop CLK (clock) inputs are Negative Edge Triggered. The Clock waveform and the D Flip-Flops output waveforms when the Manual Switch block is the Count up position, are shown in Figure 19.22.

![Figure 8.23A Simulink model for a 3-bit Up / Down binary counter](image)

![Figure 8.23B Waveforms for the model of Figure 8.23A](image)
Chapter 9

Memory Devices

This chapter is an introduction to computer memory devices. We discuss the random-access memory (RAM), read-only memory (ROM), row and column decoders, memory chip organization, static RAMs (SRAMs) dynamic RAMs (DRAMs), volatile, nonvolatile, programmable ROMs (PROMs), Erasable PROMs (EPROMs), Electrically Erasable PROMs (EEPROMs), flash memories, memory sticks, and cache memory.

9.1 Random-Access Memory (RAM)

Random access memory (RAM) is the best known form of computer memory. RAM is considered "random access" because we can access any memory cell directly if we know the row and column that intersect at that cell. In a typical RAM, the access time is independent of the location of the data within the memory device, in other words, the time required to retrieve data from any location within the memory device is always the same and can be accessed in any order. RAM is volatile, that is, data are lost when power is removed.

Another type of memory is the serial access memory (SAM). This type of memory stores data as a series of memory cells that can only be accessed sequentially (like a cassette tape). If the data is not in the current location, each memory cell is checked until the required data is found. SAM works very well for memory buffers, where the data is normally stored in the order in which it will be used such as the buffer memory on a video card. One advantage of SAM over RAM is that the former is nonvolatile memory, that is, stored data are retained even though power is removed.

Our subsequent discussion will be restricted to RAM devices.

In a RAM device, the access time, denoted as \( t_a \), is illustrated with the timing diagram of Figure 9.1.

![Figure 9.1. Access time defined](image-url)
Chapter 9  Memory Devices

\[ \cos 30^\circ = 0.0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} + \ldots + 1 \times 2^{-10} = 0.499 \approx 0.5 \]

The Simulink Lookup Tables Library consists of nine look-up table blocks. In this text we will describe only the Lookup Table block.*

Example 9.2
Using Simulink, create a model with a Lookup Table block configured to use a vector of input values given by \([1:10]\), and a vector of output values given by \(\log([1:10])\).†

Solution:
The model is shown in Figure 9.6 where the Display 1 block shows the true values of the natural log for the range \([1:10]\) and the Display 2 block shows the Lookup Table values for the same range of numbers. In the Constant block we have specified the range \([1:10]\) and the Lookup Table block has been configured with Vector of input values \([1:10]\), Vector output values \(\log([1:10])\), and Lookup method Interpolation – Extrapolation. The Math Function block is part of the Simulink’s Math Operations Library, and the Lookup Table block is part of the Simulink’s Lookup Tables Library.

† We recall that in MATLAB and Simulink \(\log(x)\) implies the natural logarithm of \(x\). The common (base 10) logarithm is denoted as \(\log_{10}(x)\).
Chapter 10

Advanced Arithmetic and Logic Operations

This chapter begins with an introduction to the basic components of a digital computer. It continues with a discussion of the basic microprocessor operations, and concludes with the description of more advanced arithmetic and logic operations.

10.1 Computers Defined

There are two classes of computers, analog, and digital. Analog computers are automatic computing devices that operate with electronic signals exhibiting continuous variations of physical quantities such as electrical voltages and currents, mechanical shaft rotations or displacements, and are used primarily to solve the differential equations that describe these time-varying signals. Results are normally displayed on oscilloscopes, spectrum analyzers, and pen recorders. A basic component in analog computers is a very versatile electronic device known as operational amplifier* or op amp for short.

Digital computers are automatic computing devices that operate with electronic signals exhibiting discrete variations, and are used for a variety of tasks such as word processing, arithmetic and logic operations, database construction, e-mail, etc. Digital computer operation is based on the binary numbering system which, as we learned in Chapter 1, employs two numbers only, zero (0) and one (1). Our subsequent discussion will be on digital computers. Two important characteristics of any digital computer are the ability to store information, and the speed of operation.

Digital computers are classified either as general purpose (stored program) or special purpose (dedicated) computer. A general purpose computer is one in which the sequence of instructions (program) is read into the computer via the input unit. A special purpose computer is one in which the sequence of operations are predetermined by the actual construction of the control circuitry.

Some important digital computer terminology

Computer Interfacing is the interconnection and synchronization of digital information transmission between the main computer and separate units such as the input/output devices often referred to as peripherals.

Computer Hardware are the electronic and mechanical parts and materials from which the computer is fabricated.

* For a thorough discussion on operational amplifiers, please refer to Electronic Devices and Amplifier Circuits, ISBN 0–9709511–7–5.
Chapter 10  Advanced Arithmetic and Logic Operations

TABLE 10.1 Truth table for a full adder

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The model is shown in Figure 10.11 where we have specified:

Constant blocks – Constant value: [0 0 0], [0 0 1], ... [1 1 1] in Constant blocks 1 through 8 respectively – Signal data types: boolean – Interpret vector parameters: check mark

Combinatorial Logic blocks (all) – Truth table: [0 0; 1 0; 1 0; 0 1; 1 0; 0 1; 0 1; 1 1] – Sample time: −1

Display blocks – Format: short

![Figure 10.11. Model for Example 10.17](image-url)
Chapter 11

Introduction to Field Programmable Devices

This chapter is an introduction to Field Programmable Devices (FPDs) also referred to as Programmable Logic Devices (PLDs). It begins with the description and applications of Programmable Logic Arrays (PLAs), continues with the description of Simple PLDs (SPLDs) and Complex PLDs (CPLDs), and concludes with the description of Field Programmable Gate Arrays (FPGAs).

11.1 Programmable Logic Arrays (PLAs)

A Programmable Logic Array (PLA) is a small Field Programmable Device (FPD) that contains two levels of logic, AND and OR, commonly referred to as AND–plane and OR–plane respectively. In concept, a PLA is similar to a ROM but does not provide full decoding of the variables, in other words, a PLA does not generate all the minterms as a ROM does. Of course, a ROM can be designed as a combinational circuit with any unused words as don’t care conditions, but the unused words would result in a poor design. Consider, for example, the conversion of a 16–bit code into an 8–bit code. In this case, we would have $2^{16} = 65536$ input combinations and only $2^8 = 256$ output combinations, and since we would only need 256 valid entries for the 16–bit code, we would have $65536 - 256 = 65280$ words wasted. With a PLA we can avoid this waste.

The size of a PLA is defined by the number of inputs, the number of product terms in the AND–plane, and the number of the sum terms in the OR–plane. Figure 11.1 shows the block diagram of a typical PLA. As shown, it consists of $n$ inputs, $m$ outputs, $k$ product terms, and $m$ sum terms. The group of the $k$ terms constitutes the AND–gate plan, and the group of $m$ terms constitutes the OR–gate plane. Links, shown as fuses, are inserted at all inputs and their complemented values to each of the AND gates. Links are also provided between the outputs of the AND gates and the inputs of the OR gates. Another set of links appears at the output lines that allows the output function to appear either in the AND–OR form or the AND–OR–Invert form. These forms are shown in Figure 11.2. The inverters at the output section of the PLA are tri–state devices.†

---

* We recall that a ROM contains $2^n$ words and $m$ bits per word where $n$ is the number of inputs in a decoder circuit.

† Tri–state devices have three outputs, logic 0, logic 1, and Hi–Z. For a detailed discussion on tri–state devices, please refer to Electronic Devices and Amplifier Circuits, ISBN 0–9709511–7–5.
**Programmable Interconnect Array**

The *Programmable Interconnect Array* (PIA) is a global bus with programmable paths that connect any signal source to any destination on the device. Logic is routed between LABs via the PIA. All MAX 7000 dedicated inputs, I/O pins, and macrocell outputs feed the PIA which makes the signals available throughout the entire device. Figure 11.12 shows how the PIA signals are routed into the LAB. As shown, an EEPROM cell controls one input to a 2-input AND gate, which selects a PIA signal to drive a LAB. Whereas in other PLDs the routing delays of channel-based routing schemes are cumulative, variable, and path-dependent, the MAX 7000 PIA has a fixed delay. The PTA thus eliminates skew between signals and makes timing performance easy to predict.
11.4.4 Altera FPGAs

Altera’s Stratix II device family is the newest high-density FPGA family from Altera. Stratix II devices feature a brand new, unique architecture comprised of an innovative logic structure, allowing designers to pack more functionality into less space, thereby reducing developments costs. Combined with the 90-nm process technology, the Stratix II family delivers on average 50 percent faster logic performance, more than twice the logic capacity, and is typically 40 percent lower cost than first generation Stratix devices allowing designers to leverage the advantages of programmable technology in a much wider set of applications. The architecture of the Stratix II FPGA is shown in Figure 11.42.

**Figure 11.42. Architecture of Altera’s Stratix II SRAM–based FPGA**

The *adaptive logic module* (ALM) shown in Figure 11.42 is the fundamental innovation in the Stratix II architecture. An ALM is the basic building block of logic in the Stratix II architecture. It provides advanced features with efficient logic utilization and fast performance. Each ALM contains two *adaptive look–up tables* (ALUTs). With up to eight inputs to the combinational logic block, one ALM can implement up to two independent functions each of varying widths. One
through the Xilinx University Program and Workshops. In some cases, Xilinx will allow one to
download the software necessary for completing some of the labs at home. An example in design-
ing a multiply–accumulate (MAC) circuit is presented below.

As shown in Figure 11.69, a MAC can be implemented as a single engine or multiple engines.

![Single and multiple-engine MAC implementation](image)

Figure 11.69. Single- and multiple-engine MAC implementation

As shown in Figure 11.69, in the single engine the MAC unit is time-shared, and using a 256–
tap FIR filter we need to perform 256 multiply and accumulate operations per data sample, and
we get one output every 256 clock cycles. However, with the multiple engines configuration
using again a 256–tap FIR filter we can perform 256 multiply and accumulate operations per data
sample, but we get one output every clock cycle.

One of the downloadable Xilinx files available at university.xilinx.com pertains to the creation of
a 12–bit × 8–bit MAC using VHDL. A block diagram of this 12–bit × 8–bit MAC is shown in Fig-
ure 11.70.

![12-bit x 8-bit MAC](image)

Figure 11.70. 12–bit × 8–bit MAC

The multiply–accumulate operations are as follows:

\[
\begin{align*}
1 \times 1 &= 1 \\
(1 \times 1) + (2 \times 2) &= 5 \\
(1 \times 1) + (2 \times 2) + (3 \times 3) &= 14
\end{align*}
\]

and so on

The VHDL script for the 12 × 8 MAC is listed in Section B.27, Appendix B.

The VHDL folder contains the file accum whose content is as follows:
In the example above, a multiplication (*) sign between 4 and j was not necessary because the complex number consists of numerical constants. However, if the imaginary part is a function or variable such as \( \cos(x) \), we must use the multiplication sign, that is, we must type \( \cos(x) \times j \) or \( j \times \cos(x) \).

### A.2 Roots of Polynomials

In MATLAB, a polynomial is expressed as a row vector of the form \([a_n \ a_{n-1} \ a_2 \ a_1 \ a_0]\). The elements \(a_i\) of this vector are the coefficients of the polynomial in descending order. We must include terms whose coefficients are zero.

We can find the roots of any polynomial with the `roots(p)` function where \(p\) is a row vector containing the polynomial coefficients in descending order.

#### Example A.1

Find the roots of the polynomial

\[
p_1(x) = x^4 - 10x^3 + 35x^2 - 50x + 24
\]  \hspace{1cm} (A.1)

**Solution:**

The roots are found with the following two statements. We have denoted the polynomial as \(p_1\), and the roots as `roots_\ p1`.

```
p1=[1 -10 35 -50 24] % Specify the coefficients of p1(x)
p1 = 
   1   -10   35   -50   24

roots_\ p1=roots(p1) % Find the roots of p1(x)
roots_\ p1 =
   4.0000
   3.0000
   2.0000
   1.0000
```

We observe that MATLAB displays the polynomial coefficients as a row vector, and the roots as a column vector.
Appendix B

Introduction to Simulink®

This appendix is a brief introduction to Simulink. This author feels that it is best introduce Simulink with a few examples. Some familiarity with MATLAB is essential in understanding Simulink, and for this purpose, Appendix A is included as an introduction to MATLAB.

B.1 Simulink and its Relation to MATLAB

The MATLAB® and Simulink® environments are integrated into one entity, and thus we can analyze, simulate, and revise our models in either environment at any point. We invoke Simulink from within MATLAB. We will introduce Simulink with a few illustrated examples.

Example B.1

For the circuit of Figure B.1, the initial conditions are \( i_L(0^-) = 0 \), and \( v_c(0^-) = 0.5 \text{ V} \). We will compute \( v_c(t) \).

\[ i = i_L = i_C = C \frac{dv_c}{dt} \]  \hspace{1cm} (B.1)

and by Kirchoff’s voltage law (KVL),

\[ R i_L + L \frac{di_L}{dt} + v_C = u_0(t) \]  \hspace{1cm} (B.2)

Substitution of (B.1) into (B.2) yields

\[ \frac{dv_c}{dt} + \frac{1}{C} v_c(t) + \frac{1}{L} R i_L = u_0(t) \]
The initial conditions \([x_1 \ x_2]'\) are specified in MATLAB's Command prompt as
\[x_1=0; \ x_2=0.5;\]

As before, to start the simulation we click clicking on the ▶ icon, and to see the output waveform, we double click on the Scope block, and then clicking on the Autoscale icon, we obtain the waveform shown in Figure B.16.

![Scope block](image)

Figure B.16. The waveform for the function \(v_C(t)\) for Example B.1 with the State−Space block.

The state−space block is the best choice when we need to display the output waveform of three or more variables as illustrated by the following example.

**Example B.2**

A fourth−order network is described by the differential equation

\[
\frac{d^4 y}{dt^4} + a_3 \frac{d^3 y}{dt^3} + a_2 \frac{d^2 y}{dt^2} + a_1 \frac{dy}{dt} + a_0 y(t) = u(t)
\]  

where \(y(t)\) is the output representing the voltage or current of the network, and \(u(t)\) is any input, and the initial conditions are \(y(0) = y'(0) = y''(0) = y'''(0) = 0\).

a. We will express (B.27) as a set of state equations.
Appendix C

Introduction to ABEL Hardware Description Language

This appendix provides a brief overview of the Advanced Boolean Equation Language (ABEL) which is an industry–standard Hardware Description Language (HDL) used in Programmable Logic Devices (PLDs).

C.1 Introduction

The Advanced Boolean Equation Language (ABEL) is an easy–to–understand and use programming language that allows the user to describe the function of logic circuits. It is now an industry–standard allows you to enter behavior–like descriptions of a logic circuit. Developed by Data I/O Corporation for Programmable Logic Devices (PLDs), ABEL is now an industry–standard Hardware Description Language (HDL). An example of how ABEL is used, was given in an example in Chapter 11. There are other hardware description languages such as the VHSIC* Hardware Description Language (VHDL) and Verilog. ABEL is a simpler language than VHDL and Verilog. ABEL can be used to describe the behavior of a system in a variety of forms, including logic equations, truth tables, and state diagrams using C–like statements. The ABEL compiler allows designs to be simulated and implemented into PLDs such as PALs, CPLDs and FPGAs.

We will not discuss ABEL in detail. We will present a brief overview of some of the features and syntax of ABEL. For more advanced features, the reader may refer to an ABEL manual or the Xilinx on–line documentation.

C.2 Basic Structure of an ABEL Source File

An ABEL source file consists of the following elements:

- Header: including Module, Options, and Title
- Declarations: Pin, Constant, Node, Sets, States, Library
- Logic Descriptions: Equations, Truth_table, State_diagram
- Test Vectors: Test_vectors
- End

Keywords (words recognized by ABEL as commands, e.g. goto, if, then, module, etc.) are not case sensitive. User–supplied names and labels (identifiers) can be uppercase, lowercase or mixed–case, but are case–sensitive, for instance output1 is different from Output1.

* VHSIC is an acronym for Very High Speed Integrated Circuits.
Appendix C  Introduction to ABEL Hardware Description Language

TRUTH_TABLE ([ A, B, CIN] -> [SUM, COUT])

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 ];

The truth table above can be simplified if we define the set

IN = [A,B];

and

OUT = [SUM, COUT]

Then, the truth table is written as

TRUTH_TABLE ( IN -> OUT)

0 -> 0;
1 -> 2;
2 -> 2;
3 -> 1;
4 -> 2;
5 -> 1;
6 -> 1;
7 -> 3;

Example C.11

The output of an XOR gate is connected to an ON/OFF switch. Write the truth table.

Solution:

Let us denote the switch as SW where the OFF position is represented as logic 0, and the ON position as logic 1, and the inputs to the XOR gate as A and B. Using a short notation as in Example C.10, the truth table is written as follows where .X. indicates a don’t care.

TRUTH_TABLE ([ SW, A, B ] -> OUT)

[ 0, .X., .X. ] -> [ .X. ];
[ 1, 0, 0 ] -> 0;
[ 1, 0, 1 ] -> 1;
[ 1, 1, 0 ] -> 1;
[ 1, 1, 1 ] -> 0;
of the entity (design) by specifying its external interfaces which include a description of its ports (inputs and outputs). The entity declaration in VHDL is as follows:

```vhdl
entity count3 is
  generic (prop_delay: Time := 10 ns);
  port (clock: J0, K0, J1, K1, J2, K2: in bit;
        Q0, Q1, Q2: out bit);
end count3;
```

The first line (entity) specifies that the entity `count3` has three inputs and three outputs; these are logic 1 or logic 0 (True or False) values. The second line (generic) is optional. If used, it defines the generic constant `prop_delay` which represents the propagation delay. If omitted, the default value 10ns is assumed. The third and fourth lines define the port clause, that is, external interfaces (inputs and outputs). The last line (end) signifies the end of entity `count3`.

The entity `count3` is the first part of 3-bit counter design. The second part of the description of the `count3` design is a description of how the design operates. This is defined by the architecture declaration. The following is an example of an architecture declaration for the entity `count3`.

Figure D.1 shows the structure of `count3` where \( T_2 = Q_1Q_2, T_1 = Q_2, \) and \( T_0 = 1. \)

\[ T_2 = T_1 = T_0 \]

\[ Q_2, Q_1, Q_0 \]

```
Q2 Q1 Q0

CLK

A B C

\( T_2 \)

\( T_1 \)

\( T_0 \)

\( Q_2, Q_1, Q_0 \)

\( FF_2, FF_1, FF_0 \)

D.1. Structure of count3
```

The description of the entity `count3` as an architectural body is shown below. However, we must remember that there may be more than one architectural body corresponding to a single

* This 3-bit counter is a modification of Example 8.3, Chapter 8, where, for convenience, we have replaced the JK-type flip flops with T-type flip flops and the circuit has been simplified to an up-counter, that is, it counts from 000 to 001 to 010...111 resets to 000 and repeats the count. We have assumed that this circuit is implemented with TTL devices where unconnected inputs behave as logic 1. Accordingly, the input to flip flop \( T_0 \) is not shown as an external input.
Appendix E  Introduction to Verilog

E.2 Verilog Applications

As we’ve learned in Chapter 11, present-day digital systems are highly complex and are available in CPLD and FPGA technology. Accordingly, digital circuit design engineers must implement their designs with hardware description languages (HDLs) prior to fabrication. The most prominent modern HDLs in industry are Verilog and VHDL.

Verilog allows hardware designers to express their design with behavioral constructs, putting aside the details of implementation to a later stage of design in the design. Even though a behavioral construct is a high level description of a digital system, it is still a precise notation.

E.3 The Verilog Programming Language

Verilog describes a digital system as a set of modules. Comments begin with the symbol “//” and terminate at the end of the line or by /* to */ across several lines. Keywords, e.g., module, are reserved and in all lower case letters. The language is case sensitive, meaning upper and lower case letters are different. Spaces are important. A simple example is given below.

Example E.1
Using Verilog, design the behavioral model of a 2-input NAND gate.

Solution:

```
// Behavioral model of a 2-input NAND gate
// Designer’s name and date
module NAND (in1, in2, out);
input in1, in2;
output out;
// Continuous assign statement follows
assign out = ~(in1 | in2);
endmodule
```

In Example E.1 above, the lines with ‘//’ are comment lines, and module is the behavioral specification of module NAND with ports (inputs and output) in1 and in1 and out. The continuous assignment assign monitors continuously for changes to variables in its right hand side and whenever that happens the right hand side is re-evaluated and the result immediately propagated to the left hand side (out). The continuous assignment statement is used to model com-
The principles of interconnect test using boundary-scan are illustrated in Figure F.2, where two boundary-scan compliant devices, U1 and U2 interface with each other.
Index

Symbols

% (percent) symbol in MATLAB A-2
@alternate Directive in ABEL C-6
@radix Directive in ABEL C-7
@Radix in ABEL C-5
@standard Directive in ABEL C-7
2*421 code 4-3

A

ABEL C-1
abs(z) MATLAB function A-24
abstract data types in Verilog E-11
accessing a set in ABEL C-8
accumulator register 10-4
ACK 4-6,
Actel FPGAs 11-44
active-low declarations in ABEL C-23
adaptive logic module 11-42
adaptive look-up tables 11-42
Advanced Boolean Equation Language (ABEL) C-1
Algebraic Constrain blocks B-18
ALM 11-42
alphanumerics 4-1
Altera FPGAs 11-42
Altera MAX 7000 Family of CPLDs 11-7
ALUT 11-42
AMD Mach Family of CPLDs 11-13
American National Standards Institute 4-5
American Standard Code for Information Interchange (ASCII) 4-13
analog computer 10-1
analog-to-digital conversion 10-2
AND operation 5-1
angle(z) MATLAB function A-24
ANSI 4-5
Antifuse-based FPGA 11-38
Antifuse-Based FPGAs 11-44
architectural declaration in VHDL D-29
architecture bodies in VHDL D-39
architecture declaration in VHDL D-2
argument in Verilog E-10
Arithmetic / Logic Unit 10-3
arithmetic operators in ABEL C-12
array aggregate in VHDL D-10
array in VHDL D-9
ascending ranges in VHDL D-5
assertion statement in VHDL D-19
assignment operators in ABEL C-13
assignment statement in VHDL D-15
associative laws 5-2
asynchronous binary ripple counter 8-20
asynchronous flip-flop 8-2
asynchronous transfer 8-28
AT40KAL 11-41
Atmel FPGAs 11-41
attributes in VHDL D-13
autoscale icon B-12
Axcelerator 11-44
axioms 5-1
axis MATLAB command A-17

B

base of a number 1-1
base 2 number 1-1
base 8 number 1-2
base 10 number 1-1, 1-3
base 16 number 1-2
base2dec(s,b) MATLAB function 2-4
basic input/output system 10-4
base2dec function in MATLAB A-7
BCL 4-1
behavioral constructs in Verilog E-2
behavioral in VHDL D-1
behavioral specification in Verilog E-7
behavioural description in VHDL D-1
BEL 4-6
binary number 1-1
binary arithmetic operators in Verilog E-11
Binary Coded Decimal 4-1
binary-to-decimal number conversion 1-3
binary-to-hexadecimal number conversion 1-8
binary-to-octal decoder 7-37
binary-to-octal number conversion 1-7
BIOS 9-10, 10-4
 bistable multivibrators 8-1
Bit Clear block in Simulink 10-27
Bit Set block in Simulink 10-26
bits in VHDL D-5
Bitwise Operator block in Simulink 10-28
block in VHDL D-1, D-30
Boolean algebra 5-1
Bose-Chauduri codes 4-10
boundary-scan board F-4
Boundary-Scan Description Language F-1
Boundary-scan F-1
box in VHDL D-10
box MATLAB command A-11
BS 4-5
BSIDL F-1
buried macrocell 11-14

C

cache memory 9-11
cAN 4-7
canonical form 6-5
carry bit flip flop 10-6
CarryConnect 11-49
Case statement in ABEL C-20
cell sensor 9-10
central processing unit 10-3
central switch matrix 11-14
CGROM 9-7
character generator 9-7
characteristic of a number 3-2
characteristic table 8-2, 10-22
checksum 4-10
chip 10-4
CLB 11-39
clear MATLAB command A-2
clocked circuit 8-14
cluster 11-46
coefficients of a number 1-1
column vector in MATLAB A-19
combinational logic circuit 7-1
combinational logic circuit 8-1
combinational operator in ABEL C-13
Combinatorial Logic block in Simulink 10-19
command screen in MATLAB A-1
command window in MATLAB A-1
commas in MATLAB A-7
Commonly Used Blocks in Simulink B-7
commutative laws 5-2
Compare To Constant block in Simulink 10-25
Compare To Zero block in Simulink 10-24
complementation 5-1
complements of numbers 2-6
complex conjugate of a number A-4
complex numbers A-2
Complex Programmable Logic Devices (CPLDs) 11-6
computer hardware 10-1
computer interfacing 10-1
computer software 10-2
Concat block in Xilinx 11-57
continuous signal assignment in VHDL D-38
conditional signal assignment in VHDL D-38
configurable logic block 11-39
Configuration Parameters in Simulink B-12
conjunction 5-1
constant in VHDL D-12
Contents Pane in Simulink B-7
continuous assignment statements in Verilog E-17
counter codes 4-5
counter gate 9-9
control statements in Verilog E-15
control unit 10-3
cov MATLAB function A-6
Convert block in Xilinx 11-57, 11-58
CPLD 11-6
CPLD Applications 11-31
CPLD versus FPGA Trade-Off 11-59
CPU 10-3
CR 4-7
CRC code 4-10
Cyclic codes 4-9
cyclic redundancy check code 4-10
Cypress Flash370 Family of CPLDs 11-16

D

D flip flop 8-4
data flip flop 8-4
data points in MATLAB A-14
data types in Verilog E-9
DC1 4-7
join control in Verilog E-23
Joint Test Action Group F-1
JTAG F-1

K
Karnaugh maps 7-13
K-maps 7-13

L
L1 cache 9-11
L2 cache 9-11
L3 cache 9-11
LAB 11-7
Lattice 3000, 4000, 5000 devices 11-15
Lattice Family of CPLDs 11-15
Lattice ispMACH 11-15
Lattice pLSI, ispLSI devices 11-15
Level 1 cache 9-11
Level 2 cache 9-11
Level 3 cache 9-11
lexical conventions E-6
LF (Line Feed) 4-7
library unit in VHDL D-42
lins = in MATLAB A-27
linear factor A-9
Link Library in Simulink 7-58
linspace MATLAB command A-14
literary characters in VHDL D-5
literal numbers in VHDL D-4
literal strings in VHDL D-5
ln (natural log) in MATLAB A-12
load command 8-29
log(x) MATLAB function A-12
log10(x) MATLAB function A-12
log2(x) MATLAB function A-12
logic block 11-17
logic description in ABEL C-14
logic element 11-38
logic operations 5-1
logical name in VHDL D-42
Logical Operator block in Simulink10-16
logical operators in ABEL C-11
logical operators in Verilog E-12
logical product 5-1
logical sum 5-1
loglog(x,y) MATLAB command A-12
look up table 9-5, 11-38
look-ahead carry 10-9
loop statements in VHDL D-17
LUT (Look Up Table) 11-38

M
MAC circuit 11-60
Mach 1 device 11-13
Mach 2 device 11-13
Mach 3 device 11-13
Mach 5 device 11-13
macrocell arrays 11-7
mantissa of a number 3-2
mask-programmable 11-2
master/slave flip flop 8-8
Math Operations blocks in Simulink B-10
MATLAB Demos A-2
MATLAB’s Editor/Debugger A-1
matrix multiplication in MATLAB A-19
maxterms 6-2
memory circuit 8-1
memory stick 9-10
memory stick duo media 9-10
memory unit 10-3
mesh(x,y,z) MATLAB command A-17
meshgrid(x,y) MATLAB command A-17
message polynomial 4-10
m-file in MATLAB A-1, A-26
microcomputer 10-3
microprocessor 10-3
minterms 6-1
module in ABEL C-3
module in Verilog E-2
module in VHDL D-1
module-2 addition 4-11
modulo-2 subtraction 4-11
Muller-C element in VHDL D-38
multiplexer 7-44
Multiplexing 7-44
multiply-accumulate 11-60

N
NAK 4-7
NaN in MATLAB A-27
natural subtype in VHDL D-10
negation 5-1
negation laws 5-2
negative pulse 8-7
negedge in Verilog E-22
nines complement 2-7
node in ABEL C-5
non-blocking assignment statements in Verilog E-18
non-standard form 6-7
normalized 3-2
NOT operation 5-1
NUL 4-5
null statement in VHDL D-19
number conversions 1-3
numbers in ABEL C-5
numbers in Verilog E-6

O
object in VHDL D-12
octal number 1-2
octal numbers 2-2
octal system operations 2-2
octal-to-binary encoder 7-27
octal-to-binary number conversion 1-7
octal-to-decimal number conversion 1-3
ones complement 2-9
operating system 10-2
operator priorities in ABEL C-13
operators in ABEL C-11
operators in Verilog E-11
optimized reconfigurable cell array 11-43
OR operation 5-1
ORCA device 11-43
other declarations in ABEL C-4
output unit 10-3
OutReg (Output Register) 11-48
overflow condition 10-6, 11-56
overloading in VHDL D-23

P
package in VHDL D-20, D-24
page file 9-12
PAL 11-5
parallel expanders 11-9, 11-10
parallel transfer 8-29
parity bit 4-8
pass gate 11-41
pause 10-3
peripherals 10-1
PFU 11-43
physical data types in Verilog E-9
physical type in VHDL D-7
PIA 11-8, 11-11
PIM 11-17
pin declarations in ABEL C-4
PLA 11-1
Place and Route Report 11-61
PLD C-1
plot MATLAB command A-9
plot3 MATLAB command A-16
plot
MATLAB command A-9

quadratic factor A-9