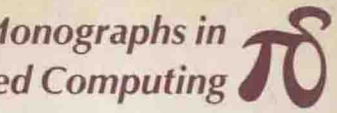
Research Monographs in
Parallel and Distributed Computing



ANDREW RUSHTON

Reconfigurable Processor–Array: a bit–sliced parallel computer

RESEARCH MONOGRAPHS IN PARALLEL AND DISTRIBUTED COMPUTING

Andrew Rushton
University of Southampton

Reconfigurable Processor-Array: a bit-sliced parallel computer

Pitman, London

The MIT Press, Cambridge, Massachusetts

PITMAN PUBLISHING
128 Long Acre, London WC2E 9AN

First published 1989

Available in the Western Hemisphere and Israel from
The MIT Press
Cambridge, Massachusetts (and London, England)

ISSN 0953-7767

British Library Cataloguing in Publication Data
Rushton, Andrew
Reconfigurable processor-array.
1. Computer systems. Parallel – processor systems.
I. Title II. Series
004'.35
ISBN 0-273-08799-1

Library of Congress Cataloging-in-Publication Data
Rushton, Andrew.
Reconfigurable processor array : a bit-sliced parallel computer / Andrew Rushton.
p. cm.—(Research monographs in parallel and distributed processing)
Bibliography: p.
Includes index.
ISBN 0-262-68057-2 (pbk.)
1. Parallel processing (Electronic computers) 2. Bit slice microprocessors 3. Electronic data processing—Distributed processing. I. Title. II. Series.
QA76.5.R865 1989
004'.35—dc19

Reproduced and printed by photolithography
in Great Britain by Biddles Ltd, Guildford

Foreword

As one of the main editors of this series, it gives me great pleasure to be able to write this foreword for one of the first monographs to be published.

Parallel processing is not a new field, but is sufficiently active now to require a series such as this to bring up-to-date research or a consolidation of significant contributions into a forum for wider dissemination. It gives me particular pleasure, however, in being able to write this preface for a manuscript from one of my own research students, concerning a project that has occupied much of my interest over the last five years: the Reconfigurable Processor-Array (RPA).

Contrary to much recent opinion, highly parallel computer systems have been exploited now for some time. The first ICL DAP, comprising 1024 albeit very simple processors, was in operation over ten years ago. That period also saw the beginnings of major changes in the electronics industry, with prophets such as Carver Mead taking a long, forward look at the potential of MOS systems. The RPA array processor has its roots back in that period, in the convergence of VLSI technology with parallel processing. Another influence in the forming of the RPA project came from the seemingly unavoidable dichotomy between the requirement for a fixed regular structure for implementation in silicon, and the user's requirement to represent operations and communications dictated by applications data structures, not necessarily the same as the fixed silicon structure.

The history of this research project is long and very arduous, involving grant applications to industry, SERC and, when put in motion, the ALVEY program. However as is usual in any university environment, research work does not start and finish with large research grants to lubricate that research; much significant work is carried out by research students. Andrew Rushton was one such student; he was involved with the research from the outset, and played a significant part in the design of the RPA architecture. In the event the research was finally funded by the Alvey program under its VLSI architecture section. Although the Alvey support was modest (the support was for a university-only project, with industrial 'uncles'), it provided a core of support for simulation and software development, and facilities to take the project to a prototype chip implementation.

This monograph is primarily concerned with the silicon architecture, and explores the synergy between VLSI technologies and parallel cellular architectures. As well as the chip implementation, on which this monograph is based, this Alvey-funded research project investigated most aspects of the RPA concept, from controller and host system design, through early program development tools, to applications investigation.

The common element, or binding, of the different aspects of this work on the RPA computer system has been a micro-program development tool, based on a low-level simulation of the RPA chip and controller. Using this software, it has been possible to develop and evaluate low-level code, and to iterate the silicon architecture, before it was committed to processing. The work within this group was therefore very organic, and although the divisions of responsibility were quite specific, many of the advances made can be attributed to the group as a whole. I would therefore like to add my

acknowledgement at this opportunity to Alvey for funding this work and also to Adriano Cruz, Jimmy Stewart and Russel O'Gorman.

This monograph, then, gives the view of the research project from the position of the silicon designer, Andrew Rushton, but this is quite appropriate as it is in the silicon implementation that we find both the drive and the constraints on the RPA architecture. This is as ever the two-edged sword of technology.

More as a postscript to this work than a preface, some comments are appropriate concerning the outcome of the project. At the time of writing this foreword, the RPA project had come to its conclusion. Funds are not available to take this work to a full production prototype, nor would we wish to do so, as this research has inevitably revealed limitations in the RPA design, which have become apparent as our understanding of the problem has increased. The work is still proceeding, but based on a more abstract model for a virtual systems architecture for concurrent data operations (also funded by Alvey). RPA II will be an implementation of that system!

My hope is, therefore, that this monograph will act both as a case study on a particular design strategy and as background to further research. Perhaps what has been discovered in this design cycle for the RPA is that the VLSI/parallel processing synergy will have to be extended to more abstract problem-solving paradigms, before a true exploitation of the technology can be achieved. Neither can proceed without a thorough understanding of the other; moreover, with the rate at which the technology advances, this knowledge is at best a moving target.

Chris Jesshope

Contents

List of figures
Glossary

1 Introduction 1

2 Introduction to parallel computers 5
2.1 A brief history of parallel computers 5
2.1.1 First proposals 5
2.1.2 The first word-slice parallel computers 7
2.1.3 Bit-slice architectures 9
2.2 Types of parallel computer 10
2.2.1 The Flynn classification 10
2.2.2 Inter-processor communications 11
2.3 Why choose a processor-array? 12
2.3.1 SIMD versus MIMD 12
2.3.2 Distributed network versus lumped switch 13
2.3.3 Topology of the network 13
2.3.4 Conclusions 14
2.4 Applications of processor-arrays 15
2.4.1 Meteorology and oceanography 15
2.4.2 Earth resources 16
2.4.3 Medical 16
2.4.4 Engineering design 16
2.4.5 Signal processing 17
2.4.6 Artificial intelligence 17
2.4.7 References 17
2.5 A case study of two SIMD networks 18
2.5.1 The Distributed-Array Processor (DAP) 18
2.5.2 The Connection Machine 23

3 Architecture of the Reconfigurable Processor-Array 27
3.1 System description 27
3.1.1 Overview of the RPA system 27
3.1.2 Simulation and micro-code development 30
3.1.3 Micro-controller interface to the array 33
3.1.4 Data stream interface to the host 34
3.2 Adaptive parallelism and reconfigurability 35

3.2.1 The problem with fixed networks 36
3.2.2 A solution - intermediate modes 36
3.2.3 Hardware support for intermediate modes 38
3.2.4 Intermediate mode add and multiply algorithms 43
a) Addition 43
b) Vector mode multiplication 43
c) Intermediate mode multiplication 46
3.2.5 Architecture of the bit-slice PE 47
3.3 Data storage in the array 47
3.3.1 Storage on existing processor-arrays 47
3.3.2 On-chip storage 49
a) Word store 50
b) Bit stores 54
3.3.3 Off-chip storage 54
3.4 Conditional operations 54
3.4.1 Global conditions - the status flag 54
3.4.2 Local conditions - activity control 56
3.5 Floating-point enhancements 57
3.5.1 Problems with SIMD floating-point operations 57
3.5.2 Basic floating-point algorithms 57
a) Multiplication 58
b) Addition 59
3.5.3 Hardware for sign-magnitude arithmetic 61
3.5.4 Hardware for mantissa normalisation 62
3.5.5 Mantissa alignment 65
3.6 A description of the RPA PE 65
3.6.1 Instruction and ALU control field 65
3.6.2 External RAM port and host interface 74
3.6.3 Activity stack 75
3.6.4 Bitstack 76
3.6.5 Wordstack 76
3.7 RPA performance estimates 82
3.7.1 Integer addition 82
3.7.2 Integer multiplication 83
3.7.3 Floating-point routines 84
3.7.4 Applications 85

4 Implementation of the RPA processor chip 89
4.1 Choice of technology and design rules 89
4.1.1 Choice of manufacturing technology 89
4.1.2 Generic design rules 90
4.2 Design methodology 92
4.2.1 Clocking scheme 92
4.2.2 Layout of 16 PE chip 94
a) Exploiting regularity in the layout 94

b) Micro-control bus distribution 94
c) Neighbour interconnect and off-chip RAM 96
4.2.3 Layout of 1 PE chip 96
a) The latch element 100
b) Pass logic 104
4.2.4 Design for testability 104
a) Go/no-go testing 105
b) Debug testing 105
4.2.5 Gate Matrix Layout 108
4.2.6 Simulation and modelling 110
a) Modelling restoring logic 110
b) Modelling interconnect 112
4.3 Design of the PE components 113
4.3.1 The Bitstack and Activity stack 113
4.3.2 The Wordstack 117
a) Static RAM 117
b) Barrel shifter 125
c) Registers 125
d) Operand bus interface 129
4.3.3 The ALU and bit-slice control block 129
a) The ALU 129
b) Neighbour interface 136
c) Result bus switches 136
d) Off-chip RAM port 136
e) Assembling the ALU and bit-slice control block 139
4.4 Chip packaging 139

5 Conclusions 145
5.1 Summary 145
5.2 Evaluation of the architecture 146
5.3 Future directions 148

References 151

Appendix A - A register transfer description of the PE and chip 161
A.1 The specification language 161
A.1.1 Control structures 161
A.1.2 Manipulation of variables 162
A.2 Description of the PE and chip 164
A.2.1 Neighbour communications network 164
A.2.2 PE instructions and ALU functions 166
A.2.3 External RAM-port and host RAM interface 170
A.2.4 Activity stack 171
A.2.5 Bitstack 172
A.2.6 Wordstack 172

1 Introduction

It was in 1909 at the Cavendish laboratory, Cambridge, that they drank a toast 'To the electron, may it never be of use to anyone'

Cyril Connolly
Observer 1977

This book describes the development of a parallel computer of the Single Instruction-stream, Multiple data-stream (SIMD) class. It is a type of SIMD computer known as a processor-array; the computer described in this book is known as the Reconfigurable Processor-Array or RPA.

Processor-arrays have established themselves as an inexpensive form of parallel computer, which are suitable for a wide range of highly parallel applications. They achieve their performance by huge replication of simple processors, known as Processing Elements or PEs, rather than using a small number of complex processors or special functional units. This performance is achieved without recourse to special circuit techniques such as pipelining or special high-speed technologies such as Emitter-Coupled Logic (ECL). Furthermore, in a processor-array, only the processing and data memory circuits are replicated. This is because the control circuits and program memory are shared by all of the PEs in the array; the PEs all perform the same instruction which is broadcast to them by the single controller, but on different data.

Because processor-arrays are highly replicated structures, they are excellent candidates for large or very large scale integration (LSI or VLSI). By integrating one or more PEs onto a single chip, a complete processor-array can be built using one chip type. This allows development and fabrication costs for this single chip type to be spread over a large number of chips, even if only a few machines are built. This is an important consideration in the low-volume scientific applications market.

Unfortunately there are some situations in which the small-grain parallelism of a SIMD architecture is inefficient. The purpose of this book is to investigate enhancements to the conventional bit-serial PE, which improve its performance in these situations whilst still maintaining the advantages of flexibility and simplicity.

One of the problems is that the large, fixed, parallelism of a processor-array can only be fully exploited by an equally large parallelism in the data to be processed. If the data being processed does not have this parallelism, some PEs are not used, with a proportional decrease in potential processing performance. The single instruction stream of a processor-array means that these wasted PEs can not be used for another task unless there are many identical tasks to be performed.

A processor-array is most viable if one or more PEs are integrated on a single LSI or VLSI chip. There are already several processor-arrays of this type on the market which will be discussed in chapter 2. However, these designs have also tended to depend on commercial RAM chips for the PE storage, and so a memory-processor bottleneck exists, limiting the processing speed to the bandwidth of the memory chips. If temporary storage is incorporated onto the chip then performance can be immediately improved with only minor alterations to the processor architecture.

An alternative way of looking at this problem is to consider the number of buses feeding data to the ALU. For example, for a typical arithmetic operation, there will be two operands

passing from memory to the processor, and one result passing from processor back to memory. Therefore a 3-bus architecture could conceivably achieve maximum ALU throughput. If less than 3 buses are used, an accumulator architecture results in which a large proportion of time is used transferring data between main memory and the accumulator. Thus, by adopting conventional RAM as the only PE store with only one bus to the ALU, the operations to read the operands from memory and to write the result to memory have to be carried out sequentially, with an immediate penalty in ALU throughput. Multiple-bus memory access requires only that some memory is integrated onto the chip.

It can be shown that, for simple operations, a large array of bit-serial PEs is a better source of processing power than a small array of complex processors, due to its flexibility [2.5]. However, there are some areas in which this argument breaks down and bit-serial processors can not possibly compete with specialised hardware. Examples of this problem are found in integer multiplication and nearly all floating-point operations. However, it is possible to improve these operations with relatively simple special-purpose circuits.

This summarises the areas of processor-array architecture that are researched in this book. Solutions to these problems are developed and a PE architecture incorporating these new ideas is proposed. This architecture has been implemented as a chip design to investigate its suitability for VLSI implementation.

The RPA project was a group project involving five people, each of whom had different responsibilities. Chris Jesshope was the instigator of the project in 1983 and has been its supervisor throughout. The project is split into four areas: simulation and program development software by Jim Stewart [1.3, 1.4], applications programming by Russell O'Gorman [1.8, 1.11], control path design by Adriano Cruz [1.6] and data path design by myself. This book covers the data path design of the RPA. However, results of other team members' work have influenced the architectural design of the PE. In particular, feedback from the simulations and the applications programs have helped considerably in the development of the architecture. Some of this work has been reproduced in sections 3.1 and 3.7 of this book for completeness.

The main text of the book is in three large chapters and there is also a conclusion and an appendix.

Chapter 2 reviews the history of parallel processing from the first tentative suggestions in the mid-19th century to the explosion of interest in parallel processing in the late 1970s and early '80s. The different types of parallel computer are described and the reasons for choosing a processor-array from amongst the wide range of options available to the computer designer are discussed. The chapter concludes with a brief summary of applications followed by case studies of two commercially available SIMD arrays: ICL's Distributed-Array Processor (DAP) and Thinking Machines' Connection Machine.

Chapter 3 discusses the high-level organisation of the RPA and investigates the disadvantages of conventional processor-arrays. A number of problems are identified and possible solutions to each problem are proposed. A PE architecture is developed from these proposals which incorporates all of the proposed improvements. The chapter concludes with a functional description of this architecture and estimates of its processing performance for some standard arithmetic functions.

Chapter 4 describes an implementation of the PE architecture developed in Chapter 3. The implementation is a VLSI chip containing 16 PEs. The description takes a top-down approach, introducing and discussing issues as they become relevant. The description culminates in a brief

overview of the circuit design and layout of the individual PE components. Finally, packaging of the 16 PE chip is discussed.

Chapter 5 is the conclusion of the book and suggests areas of further research on processor-array architectures.

There is also one appendix containing a high-level formal description of the PE in a register-transfer language.

2 Introduction to parallel computers

The historical sense involves a perception not only of the pastness of the past, but of its presence

TS Eliot
Tradition and the individual talent
1919

We have constantly to check ourselves in reading history with the remembrance that, to the actors in the drama, events appeared very different from the way they appear to us. We know what they were doing far better than they knew themselves

Randolph Bourne
Youth and Life

2.1 A brief history of parallel computers

The history of computing has been shaped by the technology available to the engineer. Computers have been limited in performance by the state of the art of technology, and by the cost of that technology; indeed, designers think within the confines of what is possible with existing techniques [6.2]. Thus Charles Babbage conceived many of the principles of what is now called a von Neumann computer but in a purely mechanical form. He didn't build his computer - the Analytical Engine - because of the prohibitive cost of the precisely machined components.

Two characteristics of electronic component technology have brought about the interest in parallel processing. One is the rapid decrease in cost of a particular level of technology once it has become established, making that technology economic to use in large volumes. This makes it more attractive to build a computer out of replicated state of the art components than to attempt to advance the state of the art. The second is that computing power has had to expand faster than circuit speed could allow, to supply the rapidly increasing (and accelerating) demand [2.7]. Some of the increase in computer power has been achieved by developing existing architectures - for example by increasing the word-length of the processor. Some has been achieved through better algorithms. However, it has become apparent that only through the exploitation of parallelism can the future demands of users be met.

This section traces the development of the Single Instruction stream, Multiple Data stream (SIMD) class of parallel computers, in which only the data processing components are replicated. These Processing Elements (PEs) share a single controller. The arguments for choosing this type of parallelism will be discussed later in this chapter.

2.1.1 First proposals

The first notable reference to multi-processing was by Menabrea in 1842 [2.10] in his report on meetings with Babbage in Turin. These proceedings were published in French but were

translated and annotated by Ada Augusta, Countess of Lovelace - who took a great interest in Babbage's work and worked on the programming of the Analytical Engine [2.9]. The suggestion was made that in repetitive work, several results could be calculated simultaneously. How this was to be achieved within the design of the Analytical Engine was never explained.

The Analytical Engine was never built, although the processing unit was built after Babbage's death. The ambitious nature of Babbage's ideas was a contributory factor in this demise - the Analytical Engine was to have a 50-digit decimal word with a 1000 word store. Each word of the store was a column of numbered disks with one digit per disk and the central processor - or 'mill' as it was called - worked on the whole word in parallel using a ripple carry technique. Babbage's machines were purely mechanical devices and required accurately machined components - the expertise needed to manufacture these components did exist as demonstrated by the examples of the Difference Engine (a calculator as opposed to a computer) which were built. However the expense would have been enormous and the design of the Analytical Engine was mainly an intellectual exercise.

The work of Babbage was not surpassed until the advent of electronics revolutionised computers a century later. A good survey of this early period is contained in Bell and Newell [2.4]. The first stored program computer to go into operation was the Manchester Mark 1 in 1948 [2.11, 2.12]. These early electronic computers were very simple to minimise the number of components, but it was realised that the new technology of electronics could make parallel processing possible.

The first electronic computers used vacuum tubes as their main active component, though the storage medium had to be a different technology to achieve sufficient capacity. The Mark 1 used a cathode-ray tube (CRT) as its storage medium [2.13]. Data was written to the screen as dots and dashes which were retained for tens of milliseconds by the phosphors; periodic refresh was needed to retain the data indefinitely. The CRT storage had a number of benefits - it was a random-access memory and stored words in bit-parallel format. Initially this storage medium had a capacity of 32 words of 32 bits with addressing facilities for up to 256 tubes (a possible capacity of 8k words). Other machines used mercury delay-lines to store their data (e.g. the Univac [2.14] which had 100 10-word long delay lines as its main store). These were bit-serial, word-serial storage devices so programs had to take into account the inherent latency of the store. Delay lines did not have the capacity of the CRT storage nor the convenience of random access. Furthermore they needed to be made to very exact tolerances of delay, whereas the CRT design was far less critical.

Although von Neumann suggested in 1952 that a square array of identical processors could be used as a general automaton [2.15], the first hardware design of an electronic parallel processor is attributed to Unger [2.16]. His 1958 proposal suggested a 2-dimensional grid of bit-serial processors as an architecture for spatial problems such as image processing. Each processor was connected to its four nearest neighbours to allow inter-processor communication - a natural choice for a computer working on 2- or 3-dimensional space. The paper included circuit diagrams and some elementary picture-processing algorithms. However, he acknowledged that the state of the art technology - thin-film components and ferrite core storage were just being developed - made the construction of such a machine an awesome task. Unger's computer wasn't built but it is commonly considered to be the common ancestor of all subsequent 2-dimensional grid architectures.

2.1.2 The first word-slice parallel computers

The SOLOMON computer design of 1962, described by Slotnick and Borck in [2.18], went one step closer to completion, with some of the circuits being constructed [2.19]. By this time small-scale integrated (SSI) bipolar technology was being used with one latch or gate per package. Using this technology a SOLOMON PE fitted on a single circuit board. The proposal called for a minimum system of 256 PEs arranged as a 32x8 array. As in Unger's design, each PE was connected to the four nearest neighbours. This connectivity carried over the array boundaries - the rectangular array could be configured as either a rectangle, a cylinder or a torus by connecting appropriate array edges together.

The SOLOMON PE consisted of a bit-serial ALU, circuitry to control the routing network, local memory and local on/off control (activity control) so that each PE had the option of obeying or ignoring instructions depending on the value of local data.

Although the SOLOMON computer was not quite completed, the expertise gained in designing it led to the development of the ILLIAC IV. This was an extremely important computer and had a strong influence on all aspects of computer engineering. Work started on the ILLIAC IV in 1965 under the leadership of Daniel Slotnick [2.20, 2.21, 2.24, 2.25]. Although based on the SOLOMON architecture, the ILLIAC IV was a much more powerful machine. Technology had advanced by the mid-60s to the stage of SSI bipolar logic and thin-film memory, but the design of the ILLIAC IV set out to push the technology forward. The proposal called for the use of Medium Scale Integration (MSI) ECL throughout, with about 20 gates per chip. A complete new logic family was to be developed by Texas Instruments for this purpose. This ambitious approach was the main cause of ILLIAC's problems which resulted in a time over-run of about 5 years and a reduction in the performance of the eventual machine by an order of magnitude over the proposal.

The initial design for the ILLIAC IV called for 256 64-bit floating-point processors giving a hoped-for total performance of 1 Gflop. The PEs were divided between 4 quadrants, each of which was an array of 64 PEs working in synchronisation. Thus each quadrant was a SIMD computer. These quadrants could be synchronised to make a 256 PE SIMD array or they could be independent - making a Multiple-SIMD (MSIMD) architecture. The PEs were connected in a string with extra connections to neighbours 8 PEs away in either direction. This form of interconnect is essentially a square array with the edges connected as a torus but with the row wrap-round connections offset by one row to make a spiral. The ends of the spiral could be connected together or they could be used as the array's I/O ports.

Only one quadrant of the ILLIAC IV was ever built due to cost and time overruns. It was delivered to NASA Ames Research Centre in 1972 although reliable service was not achieved until 1975 after extensive hardware debugging work [2.23].

Although the ILLIAC IV didn't achieve the objectives of the original proposal, the single quadrant machine - with a performance of about 50 Mflops - was still one of the most powerful computers available at the time (although the CRAY 1 was released in 1976 and has an estimated performance of over 100 Mflops, see Hockney and Jesshope [2.5]). It was used to perform the aerodynamic simulations required by NASA and a lot of work was done on parallel languages and algorithms to exploit the architecture fully [5.1, 5.2, 5.3, 2.22]. As well as being influential in these fields, the construction of the ILLIAC IV pushed forward two important technologies in the field of micro-electronics. Texas Instruments (TI) was contracted to produce a family of MSI ECL chips and although development problems with these chips meant that they were not ready on time to be used in the ILLIAC IV they made ECL a usable technology

for high-speed logic. Nevertheless, the ILLIAC did use ECL (SSI) throughout and was the first computer to do so. The other development that was pushed ahead was the bipolar RAM chip. ILLIAC IV was intended to use thin-film memories but the delays in TI's work on the ECL chips led to space problems on the circuit boards (more chips were needed for the logic than had been planned) and so a more compact memory technology was sought. Fairchild was at the time developing bipolar RAM chips and they eventually supplied the 256-bit RAMs used throughout the ILLIAC IV. This made it probably the first mainframe to boast all-semiconductor main memory.

The design of the circuit boards - 12-layer PCBs - required the development of a design-automation system which was also one of the first of its kind.

Burroughs was the main contractor for the ILLIAC IV project and they went on to develop their own parallel computer known as the PEPE [2.26, 2.27] which was an ensemble (an array with no communications network) of 288 associative processors intended for radar processing tasks for the defence industry. The project started in 1968 and by 1971 the prototype PEPE had been built using standard TTL logic. PEPE was redesigned using custom LSI ECL chips for the production version released in 1976. The 32-bit floating-point PE contained 80 chips of 12 types plus memory (32 1kbit chips). Of these 12 different designs, 9 were custom LSI chips and the rest were semi-custom chips. This marked a massive improvement over the prototype, with the PE occupying a twentieth of the area of the TTL implementation due to a twenty times reduction in chip count.

The experience gained in these two projects was put to use in the design of a new type of parallel computer, the Burroughs Scientific Processor (BSP) [2.28, 2.29, 2.31]. Work on this project started in 1973.

The BSP used a different approach to parallelism than previous parallel computers. Instead of replicating the processor-memory combination, the BSP had separate arrays of processors and memory units connected by two crossbar switches - one switch for each direction of data flow. Sixteen 48-bit floating-point processors were used with seventeen memory units - Burroughs used a system of prime memory banks to minimise the risk of conflicts (several processors trying to access the same bank) when working with matrices [2.30].

A prototype BSP was built in 1978 and this had a performance of about 50 Mflops - about the same as the ILLIAC IV. However, the project was cancelled after completion of this prototype and no commercially available BSPs were released. Again one of the problems seems to have been the choice of technology, despite Burroughs' intention to keep the design within the limits of the state of the art at the time. The design used an existing and well-tried MSI ECL family mounted on standard circuit boards as used in the Burroughs range of mainframes. However, they experimented with the new technology of CCDs (Charge-Coupled Devices) to build a fast file store with a capacity of 8-64 Mwords, but it was prone to high data error rates. This was later changed to MOS RAM but the project was scrapped before going into production.

Burroughs are still looking at parallel computers for the scientific community - they proposed one of the two designs commissioned by NASA for the Flow Model Processor at their Numerical Aerodynamic Simulation Facility (NASF) [2.32, 2.33] (the other was a pipelined architecture from CDC). NASA's interest stems from fluid dynamics calculations which are essential for the simulation of space vehicles in the atmosphere. The proposal was for a 512 processor machine with a performance around 1 Gflop. The architecture is similar to the BSP in that it consists of separate processors and memory connected by a multi-stage switch. The

memory is organised as a prime number of banks (521 banks in this case) to avoid access conflicts in the same way as in the BSP.

2.1.3 Bit-slice architectures

In parallel with the development of these large processor-arrays with word-length processors, the concept of bit-serial parallel processing was being developed. In 1960 Shooman suggested a design for a conventional computer which could also address memory in columns instead of rows to perform bit-serial calculations. This he dubbed the Orthogonal Computer [2.17]. He realised that processing data vertically not only used simpler hardware due to the avoidance of carry propagation problems, but also that there was greater flexibility both in the number of bits processed simultaneously and in the word length of the data being processed. Furthermore he realised that conditional branching was not possible on such an architecture, so included provision for masking off bits during a calculation - a feature which is now known as Activity control after the terminology used for the ICL DAP.

Shooman's ideas were put into practice with the development of the STARAN associative processor between 1962 and 1972 [2.34, 2.35]. STARAN is an array of 256 simple bit-serial processing elements (providing logic functions only) connected through a multi-stage switch called the flip network to a 256 bit wide multi-dimensional store. This store has a wide range of access modes including bit-slice mode (1 bit from each of 256 consecutive words), word mode (one 256-bit word) and other intermediate modes (for example, one byte from each of 32 consecutive words). The power of the multi-dimensional memory plus the flip network means that most data manipulation tasks only use the PE for temporary storage. Thus the simplicity of the PE is not a handicap for the type of operations for which the STARAN is intended - like database management, text searching and image processing. The system can be built up of any number of basic array units up to 32 - the systems described in the literature have 2 or 4 arrays - all sharing the same controller. The technology used for the STARAN is off-the-shelf ECL logic and 256-bit bipolar RAM chips. A later, airborne, version of the STARAN was proposed in 1978 [2.43]. This took advantage of developments in micro-electronics by using custom CMOS/SOS (Silicon On Sapphire CMOS) LSI chips for the PEs and flip network. 32 PEs and a 32-bit wide flip network were fitted onto the custom chip; the memory was implemented as a hybrid containing 16 1kx4 MOS RAM chips - two hybrids were associated with each PE chip.

A British machine developed from the SOLOMON and Unger designs is the ICL DAP (Distributed-Array Processor) [2.36, 2.37]. The DAP project started in 1972, with a prototype completed in 1976 and the first customer model delivered to Queen Mary's College, London, in 1980. The DAP uses bit-serial processing elements, like its ancestors, arranged in a square array of 64x64 PEs (32x32 in the prototype). Each PE is connected to its four nearest neighbours. The DAP is one of the examples described in more detail in section 2.5.

Meanwhile, the work on the STARAN by Goodyear Aerospace developed, after a brief examination of electro-optics [2.39], into the Massively Parallel Processor (MPP). This was first described in 1977 [2.40], but the design reached its final form in about 1979 [2.41, 2.43]. Although based on the STARAN associative processor, the MPP is also very similar to the SOLOMON and the DAP. Goodyear were contracted to build the MPP by NASA to process satellite images from their LANDSAT satellites.

The MPP is a straightforward processor-array with a 128x128 array of bit-serial PEs, each of which has an ALU plus 1k of RAM. The flip network and multi-dimensional memory of the