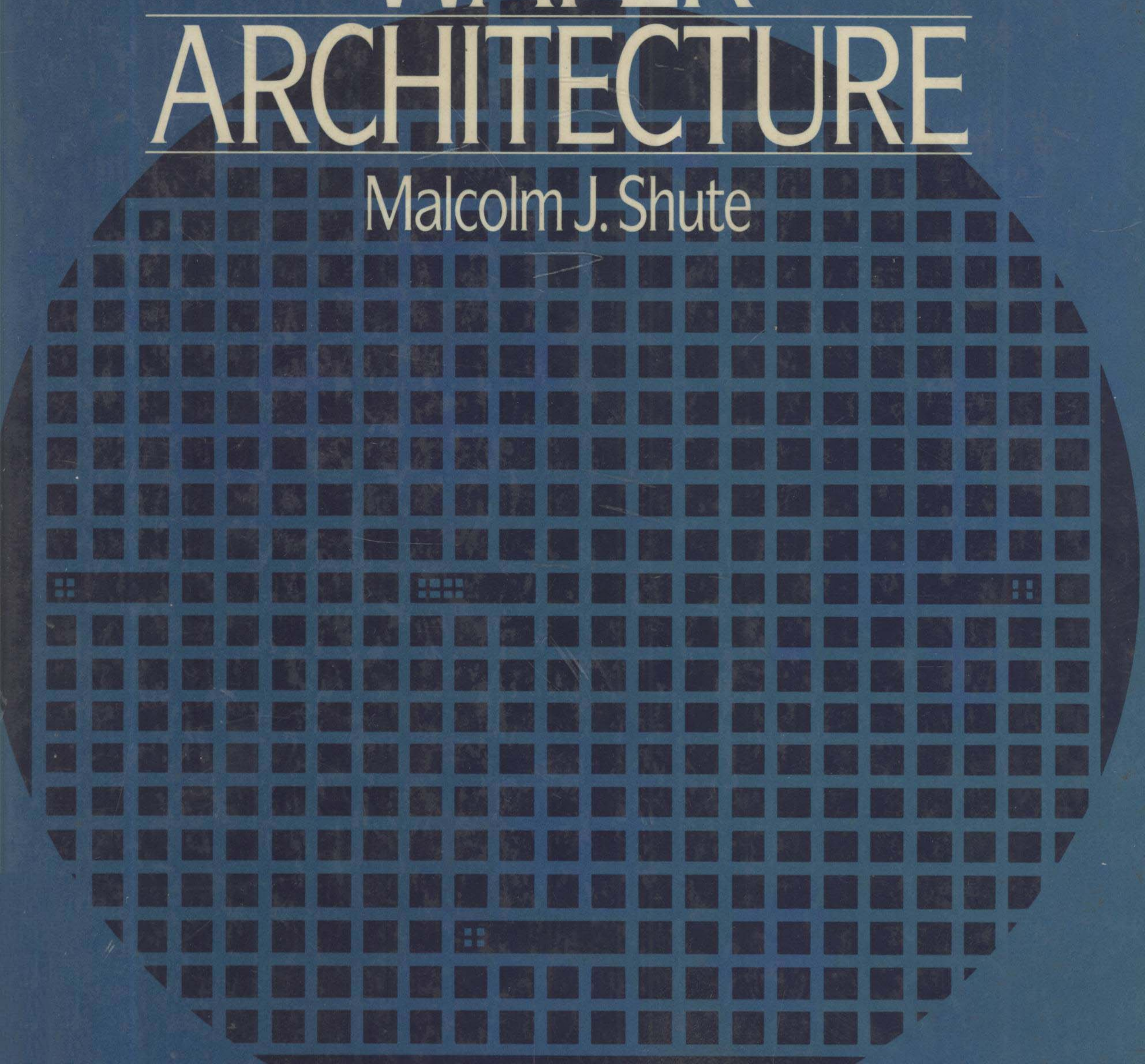
FIFTH GENERATION WAFER ARCHITECTURE

Malcolm J. Shute



FIFTH GENERATION WAFER ARCHITECTURE

MALCOLM J. SHUTE
Microelectronics Centre,
Middlesex Polytechnic, London

PRENTICE HALL
NEW YORK LONDON TORONTO SYDNEY TOKYO

First published 1988 by
Prentice Hall International (UK) Ltd,
66 Wood Lane End, Hemel Hempstead,
Hertfordshire, HP2 4RG
A division of
Simon & Schuster International Group

Printed and bound in Great Britain at
the University Press, Cambridge

Library of Congress Cataloging-in-Publication Data

Shute, Malcolm J., 1955–
Fifth generation wafer architecture/by Malcolm
J. Shute. p. cm.
Bibliography: p.
Includes index.
ISBN 0-13-314238-8
1. Fifth generation computers. 2. Computer
architecture. 3. Integrated circuits—Design and
construction. I. Title.
QA76.85. S58 1988
004. 16–dc 19 87-27313

British Library Cataloguing in Publication Data

Shute, Malcolm J.
Fifth generation wafer architecture.
1. Computer architecture
I. Title
004.2′1 QA76.9.A73

ISBN 0-13-314238-8

1 2 3 4 5 91 90 89 88 87

ISBN 0-13-314238-8

FIFTH GENERATION WAFER ARCHITECTURE

Remembering Gran and Grandpa,
whose inspired purchase (Hellyer 1971)
started me along this path

PREFACE

This volume explores the possible realisation of fifth generation computer architectures through wafer scale integration. It looks at architectural considerations, and designs for fifth generation, non-von-Neumann computer architectures. It also considers fault tolerance, failure tolerance, reconfiguration and wafer scale integration.

As integrated circuit technology improves, the demands on it increase too. Invariably, as hardware is developed for one application, such as a computer architecture, the demands for more functionality, and a consequential proliferation of its components, promptly follow.

Present technology now allows the fabrication of circuits with more than 100 000 transistors per die. Future systems will require even more than this, but how can these circuits be fabricated, given that present technology is stretched nearly to the limit? How can they be designed, given that present designs are already more complicated than the combined road, rail, electricity, gas, water and telephone layouts of a major city? What principles of operation will be employed, given that the current ones now show severe signs of limitation?

This book sets out to answer these questions. It describes a number of the techniques and problems of building the largest single chip computers possible, namely those which occupy an entire wafer of semiconductor. It necessarily spans the many subject areas which influence computer design, in particular those of computer science, suggesting that which is desirable in a computer, and those of microelectronics, indicating that which is possible in a computer.

The skills of the computer engineer will be in great demand to design new architectures for the future breeds of supercomputer. The exciting prospect of finding a successor to a system which has not been surpassed in forty years is now realistic.

The academic study of this subject seems to be polarised into two classes of department: those which study computing, and those which study electronic engineering. However, this book is neither a computer science text, nor one for microelectronics, though both areas are introduced here. Instead, it reports on some exciting research work from various sources around the world which is relevant to computer engineering and computer architecture design.

Although the work which is reported here is currently the subject of research, this book has been written with Master's level teaching in mind. Some of the introductory material is included simply because it is useful to have a basic introduction included in the one text rather than be constantly referred elsewhere. Ample references are given though for further reading.

ACKNOWLEDGMENTS

Firstly, I would like to thank Dr Paul Kelly who, with his amazing library of up-to-date papers and proceedings and his breadth of understanding in a vast spectrum of studies, has contributed more than significantly to the writing of this book.

I am very grateful to: everyone working on the Alvey Cobweb (Alv/IKBS/064/150) project. Professor Peter Osmon and Dr Chris Hankin have been close friends of mine since the beginnings of my postgraduate studies, and have been a constant source of encouragement, and learning. Notably, they have contributed to the computer science side of the work which is reported here, in Chapters 2 and 3.

Indeed, I am doubly indebted to Chris Hankin, and also to Andrew McCabe, both of whom gave me the highly critical, yet encouraging reviews which an earlier draft of this book needed. I am also very grateful to Drs Chris Jesshope and Tony Ambler, both of whom gave similarly constructive critisms of an even earlier draft.

From the Alvey Wafer Scale Integration project (Alv/Prj/Arch/073) I would like to thank Dr Geoff Sumerling, and Dr Russell Aubusson for their contributions to my general understanding of wafer scale integration; Dr Will Moore for his help with the performance and economics of water scale integration; Professor Mike Lea and Steve Clarke for their frequent seminars, discussions and reports on many aspects of wafer scale integration; Andrew McCabe for his contributions in systolic array and photolithography discussions; Ken Warren for his contributions to my discussion of electrical design issues; Richard Illman who helped me to understand some of the issues of built-in self test and electronic computer-aided design; Ray McKirdy for his help with my discussion of the physical design issues of wafer scale packaging; and each of the group leaders, whose reports I have used as the basis for Chapters 7 and 8.

The help given by my collegues at Middlesex Polytechnic has been invaluable, notably that from: Richard Bayford for his suggestions on the systolic array and built-in self-test sections; Kevin Johnstone for his comments on the electrical design issues sections; Keith Pitt for his contributions to the section on packaging; John White for his help with aspects of ion implantation; Dave Court for his general comments about microelectronics fabrication; Dick Gledhill for his critical comments on the final draft; Dr Richard Seals and Ray Ruocco for their many helpful suggestions concerning the presentation of Chapters 2 and 3. Thank you too, to Linda Moore and her long-suffering team of computer operation and reception staff who bore with me during the writing stages of the lecture notes which finally lead to this book.

I also acknowledge the funding which is provided by Middlesex Polytechnic, and the Alvey directorate of Great Britain, particularly on the above-mentioned Cobweb and WSI projects, which have enabled me to conduct this work. In particular, I thank Professors John Butcher and Frank Tye for securing the research post at the polytechnic. Most of all, I would like to thank Alan Bagshaw of the Alvey directorate for his tremendous support.

In addition, I thank Simon Peyton Jones and Chris Clack for many productive conversations, and Professor Ronan Sleep, also Professors John Darlington and John Gurd, respectively, for their discussions on the Flagship and Manchester University Dataflow projects. Also thanks go to Leon Bentley for suggestions during the writing of the sections on rotary switch replication, and mechanical antifuses; Dr Mary Sheeran for the sections on μFP; Hugh Glaser for discussions on the historical aspects of computing; Dr P.K. Chaturvedi for his comments about WSI packaging; Ruben Ashkenasy for discussions on direct-write photolithography; and Paul Anderson for his comments and suggestions in review of Chapters 2 and 3.

The author is indebted to Glen Murray for his continual help and advice during the preparation of this text, and for his work in finding the highly constructive reviewers of the early manuscripts; also to Glynice Smith, who provided much guidance on my writing style; and, of course, also to all of my friends and relations, especially Mum, Dad, Jancis and John, for putting up with me during the final stages of the writing.

GLOBALLY RESERVED NAMES

A number of terms are used consistently throughout this book. These are listed here, along with a brief definition. Fuller definitions can be found in the glossary at the back of the book, and in the text, mainly on pages 125–35.

a	area of cell	A	area of device
δa	area of fault/failure tolerant overhead per circuit	δA	area of non fault/failure tolerant overhead per device
$\mathbf{a}$	area of circuit with fault/failure tolerance logic	A'	area of complete fault/failure tolerant device
b	reliability of cell	B	reliability of device
δb	reliability of failure tolerant overhead per circuit	δB	reliability of non failure tolerant overhead per device
$\mathbf{b}$	reliability of circuit with failure tolerance logic	B'	reliability of complete failure tolerant device
c	number of cells needed	C	number of cells fabricated
$\mathbf{c}$	number of cells per circuit		
d	diameter of water	D	fault density
δd	unusable margin round wafer		
d'	usable diameter of wafer		
		F	number of faults/failures tolerated by device
h	harvest of cells	H	harvest of devices
j	an integer		
k	an integer	K	number of faults/failures to kill the device
m	number of rows of cells needed	M	number of rows fabricated
n	number of columns of cells needed	N	number of columns fabricated
p	a probability $(0 \leq p \leq 1)$		
r	distance from centre of wafer	R	replication factor
s	number of sides on a cell	S	speed ratio
t	time		
v	yield of circuits	V	yield of fault/failure tolerant devices
		W	usable area of wafer
y	yield of cells	Y	yield of devices

y	yield of circuits with fault/failure tolerance logic	Y'	yield of fault/failure tolerant devices
z	number of test processors per device		

α	fault clustering factor	λ	unit length in circuit layout
θ	angle relative to the major water flat	σ	standard deviation
$\varkappa$	transistor	τ	transistor switching time

crop	number of working devices obtained per wafer	*rco*	relative cell overhead
crop'	number of working fault/failure tolerant devices obtained per wafer	*rda*	relative device area
num	number of devices fabricated per wafer	*rdo*	relative device overhead
rca	relative cell area	*rpa*	relative processor area
		rpo	relative processor overhead
		TC	test coverage
		TQ	test quality

CONTENTS

Preface		xiii
Globally reserved names		xvi
1	**FIFTH GENERATION COMPUTING BACKGROUND**	**1**
1.1	Background study of fifth generation computing	2
	1.1.1 Targets and requirements	3
	1.1.2 The new study areas	6
1.2	Computer architecture: the conventional model	11
	1.2.1 The central processor unit	13
	1.2.2 The memory	14
	1.2.3 Instruction execution	17
	1.2.4 Characteristics	20
1.3	Summary and organisation of the book	20
1.4	Exercises	21
2	**EXPRESSING IDEAS TO THE COMPUTER**	**22**
2.1	Decimal, octal and hexadecimal assemblers	24
2.2	Symbolic assemblers	26
	2.2.1 Data structures	26
	2.2.2 Control structures	28
2.3	Compilers	29
	2.3.1 Data structures	30
	2.3.2 Control structures	30
	2.3.3 Functions	31
2.4	Structured imperative languages	32
	2.4.1 Data structures	33
	2.4.2 Control structures	34
	2.4.3 Functions	37
2.5	Declarative languages	39
	2.5.1 Data structures	41
	2.5.2 Control structures	43
	2.5.3 High level languages	44
	2.5.4 Very high level languages	45
2.6	Conclusions	46
2.7	Exercises	48

3 BRIDGING THE SOFTWARE GAP 49

3.1 Medium level languages and interpretation 49
3.1.1 Interpreted languages 50
3.1.2 Unstructured imperative languages 51
3.1.3 Structured imperative languages 55
3.1.4 Declarative languages 57
3.2 Program compilation and the execution model 61
3.2.1 Building a syntax tree 63
3.2.2 Reading a syntax tree 65
3.2.3 Compiling to Forth 69
3.2.4 Compiling to combinators 71
4.2.5 Evaluation mechanisms 73
3.3 Conclusions 74
3.4 Exercises 75

4 FASTER EXECUTION AND PARALLEL COMPUTATION 75

4.1 Medium level instruction sets 76
4.1.1 Structured imperative languages 78
4.1.2 Declarative languages 79
4.2 Massive-grain parallelism 81
4.2.1 Independent parallelism 82
4.2.2 Loosely coupled networks 83
4.3 Networking 83
4.3.1 Shared bus connection 84
4.3.2 Cross-bar exchange connection 84
4.3.3 Vector connection 85
4.3.4 Ring connection 86
4.3.5 Grid connection 86
4.3.6 Star connection 87
4.3.7 Tree connection 88
4.3.8 Butterfly connection 89
4.3.9 Hypercube connection 91
4.3.10 Packet-switched versus circuit-switched networks 93
4.4 Vertical parallelism: pipelining 95
4.5 Horizontal parallelism 99
4.5.1 Regular structures 99
4.5.2 Irregular structures 101
4.6 Performance 103
4.7 Models of computation 107
4.7.1 Examples of computer operation 108
4.7.2 Classification and examples 110
4.8 Novel architecture 112
4.8.1 Data flow machines 112
4.8.2 Reduction machines 115
4.8.3 Cellular automata 117
4.9 Conclusions 123
4.10 Exercises 124

5 CONFIGURATION: STRATEGIES 125

5.1 Terminology and statistics 125
  5.1.1 Fault-free processing 128
  5.1.2 The yield function 129
  5.1.3 Fault tolerance 131
  5.1.4 Failure tolerance 133
  5.1.5 The 'atleast' function 135
5.2 Direct replacement 135
  5.2.1 Patching 135
  5.2.2 Integer replication 135
5.3 Linear final device 139
  5.3.1 Bypass 141
  5.3.2 Meander cutting 141
  5.3.3 Meander growth 141
  5.3.4 Spiral growth 144
5.4 Rectangular final device 147
  5.4.1 Standby-spare replacement 147
  5.4.2 Standby-spare cell replacement 152
  5.4.3 Standby-spare hybrids 154
  5.4.4 Rotary switch connection 160
5.5 Tree-shaped final device 164
  5.5.1 Tree cutting 164
  5.5.2 Tree growth 164
5.6 Path redundancy 166
  5.6.1 Perfect harvest 167
  5.6.2 Convex wrapping 170
  5.6.3 Multidimension topologies 171
5.7 Time redundancy 171
5.8 Non-nearest-neighbour interconnection 173
  5.8.1 Partitioned bypass 174
  5.8.2 Corridor routing 175
  5.8.3 Diogenes 176
  5.8.4 Tree of matrices 178
  5.8.5 Divide-and-conquer matrix–matrix mapping 178
5.9 Masking redundancy 180
  5.9.1 Modular redundancy 182
  5.9.2 Block parity 185
  5.9.3 Hamming correction 185
5.10 Further variations and design decisions 188
5.11 Exercises 188

6 CONFIGURATION: IMPLEMENTATION AND PERFORMANCE 190

6.1 Defect tolerance (fault-free wafers) 190
6.2 Fault tolerance (static configuration) 191
  6.2.1 Patching 192
  6.2.2 Discretionary wiring 192
  6.2.3 Fusible links 193

6.2.4 Summary of static configuration techniques 199
6.3 Off-line failure tolerance (dynamic configuration) 200
6.3.1 External pattern storage 201
6.3.2 Internal pattern storage 203
6.3.3 Internal control 203
6.4 On-line failure tolerance (spontaneous configuration) 208
6.4.1 Modular redundancy 208
6.4.2 Hamming correction 209
6.5 Performance 209
6.6 Exercises 215

7 PHYSICAL AND ELECTRICAL DESIGN ISSUES 217

7.1 Clock and signal distribution 217
7.1.1 Slow clock 218
7.1.2 Clock zoning 218
7.1.3 Local, equal-length communications only 219
7.1.4 Asynchronous communication 220
7.2 Power distribution 220
7.2.1 Degraded supplies 220
7.2.2 Zoned power networks 221
7.2.3 Local power distribution 221
7.2.4 Possible future techniques 222
7.3 Lithography 222
7.3.1 One-to-one 223
7.3.2 Direct step on wafer 224
7.3.3 Direct write on wafer 226
7.4 Fault density and parameter variation 227
7.4.1 Parameter variation 228
7.4.2 Fault density 230
7.5 Packaging 232
7.5.1 Electrical contacts 233
7.5.2 Power dissipation 234
7.5.3 Thermal expansion 235
7.5.4 Protection: mechanical, chemical and electrical 235
7.5.5 Packaging material 235
7.6 Conclusions 236
7.7 Exercises 236

8 DESIGN CONSIDERATIONS 237

8.1 Partitioning 238
8.1.1 Applicability of the fault/failure techniques 238
8.1.2 Hierarchical fault/failure tolerance 239
8.1.3 Customisation 242
8.2 Design tools 242
8.2.1 Describing circuit structure 245
8.2.2 Describing circuit behaviour 246
8.2.3 μFP as a design tool 248

8.3 Testing, self testing and testability 250
8.3.1 Implications on wafer scale integration 252
8.3.2 Classification of tests 252
8.3.3 Observability and controllability enhancements 254
8.3.4 Built-in self-test 256
8.3.5 Automatic configuration 259
8.4 Conclusions 262
8.5 Exercises 262

9 ECONOMICS-DRIVEN RESEARCH AND DEVELOPMENT 263

9.1 Economics of program development and execution 264
9.1.1 Parallelism 265
9.1.2 Functional programming languages 266
9.2 Economics of hardware development and manufacture 268
9.2.1 Wafer scale integration 269
9.2.2 Modularity and regularity 270
9.3 The history and future of wafer scale integration 271
9.4 The emergence of the fifth generation computer 272

Appendix 1 Circuit models 273
Appendix 2 Pascal function to compute the atleast function 276
Appendix 3 Pascal program to build a syntax tree 278
Appendix 4 Answers to exercises 282

Glossary 287
References 299
Index 309

1

FIFTH GENERATION COMPUTING BACKGROUND

Digital electronic computer design has been an engaging subject for forty years. In this time, some five orders of magnitude of improvement of performance have been attained, but solely through trimming the original design to its present highly honed state. There is no doubt that further improvement of performance is needed, but this cannot be expected to be achieved by mere tuning of the current design any further. As a result, radically new types of computer are being investigated by research groups throughout the world, and this book reports on some of the implications of this work. A study is made of some possible applications in which these machines might be used, programmed and implemented. Necessarily, some degree of conjecture is involved, along with a backward glance at how the predecessors are used, programmed and implemented. The latter is necessary in order to understand why certain techniques and styles are still adopted, and others have been modified or dropped in the light of the lessons of the past.

Over recent decades, many lessons have been learned. Not least, there is a striking similarity between many of the problems and their possible solutions which are found in electronic engineering and computer programming. Both disciplines have learned that the design task becomes very much easier if the system is *hierarchical*, highly *regular* and highly *modular*. They have also learned how efficiency of the final product benefits from the use of *local*, highly *parallel* interconnections within the regular arrangement of modules. However, it is critical that the right *granularity* be found in all cases, and at *all* levels of detail. In other words, the modules must neither be too small and plentiful, nor too large and scarce; similarly, communications should involve messages which are neither too short and prolific, nor too long and infrequent (Figure 1.1).

From the hardware end, computer architects want programming languages which make efficient use of their hardware primitives. From the software end, functional language designers want new computer architectures which support their ideas

Parallel interconnectivity
Local interconnectivity
High modularity
High regularity
Optimum granularity

FIGURE 1.1 *Recurring themes*

efficiently. In a third corner, computer engineers need computer-aided design programs to describe the structures which they intend to implement (Ullman 1984, Sheeran 1985). There are, therefore, many lessons which each group can learn from the others, and much virtue in a closer co-operation between the disciplines to solve the imminent problems in computer design. Since this idea is so central to the aim of this book, the common themes, as listed above, are repeated in Figure 1.1. They will appear many times in this text, even across many subject boundaries. The computer architect must be on the alert for this sort of commonality, and ready to exploit it at all times.

This book concentrates on two seemingly unrelated ideas, namely functional language programming, and wafer scale integration (WSI), and uses them to illustrate the common points of computer science and microelectronics. It does not matter whether WSI ever becomes economically viable. It is not certain even that functional programming will become economic. What is important is that the techniques which are eventually adopted will bear the same sort of interrelationships. Functional language programming and WSI are therefore used here merely as the vehicles with which to illustrate, to catalogue and to classify the points which are made in this book.

A summary is given in the next section (Section 1.1) of the aims and concerns of the fifth generation programme. This helps to set the context: the goals towards which the computer engineering ideas of this book hope to contribute, at least in some small way. The section after (Section 1.2) describes the ideas on which the new architectures must be built if forty years of work and lessons are not to be ignored.

1.1 BACKGROUND STUDY OF FIFTH GENERATION COMPUTING

The fifth generation programme was initiated by the Japanese in 1981 (Simons 1983) to develop computer systems which are faster, more reliable and more intelligent than those of the present, and was scheduled, rather optimistically, for completion in 1991. Western countries, perceiving an imminent domination by the Japanese, followed with their own initiatives. Respectively, they are Alvey (in the UK), ESPRIT (in the EEC), DARPA and MCC (both in the USA). Along with the Japanese ICOT programme, they can be grouped collectively under the 'fifth generation' label.