

Processor Description Languages

Applications and Methodologies

Prabhat Mishra

University of Florida

Nikil Dutt

University of California, Irvine

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

ELSEVIER

Morgan Kaufmann Publishers is an imprint of Elsevier



MORGAN KAUFMANN PUBLISHERS

Contents

List of Contributors	xvii
Preface	xxv
About the Editors	xxvii

CHAPTER 1 Introduction to Architecture Description Languages 1

Prabhat Mishra and Nikil Dutt

1.1	What is an Architecture Description Language?	2
1.2	ADLs and Other Languages	3
1.3	Classification of Contemporary ADLs	4
1.3.1	Content-based Classification of ADLs	5
1.3.2	Objective-based Classification of ADLs	6
1.4	ADLs: Past, Present, and Future	8
1.5	Book Organization	9
	References	10

CHAPTER 2 ADL-driven Methodologies for Design Automation of Embedded Processors 13

Prabhat Mishra and Aviral Shrivastava

2.1	Design Space Exploration	13
2.2	Retargetable Compiler Generation	14
2.2.1	Retargetability Based on ADL Content	16
2.2.2	Retargetability Based on Compiler Phases	17
2.2.3	Retargetability Based on Architectural Abstractions	18
2.3	Retargetable Simulator Generation	21
2.3.1	Interpretive Simulation	22
2.3.2	Compiled Simulation	22
2.3.3	Mixed Approaches	22
2.4	Architecture Synthesis	23
2.4.1	Implementation Generation Using Processor Templates	23
2.4.2	ADL-driven Implementation Generation	24
2.5	Top-down Validation	24
2.5.1	Validation of ADL Specification	25
2.5.2	Implementation Validation	27
2.6	Conclusions	30
	References	30

CHAPTER 3	MIMOLA—A Fully Synthesizable Language	35
	<i>Peter Marwedel</i>	
3.1	Introduction	35
3.1.1	Origin of the Language	35
3.1.2	Purpose of the Language	36
3.1.3	Related Work: State of the Art in the Early Years	38
3.1.4	Outline of This Chapter	39
3.2	Salient Features of the MIMOLA Language	39
3.2.1	Overall Structure of Description	39
3.2.2	Declarations, Data Types and Operations	39
3.2.3	Program Definition	40
3.2.4	Structure Definition	43
3.2.5	Linking Behavior and Structure	46
3.2.6	Putting Things Together	48
3.3	Tools and Results	48
3.3.1	Design Flow	48
3.3.2	The Front-end and Internal Design Representations ..	48
3.3.3	Mapping to Register Transfers	49
3.3.4	Simulation	49
3.3.5	Architectural Synthesis	50
3.3.6	Test Program Generation	54
3.3.7	Code Generation	56
3.3.8	Overall View of the Dependence among MSS Tools ..	57
3.3.9	Designs Using MSS2	58
3.4	Conclusions	59
3.4.1	Evolution of Ideas and Directions	59
3.4.2	What Went Wrong and What Went Right	59
3.4.3	Summary	60
	References	60
CHAPTER 4	nML: A Structural Processor Modeling Language for Retargetable Compilation and ASIP Design	65
	<i>Johan Van Praet, Dirk Lanneer, Werner Geurts, and Gert Goossens</i>	
4.1	Introduction	65
4.2	The nML Processor Description Formalism	66
4.3	A Structural Skeleton of the Processor	66
4.3.1	Memories and Registers	67
4.3.2	Storage Aliases	68
4.3.3	Transitory Storage	68
4.3.4	Immediate Constants and Enumeration Types	69
4.3.5	Functional Units	70

4.4	Instruction-set Grammar	70
4.4.1	Breaking Down the Instruction Set: AND Rules and OR Rules	72
4.4.2	The Grammar Attributes	73
4.4.3	Synthesized Attributes	73
4.4.4	Action Attribute	74
4.4.5	Image Attribute	77
4.4.6	Syntax Attribute	78
4.4.7	Mode Rules and Value Attributes	78
4.4.8	Inherited Attributes	80
4.5	Pipeline Hazards: Stalls and Bypasses	80
4.5.1	Control Hazards	81
4.5.2	Structural and Data Hazards	82
4.6	The Evolution of nML	85
4.7	A Retargetable Tool Suite for ASIPs	87
4.7.1	Chess: A Retargetable C Compiler	87
4.7.2	Checkers: A Retargetable Instruction-set Simulator Generator	88
4.7.3	Go: A Hardware Description Language Generator ...	88
4.7.4	Risk: A Retargetable Test-program Generator	88
4.7.5	Broad Architectural Scope	88
4.8	Design Examples	90
4.8.1	Portable Audio and Hearing Instruments	90
4.8.2	Wireline Modems	90
4.8.3	Wireless Modems	91
4.8.4	Video Coding	91
4.8.5	Network Processing	91
4.9	Conclusions	92
	References	92
CHAPTER 5	LISA: A Uniform ADL for Embedded Processor Modeling, Implementation, and Software Toolsuite Generation	95
	<i>Anupam Chattopadhyay, Heinrich Meyr, and Rainer Leupers</i>	
5.1	Language-based ASIP Modeling	96
5.1.1	Intuitive Modeling Idea	96
5.1.2	ISA Modeling	96
5.1.3	Structure Modeling: Base Processor	99
5.1.4	Levels of Abstraction in LISA	100
5.1.5	LISA-based Processor Design	101
5.2	Automatic Software Toolsuite Generation	102
5.2.1	Instruction-set Simulator	103

	5.2.2	The Compiler Designer	105
	5.2.3	Custom Instruction Synthesis for LISA-based Processor Design	108
	5.2.4	Instruction Opcode Synthesis	109
5.3		Automatic Optimized Processor Implementation	110
	5.3.1	Automatic Generation of RTL Processor Description	110
	5.3.2	Area and Timing-driven Optimizations	111
	5.3.3	Energy-driven Optimizations	112
	5.3.4	Automatic Generation of JTAG Interface and Debug Mechanism	113
5.4		Processor Verification	113
	5.4.1	Equivalence Check via Simulation	114
	5.4.2	Generation of Test Vectors from LISA Descriptions for Instruction-set Verification	115
5.5		System-level Integration	117
	5.5.1	Retargetable Processor Integration	118
	5.5.2	Multiprocessor Simulation	119
5.6		Compact Modeling of Advanced Architectures	120
	5.6.1	VLIW Architecture Modeling	121
	5.6.2	Partially Reconfigurable Processor Modeling	122
5.7		Case Study	123
	5.7.1	ASIP Development for Retinex-like Image and Video Processing	124
	5.7.2	Retargetable Compiler Optimization for SIMD Instructions	127
5.8		Conclusions	130
		References	130
CHAPTER 6		EXPRESSION: An ADL for Software Toolkit Generation, Exploration, and Validation of Programmable SOC Architectures	133
		<i>Prabhat Mishra and Nikil Dutt</i>	
	6.1	Expression ADL	134
		6.1.1 Structure	135
		6.1.2 Behavior	137
	6.2	Software Toolkit Generation and Exploration	139
		6.2.1 Retargetable Compiler Generation	140
		6.2.2 Retargetable Simulator Generation	142
		6.2.3 Design Space Exploration	144
	6.3	Architecture Synthesis for Rapid Prototyping	148
		6.3.1 Synthesizable HDL Generation	148
		6.3.2 Rapid Prototyping and Exploration	149

6.4	Functional Verification	151
6.4.1	Specification Validation	152
6.4.2	Test Generation Using Model Checking	153
6.4.3	Implementation Validation	156
6.5	Conclusions	158
	References	159
CHAPTER 7	ASIP Meister	163
	<i>Yuki Kobayashi, Yoshinori Takeuchi, and Masabaru Imai</i>	
7.1	Overview of ASIP Meister	163
7.1.1	Framework Overview	164
7.1.2	Features	167
7.1.3	A Short History	168
7.1.4	ASIP Meister Usage in Academia	168
7.2	Architecture Model	168
7.3	ADL in ASIP Meister	170
7.3.1	Overview of ADL Structure	170
7.3.2	Specification Entry Using GUI	171
7.3.3	Microoperation Description	171
7.3.4	VLIW Extension	174
7.4	Generation of HDL Description	176
7.4.1	DFG Construction from Microoperation Description	177
7.4.2	Pipeline Registers and Datapath Selectors Insertion	177
7.5	Case Study	179
7.5.1	MIPS and DLX	179
7.5.2	M32R	180
7.5.3	MeP	180
7.6	Conclusion	181
7.7	Acknowledgments	181
	References	182
CHAPTER 8	TIE: An ADL for Designing Application-specific Instruction Set Extensions	183
	<i>Himanshu A. Sanghavi and Nupur B. Andrews</i>	
8.1	Introduction	183
8.1.1	Adapting the Processor to the Application	183
8.1.2	Tensilica Instruction Extension Language and Compiler	184
8.2	Design Methodology and Tools	184
8.2.1	Designing Application-specific Instructions	184
8.2.2	Design Automation with the TIE Compiler	186

8.3	Basics of TIE Language	187
8.3.1	A basic TIE Acceleration Example	187
8.3.2	Defining a Basic TIE Instruction	188
8.3.3	Instruction Encodings	189
8.3.4	Instruction Datapath	190
8.4	Adding Processor State	192
8.4.1	Defining a State Register	192
8.4.2	Defining a Register File	194
8.4.3	Data Type and Compiler Support	196
8.4.4	Data Parallelism and SIMD	198
8.5	VLIW Machine Design	199
8.5.1	Defining a VLIW Instruction	200
8.5.2	Hardware Cost of VLIW	201
8.6	Language Constructs for Efficient Hardware Implementation	201
8.6.1	Sharing Hardware between Instructions	201
8.6.2	TIE Functions	202
8.6.3	Defining Multicycle Instructions	203
8.7	Custom Data Interfaces	205
8.7.1	Import Wire and Export State	206
8.7.2	TIE Queue	206
8.7.3	TIE Lookup	209
8.8	Hardware Verification	210
8.8.1	Microarchitectural Verification	211
8.9	Case Study of an Audio DSP Design	212
8.9.1	Architecture and ISA Overview	212
8.9.2	Implementation and Performance	213
8.10	Conclusions	214
	References	215
CHAPTER 9	MADL—An ADL Based on a Formal and Flexible Concurrency Model	217
	<i>Wei Qin, Subramanian Rajagopalan, and Sharad Malik</i>	
9.1	Introduction	217
9.2	Operation State Machine Model	218
9.2.1	Static OSM Model	219
9.2.2	Dynamic OSM Model	222
9.2.3	Scheduling of the OSM Model	224
9.3	MADL	226
9.3.1	The AND-OR Graph: A Review	227
9.3.2	The Core Layer	228
9.3.3	The Annotation Layer	231
9.4	Support for Tools	233

9.4.1	Cycle Accurate Simulator (CAS)	233
9.4.2	Instruction Set Simulator	235
9.4.3	Disassembler	235
9.4.4	Register Allocator	235
9.4.5	Instruction Scheduler	237
9.5	Results	239
9.6	Related Work	242
9.7	Conclusion	244
	References	244

CHAPTER 10 ADL++: Object-Oriented Specification of Complicated Instruction Sets and Microarchitectures **247**

Soner Önder

10.1	Flexible Architecture Simulation Toolset (FAST)	248
10.1.1	Overview of the Toolset	248
10.2	The FAST/ADL Model	249
10.2.1	Timing of Events	251
10.2.2	Time Annotated Actions and Parallelism in the Microarchitecture	252
10.2.3	Microarchitecture Specification	253
10.2.4	ISA Specification	257
10.2.5	Putting it Together	259
10.3	Review of Complex Instruction Set Architectures	260
10.3.1	Variable Length Instructions	260
10.3.2	Many Memory Addressing Modes	261
10.3.3	Overlapping Registers	262
10.3.4	Mixed Arguments	263
10.3.5	Assembly Language Syntax	263
10.4	Sets and Regular Expressions as Language Constructs	264
10.4.1	Registers and Their Specification Using Sets	264
10.4.2	Regular Expressions and Addressing Modes	265
10.5	Instruction Templates and Multiple Conditional Inheritance	266
10.5.1	Inheritance with a Twist: Multiple Conditional Inheritance	267
10.5.2	Instruction Templates	267
10.6	Object-oriented Microarchitecture Specification	270
10.6.1	Artifacts as Objects	270
10.6.2	Deriving Complex Architectures From Objects	272
10.7	Epilogue	272
10.8	History of FAST and ADL++	272
	References	273

CHAPTER 11	Processor Design with ArchC	275
	<i>Guido Araujo, Sandro Rigo, and Rodolfo Azevedo</i>	
11.1	Overview	275
11.2	Syntax and Semantics	276
11.3	Integration through a TLM Interface	283
11.3.1	ArchC TLM Interfaces and Protocol	284
11.3.2	TLM Interrupt Port	285
11.3.3	A Word on ArchC Simulators	286
11.4	A Multicore Platform Example	288
11.5	Conclusions	293
	References	293
CHAPTER 12	MAML: An ADL for Designing Single and Multiprocessor Architectures	295
	<i>Alexey Kupriyanov, Frank Hannig, Dmitriy Kissler, and Jürgen Teich</i>	
12.1	History of MAML	296
12.2	Description of Single Processor Architectures	298
12.2.1	Syntax	298
12.2.2	Example of a VLIW Processor Architecture	305
12.3	Description of Multiprocessors	306
12.3.1	Related Work	307
12.3.2	Description of Parallel Processing Elements	308
12.3.3	Parametric Domains as a Description Paradigm	310
12.3.4	Description of Adaptive Interconnect Topologies	313
12.3.5	Case Study of a Tightly Coupled Processor Arrays	316
12.4	Approaches and Tools around MAML	317
12.4.1	Application Mapping	318
12.4.2	Design Framework	319
12.4.3	Interactive Visual Architecture Entry	320
12.4.4	Simulator Generation	321
12.4.5	Architecture Synthesis for Rapid Prototyping	324
12.5	Conclusions and Future Work	324
12.6	Acknowledgments	325
	References	325
CHAPTER 13	GNR: A Formal Language for Specification, Compilation, and Synthesis of Custom Embedded Processors	329
	<i>Bitá Gorjiara, Mehrdad Reshadi, and Daniel Gajski</i>	
13.1	Introduction	329
13.2	Overview of NISC Technology	332

13.3	Modeling NISC Architectures and Systems	335
13.3.1	GNR Formalism	337
13.3.2	GNR Rules	339
13.3.3	GNR Syntax	341
13.3.4	Basic Components	341
13.3.5	Hierarchical Components	342
13.3.6	Modeling an NISC Architecture	343
13.3.7	Communication Modeling	347
13.3.8	Generating RTL Code from GNR	351
13.4	Experiments: Design-space Exploration Using NISC and GNR	353
13.4.1	Designing General-purpose NISCs	353
13.4.2	Custom Datapath Design for DCT	357
13.4.3	Communicating NISC Components	363
13.5	Conclusion	365
	References	366
13.6	Index Terms	367
CHAPTER 14	HMDDES, ISDL, and Other Contemporary ADLs	369
	<i>Nirmalya Bandyopadhyay, Kanad Basu, and Prabhat Mishra</i>	
14.1	HMDDES	369
14.1.1	HMDDES Language	369
14.1.2	Structural Overview of Machine Description	373
14.1.3	Trimaran Infrastructure	383
14.2	ISDL	384
14.2.1	ISDL Language	385
14.2.2	ISDL-driven Methodologies	388
14.3	RADL	390
14.4	SIM-nML	390
14.5	UDL/I	392
14.6	Flexware	392
14.7	Valen-C	392
14.8	TDL	393
14.9	Conclusions	393
	References	394
Index	395