

经 典 原 版 书 库

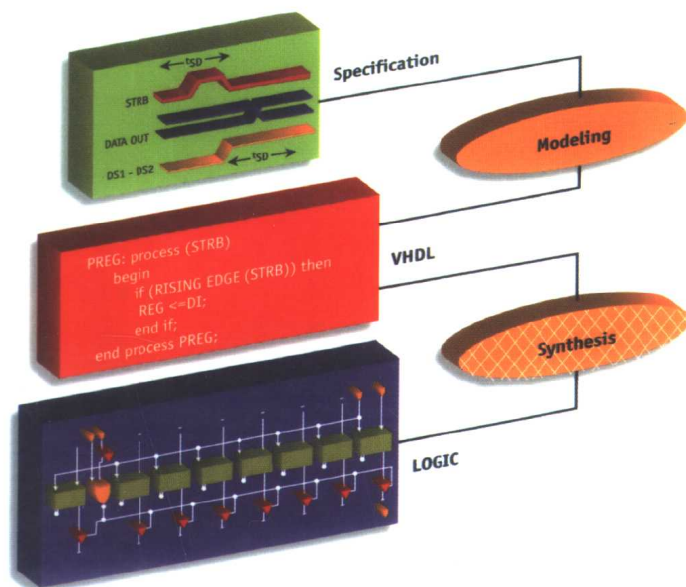
VHDL 设计、表示和综合

(英文版·第2版)

VHDL Design

Representation and Synthesis

SECOND EDITION



James R. Armstrong

(美) James R. Armstrong 著
F. Gail Gray



机械工业出版社
China Machine Press

Prentice
Hall

VHDL设计、表示和综合 (英文版·第2版)

VHDL Design Representation and Synthesis (Second Edition)

全面介绍基于综合的VHDL设计

本书清晰、全面地阐述了当今硬件设计的主流方法——使用硬件描述语言进行综合，和当今主流的综合工具方面的最新知识。两位权威作者从结构化设计的介绍开始，深入讲述了VHDL语言及其关键结构，然后循序渐进地讲解建模过程，使用了不同抽象层次的各种例子，并阐述了用于提高仿真效率和综合工具兼容性的各种技巧。

主要内容

- **设计工具**：编辑器，仿真器，检查器，分析器，优化器和综合器
- **VHDL**：主要结构，句法描述，源文件，数据类型，数据对象，语句和高级特性
- **基本的VHDL建模技术**：传输和时延，并发性，组合逻辑，时序逻辑，系统原语等等
- **将VHDL与设计流程结合**：从算法层次的可执行规范，到门级或单元级的实现PLD，门阵列，FPGA（使用Xilinx工具）和标准单元（使用Synopsys工具）建模

作者简介

James R. Armstrong 博士是弗吉尼亚理工大学 (Virginia Tech) 电机与计算机工程系的教授，VHDL方面的国际权威。IEEE标准化委员会的创始成员以及IEEE VHDL标准化工作组和设计自动化工作组的主席。



F. Gail Gray 博士是弗吉尼亚理工大学 (Virginia Tech) 电机与计算机工程系的教授。拥有密歇根大学计算机、信息和控制工程博士学位。



配套光盘含有书中所有
VHDL模型，模型测试基准，
以及项目序列，等等。

封面设计
陈子平

ISBN 7-111-11676-3



9 787111 116769



网上购书：www.china-pub.com

北京市西城区百万庄南街1号 100037

读者服务热线：(010) 68995259 8006100280 (北京地区)

总编信箱：chiefeditor@hzbook.com

限中国大陆地区销售

ISBN 7-111-11676-3/TP · 2814

定价：69.00元（附光盘）

经 典 原 版 书 库

VHDL

设计、表示和综合

(英文版·第2版)

VHDL Design Representation
and Synthesis

(Second Edition)

(美) James R. Armstrong 著
F. Gail Gray



B1262890



机械工业出版社
China Machine Press

English reprint edition copyright © 2003 by Pearson Education North Asia Limited and China Machine Press.

Original English language title: *VHDL Design Representation and Synthesis, Second Edition* by James R. Armstrong and F. Gail Gray, Copyright 2000.

All rights reserved.

Published by arrangement with the original publisher, Pearson Education, Inc., publishing as Prentice Hall PTR, Inc.

For sale and distribution only in the People's Republic of China exclusively (except Taiwan, Hong Kong SAR and Macau SAR).

本书英文影印版由Prentice Hall PTR, Inc授权机械工业出版社独家出版。未经出版者书面许可,不得以任何方式复制或抄袭本书内容。

仅限于中华人民共和国大陆地区(不包括中国台湾地区和香港、澳门特别行政区)销售。

本书封面贴有Pearson Education培生教育出版集团激光防伪标签,无标签者不得销售。

版权所有,侵权必究。

本书版权登记号:图字:01-2003-1006

图书在版编目(CIP)数据

VHDL设计、表示和综合(英文版·第2版)/(美)阿姆斯特朗(Armstrong, J.R.), (美)格雷(Gray, F.G.)著. —北京:机械工业出版社, 2003 3
(经典原版书库)

书名原文: *VHDL Design Representation and Synthesis, Second Edition*
ISBN 7-111-11676-3

I V… II ①阿… ②格… III 硬件描述语言, VHDL—英文 IV.TP312

中国版本图书馆CIP数据核字(2003)第009745号

机械工业出版社(北京市西城区百万庄大街22号 邮政编码 100037)

责任编辑:华章

北京市密云县印刷厂印刷·新华书店北京发行所发行

2003年3月第1版第1次印刷

787mm×1092mm 1/16·42.25印张

印数:0 001-3 000册

定价:69.00元(附光盘)

凡购本书,如有倒页、脱页、缺页,由本社发行部调换

Preface

The purpose of this book is to integrate hardware description languages into the digital design process at all levels of abstraction. There are two main steps in this process: (1) development of a hardware description language model and (2) synthesis of the model into an ASIC logic circuit or FPGAs. In teaching this process, we use VHDL, the VHSIC Hardware Description Language. VHDL, whose development began in 1983 under DOD sponsorship, was further developed by the IEEE and released as IEEE Standard 1076 in 1987. Further improvements were incorporated since then and the language was re-released as an updated standard in 1993. Since that time, VHDL has evolved into a de facto industry standard for hardware description languages. In the opinion of the authors, it has the most comprehensive set of modeling constructs available in any hardware description language. For these reasons, VHDL was chosen as the base language for this book. We explore the language in an in-depth, unified manner.

Most books currently on the market that treat hardware description languages, particularly VHDL, are either: (1) language texts that cover the VHDL language thoroughly, but do not show how to integrate the language into the digital design process, or (2) logic design books that primarily use VHDL models as simulation tools to validate designs that are produced in the classical manner. This book fully integrates VHDL into the design process starting with a high-level executable model that provides an unambiguous, executable version of the specification, and concluding with a gate-level implementation.

In this book, synthesis is viewed as a multistep process, beginning with an English description which is transformed first into VHDL and then from VHDL into a circuit schematic. We discuss synthesis from two viewpoints: 1) the mappings: emphasis is placed on understanding the relationship between VHDL language constructs and the implied logic circuit. A full chapter is devoted to correct modeling style for synthesis; 2) the tools: we illustrate the synthesis process using two very popular tool sets, the Synopsys Design Analyzer and Compiler (for ASICs) and the Xilinx Foundation Series (for FPGAs). Since ASICs and FPGAs are the targets,

a chapter is devoted to these technologies. The book also contains a chapter illustrating the complete top-down design process from specification to logic synthesis.

This book is written for three main educational purposes: (1) for a second course in logic design for undergraduate students in Electrical Engineering, Computer Engineering, and Computer Science; (2) for a graduate course dealing with hardware description languages and other design aids; and (3) for practicing engineers who wish to learn about design with hardware description languages. Thus the assumed background for the book is (1) a basic course in computer organization and logic design and (2) some knowledge of high-level languages, such as C, C++, or JAVA.

The authors use the text in a course, which is the second course in a logic design sequence. The students are either juniors in Computer Engineering, for whom the course is required, or Electrical Engineering seniors, for whom the course is an elective. In this semester length course we cover Chapters 1, 2, 3, 4, 5, 9, 10, and 11. The emphasis is on developing VHDL models in a conservative algorithmic style that can be synthesized. To support this in the laboratory, we use a PC version of ViewLogic, Inc.'s Workview for VHDL modeling and simulation and schematic capture. Xilinx software and XS40/XTEND boards are used for FPGA synthesis. We also employ System View from Elanix to provide for high-level design of digital filters. Workstation-based Synopsys tools are used for ASIC synthesis. All students in our department have their own PCs, so the use of a PC-based system such as Workview has been effective in being able to serve the large number of students we normally teach in our second digital design course. For this same reason, we use telnet and dc_shell scripts for Synopsys synthesis. Typical assignments include:

1. An introductory assignment to familiarize students with Workview's VHDL modeling, simulation and schematic capture environment.
2. An assignment to develop and simulate a single VHDL behavioral model.
3. An assignment to develop a model of a counter, or some similar circuit. VHDL behavioral models are developed for counter flip-flops and gates, and the schematic capture capability of Workview is used to construct the structural model of the counter.
4. An assignment to translate a system description is first translated into a VHDL behavioral model which is simulated. This is typically a state machine such as an interface protocol, a vending machine, or a traffic light controller.
5. An introductory tutorial to the Xilinx Foundation Series Software.
6. An assignment to implement a small circuit in both Synopsys ASIC logic and FPGAs and compare speed of execution.
7. A fairly complicated FPGA project such as a booth multiplier, calculator, small processor, digital filter, or graphics display. For the filter, the codec on the XSTEND board is used for A/D and D/A conversion. The Xilinx filter code is developed using System View. The graphics display displays a pattern stored in RAM on a VGA monitor.

If used for a graduate course, the entire book can be covered in one semester. In such a course, one can cover the broad range of constructs in the language and examine in detail the language semantics for both simulation and synthesis. In our graduate course at Virginia Tech, we synthesize with Synopsys and validate synthesized models. We study ways to control the synthesis to achieve optimum circuits in a delay or area sense. High-level modeling tools such as

Express VHDL, SPW, and System View are also covered. A comparison is done between VHDL and Verilog.

For this course, the student's laboratory assignments include:

1. An assignment to develop and simulate a single VHDL behavioral model.
2. An assignment to develop a model of a counter or some similar circuit. VHDL behavioral models are developed for counter flip-flops and gates, and then a VHDL structural model is developed for the whole system.
3. An assignment involving complex data types, e.g., using array aggregates and record types to implement a tabular representation of a finite state machine.
4. A system modeling assignment that involves the use of bus resolution and bus protocols. This system employs the IEEE 9 valued logic system. Examples include the URISC processor system in Chapter 6 or a histogram construction system for image processing.
5. An assignment where a model is written, simulated, and synthesized using both VHDL and Verilog and comparison's made
6. A semester project where the students model a system of their choice. One can choose projects, which stretch the language, i.e., involve applications that are not typical, such as modeling parallel processing systems or modeling systems which are not digital.

The book contains hundreds of VHDL models and code fragments. All code has been analyzed, and simulated, and synthesized (where required), using the Synopsys VHDL system. The only exception to this is the VHDL 93 code. In addition, the text contains over 300 homework problems with a wide range of difficulty. Types of problems include short answer questions, simple design problems, complex system design problems involving design, modeling, and simulation, and problems that require a study of a design or design tool issue. Some problems in this latter category would make good thesis projects!

Accompanying the book is a CD-ROM. On the CD are: 1) source files for all VHDL code in the book, 2) a set of projects accompanied by supporting data command files, and 3) packages to support common design paradigms. Problem and project solutions and Power Point lecture slides are available to instructors who adopt our text for classroom use.

Writing a book of this nature is a large undertaking. In doing so we have received the help and assistance of a number of individuals and organizations. We would like to thank:

1. Viewlogic, Synopsys, Inc., and Xilinx, for their support in providing us with the VHDL software to check out the VHDL code in the book and for use in our courses on hardware description languages.
2. Dave Barton of Intermetrics for his review of the manuscript.
3. Our production editor at Prentice Hall, Vincent Janoski.
4. Our book editor at Prentice Hall, Bernard Goodwin, for his enthusiastic support of the project.

We would also like to thank our wives, Marie and Caryl, for their encouragement and support in spite of the long hours we spent in front of our computers and their tolerance of laptops accompanying us on trips and vacations over a period of two years.

Contents

Preface	xi
<hr/>	
1 Structured Design Concepts	1
<hr/>	
1.1 The Abstraction Hierarchy	1
1.2 Textual vs. Pictorial Representations	5
1.3 Types of Behavioral Descriptions	7
1.4 Design Process	8
1.5 Structural Design Decomposition	9
1.6 The Digital Design Space	11
2 Design Tools	17
<hr/>	
2.1 CAD Tool Taxonomy	17
2.1.1 Editors	18
2.1.2 Simulators	18
2.1.3 Checkers and Analyzers	18
2.1.4 Optimizers and Synthesizers	18
2.1.5 Cad Systems	18
2.2 Schematic Editors	19
2.3 Simulators	22
2.3.1 Simulation Cycle	24

2.3.2 Simulator Organization	25
2.3.3 Language Scheduling Mechanism	25
2.3.4 Simulation Efficiency	25
2.4 The Simulation System	27
2.5 Simulation Aids	28
2.5.1 Model Preparation	28
2.5.2 Model Test Vector Development	29
2.5.3 Model Debugging	29
2.5.4 Results Interpretation	31
2.6 Applications of Simulation	32
2.7 Synthesis Tools	33

3 Basic Features of VHDL	41
---------------------------------	-----------

3.1 Major Language Constructs	43
3.1.1 Design Entities	43
3.1.2 Architectural Bodies	44
3.1.3 Model Testing	48
3.1.4 Block Statements	49
3.1.5 Processes	50
3.1.6	
* 3.2 Lexical Description	51
3.2.1 Character Set	52
3.2.2 Lexical Elements	52
3.2.3 Delimiters	53
3.2.4 Identifiers	53
3.2.5 Comments	54
3.2.6 Character Literal	55
3.2.7 String Literal	55
3.2.8 Bit String Literal	55
3.2.9 Abstract Literal	56
3.2.10 Decimal Literal	56
3.2.11 Based Literal	56
3.3 VHDL Source File	57
3.4 Data Types	57
3.4.1 Classification of Types	58
3.4.2 Scalar Data Types	58
3.4.3 Composite Data Types	64

3.4.4 Access Types	68
3.4.5 File Types	68
3.4.6 Type Marks	68
3.5 Data Objects	68
3.5.1 Classes of Objects	68
3.5.2 Declaration of Data Objects	69
3.6 Language Statements	72
3.6.1 Assignment Statements	72
3.6.2 Operators and Expressions	77
3.6.3 Sequential Control Statements	83
3.6.4 Architecture Declarations and Concurrent Statements	86
3.6.5 Subprograms	90
3.7 Advanced Features of VHDL	96
3.7.1 Overloading	96
3.7.2 Packages	99
3.7.3 Visibility	100
3.7.4 Libraries	103
3.7.5 Configurations	104
3.7.6 File I/O	107
3.8 The Formal Nature of VHDL	114
3.9 VHDL 93	115
3.9.1 Lexical Character Set	115
3.9.2 Syntax Changes	115
3.9.3 Process and Signal Timing and New Signal Attributes	116
3.9.4 New Operators	118
3.9.5 Improvements to Structural Models	118
3.9.6 Shared Variables	119
3.9.7 Improved Reporting Capability	120
3.9.8 General Programming Features	120
3.9.9 File I/O	121
3.9.10 Groups	122
3.9.11 Extension of Bit String Literals	122
3.9.12 Additions and Changes to Package Standard	122
3.10 Summary	122

4 Basic VHDL Modeling Techniques	135
4.1 Modeling Delay in VHDL	135
4.1.1 Propagation Delay	135
4.1.2 Delay and Concurrency	138
4.1.3 Sequential and Concurrent Statements in VHDL	140
4.1.4 Implementation of Time Delay in the VHDL Simulator	141
4.1.5 Inertial and Transport Delay in Signal Propagation	146
4.2 The VHDL Scheduling Algorithm	147
4.2.1 Waveform Updating	147
4.2.2 Side Effects	150
4.3 Modeling Combinational and Sequential Logic	150
4.4 Logic Primitives	153
4.4.1 Combinational Logic Primitives	153
4.4.2 SEQUENTIAL LOGIC	163
4.4.3 Testing Models: Test Bench Development	168
5 Algorithmic Level Design	185
5.1 General Algorithmic Model Development in the Behavioral Domain	186
5.1.1 Process Model Graph	187
5.1.2 Algorithmic Model of a Parallel to Serial Converter	189
5.1.3 Algorithmic Models with Timing	192
5.1.4 Checking Timing	195
5.2 Representation of System Interconnections	198
5.2.1 Comprehensive Algorithmic Modeling Example	199
5.3 Algorithmic Modeling of Systems	204
5.3.1 Multivalued Logic Systems	204
5.3.2 Comprehensive System Example	212
5.3.3 Time Multiplexing	222
6 Register Level Design	237
6.1 Transition from Algorithmic to Data Flow Descriptions	237
6.1.1 Transformation Example	238
6.2 Timing Analysis	241

6.3	Control Unit Design	243
6.3.1	Types of Control Units	243
6.4	Ultimate RISC Machine	245
6.4.1	Single URISC Instruction	246
6.4.2	URISC Architecture	246
6.4.3	URISC Control	248
6.4.4	URISC System	252
6.4.5	Design of the URISC at the Register Level	252
6.4.6	Microcoded Controller for the URISC Processor	254
6.4.7	Hardwired Controller for the URISC Processor	256

7	Gate Level and ASIC Library Modeling	261
----------	---	------------

7.1	Accurate Gate Level Modeling	261
7.1.1	Asymmetric Timing	262
7.1.2	Load Sensitive Delay Modeling	264
7.1.3	ASIC Cell Delay Modeling	269
7.1.4	Back Annotation of Delays	272
7.1.5	VITAL: A Standard for the Generation of VHDL Models of Library Elements	275
7.2	Error Checking	277
7.3	Multivalued Logic for Gate Level Modeling	280
7.3.1	Additional Values for MOS Design	280
7.3.2	Generalized State/Strength Model	281
7.3.3	Interval Logic	286
7.3.4	Vantage System	286
7.3.5	Multivalued Gate-Level Models	289
7.3.6	Accurate Delay Modeling	292
7.4	Configuration Declarations for Gate Level Models	292
7.4.1	Default Configuration	296
7.4.2	Configurations and Component Libraries	297
7.5	Modeling Races and Hazards	299
7.6	Approaches to Delay Control	307

8	HDL-Based Design Techniques	315
----------	------------------------------------	------------

8.1	Design of Combinational Logic Circuits	315
8.1.1	Combinational Logic Design at the Algorithmic Level	316

8.1.2	Design of Data Flow Models of Combinational Logic in the Behavioral Domain	323
8.1.3	Synthesis of Gate-Level Structural Domain Combinational Logic Circuits	324
8.1.4	Summary of Design Activity for Combinational Logic Circuits	329
8.2	Design of Sequential Logic Circuits	329
8.2.1	Moore or Mealy Decision	332
8.2.2	Construction of a State Table	333
8.2.3	Creating a State Diagram	333
8.2.4	Transition List Approach	336
8.2.5	Creating a VHDL Model for State Machines	337
8.2.6	Synthesis of VHDL State Machine Models	343
8.3	Design of Microprogrammed Control Units	345
8.3.1	Interface Between Controller and Device	345
8.3.2	Comparison of Hardwired and Microprogrammed Control Units	345
8.3.3	Basic Microprogrammed Control Unit	348
8.3.4	Algorithmic-Level Model of BMCU	349
8.3.5	Design of Microprogrammed Controllers for State Machines	350
8.3.6	Generalities and Limitations of Microprogrammed Control Units	358
8.3.7	Alternative Condition Select Methods	361
8.3.8	Alternative Branching Methods	364
9	ASICs and the ASIC Design Process	377
9.1	What is an ASIC?	377
9.2	ASIC Circuit Technology	379
9.2.1	CMOS Switches	380
9.3	Types of ASICs	381
9.3.1	PLDs	381
9.3.2	Field Programmable Gate Arrays	381
9.3.3	Gate Arrays	392
9.3.4	Standard Cells	394
9.3.5	Full Custom Chips	398
9.3.6	Relative Cost of ASICs and FPGAs	399

9.4	The ASIC Design Process	402
9.4.1	Standard Cell ASIC Synthesis	404
9.4.2	Post Synthesis Simulation	415
9.5	FPGA Synthesis	418
9.5.1	FPGA Example	419
9.5.2	Comparison with an ASIC Design	424

10	Modeling for Synthesis	429
-----------	-------------------------------	------------

10.1	Behavioral Model Development	429
10.1.1	Creation of the Initial Behavioral Model	430
10.1.2	Application-Domain Tools	431
10.1.3	Language-Domain Modeling	434
10.1.4	Modeling and Model Efficiency	437
10.1.5	Application-Domain vs. Language-Domain Modeling	438
10.2	The Semantics of Simulation and Synthesis	439
10.2.1	Delay in Models	444
10.2.2	Data Types	455
10.3	Modeling Sequential Behavior	455
10.4	Modeling Combinational Circuits for Synthesis	452
10.4.1	Synthesis of Arithmetic Circuits	457
10.4.2	Hierarchical Arithmetic Circuit: BCD to Binary Converter	459
10.4.3	Synthesis of Hierarchical Circuits	460
10.5	Inferred Latches and Don't Cares	465
10.6	Tristate Circuits	469
10.7	Shared Resources	471
10.8	Flattening and Structuring	472
10.9	Effect of Modeling Style On Circuit Complexity	474
10.9.1	Effect of Selection of Individual Construct	474
10.9.2	Effect of General Modeling Approach	476

11	Integration of VHDL into a Top-Down Design Methodology	489
-----------	---	------------

11.1	Top-Down Design Methodology	489
11.2	Sobel Edge Detection Algorithm	492
11.3	System Requirements Level	495
11.3.1	Written Specifications	495
11.3.2	Requirements Repository	495

11.4 System Definition Level	499
11.4.1 Executable Specification	499
11.4.2 Test Bench Development for Executable Specifications	508
11.5 Architecture Design	523
11.5.1 System Level Decomposition	523
11.5.2 Hierarchical Decomposition	527
11.5.3 Methodology for Development of Test Benches for a Hierarchical Structural Model	529
11.6 Detailed Design at the RTL Level	530
11.6.1 Register Transfer Level Design	531
11.6.2 Simulating Structural Models Using Components with Different Data Types	538
11.6.3 Test Bench Development at the RTL	544
11.7 Detailed Design at the Gate Level	545
11.7.1 Gate-Level Design of Horizontal Filter	545
11.7.2 Optimization of Gate-Level Circuits	545
11.7.3 Gate-Level Testing	547
11.7.4 Methodology for Back Annotation	548
12 Synthesis Algorithms for Design Automation	553
12.1 Benefits of Algorithmic Synthesis	553
12.2 Algorithmic Synthesis Tasks	554
12.2.1 Compilation of VHDL Description into an Internal Format	556
12.2.2 Scheduling	556
12.2.3 Allocation	557
12.2.4 Interaction of Scheduling and Allocation	558
12.2.5 Gantt Charts and Utilization	562
12.2.6 Creating FSM VHDL from an Allocation Graph	563
12.3 Scheduling Techniques	565
12.3.1 Transformational Scheduling	566
12.3.2 Iterative/Constructive Scheduling	567
12.3.3 ASAP Scheduling	567
12.3.4 ALAP Scheduling	568
12.3.5 List Scheduling	570
12.3.6 Freedom-Directed Scheduling	573
12.4 Allocation Techniques	574
12.4.1 Greedy Allocation	574

12.4.2 Allocation by Exhaustive Search	575
12.4.3 Left Edge Algorithm	575
12.4.4 Assigning Functional Units and Interconnection Paths.	577
12.4.5 Analysis of the Allocation Process	582
12.4.6 Nearly Minimal Cluster Partitioning Algorithm	584
12.4.7 Profit Directed Cluster Partitioning Algorithm (PDCPA)	589
12.5 State of the Art in High-Level Synthesis	600
12.6 Automated Synthesis of VHDL Constructs	602
12.6.1 Constructs that Involve Selection	602
12.6.2 Mapping <i>case</i> Statements to Multiplexers	602
12.6.3 Mapping <i>if...then...else</i> Statements to Multiplexers	604
12.6.4 Mapping Indexed Vector References to Multiplexers	605
12.6.5 Loop Constructs	605
12.6.6 Functions and Procedures	609

References	623
-------------------	------------

Index	633
--------------	------------

Structured Design Concepts

In this chapter we present basic definitions that relate to the design process. It is necessary to introduce them now so that other concepts can be explained. The reader should study them carefully in order to comprehend material introduced later. It will also be useful to revisit this chapter as one proceeds through the text since the full meaning of the terms will only become clear through use and example.

1.1 THE ABSTRACTION HIERARCHY

In this section we present the abstraction hierarchy employed by digital designers. Abstraction can be expressed in the following two domains:

Structural domain. A domain in which a component is described in terms of an interconnection of more primitive components.

Behavioral domain. A domain in which a component is described by defining its input/output response.

Figure 1.1 shows structural and behavioral descriptions for a logic circuit, which detects two or more consecutive 1's or two or more consecutive 0's on its input X. The structural description is an interconnection of gate and flip-flop primitives. The behavioral description is expressed textually in a hardware description language (HDL).

An abstraction hierarchy can be defined as follows:

Abstraction hierarchy. A set of interrelated representation levels that allow a system to be represented in varying amounts of detail.

Figure 1.2 shows a picture of a typical abstraction hierarchy. For each level i in the hierarchy there exists a transformation to level $i+1$. The level of detail usually increases monotonically as one moves down in the hierarchy.