VLSI ARCHITECTURE

Philip C. Treleaven and Safwat A. Mansi

Department of Computer Science
University of Reading
Reading RG6 2AX
England

ABSTRACT

The veritable explosion of current research into novel VLSI architectures is radically affecting the design of microprocessors and microcomputers. A whole spectrum of VLSI processors are under development based on the Mead-Conway design philosophy of simplicity, regularity and replication. These VLSI architectures range from special-purpose: including dedicated single processors such as Lyon's Optical Mouse chip and adaptable multi-processors such as Systolic Arrays, to general-purpose: including single microprocessors such as the Berkeley RISC and multi-processors such as the INMOS TRANSPUTERS. This brief survey of VLSI architectures will illustrate the exciting work going on in the area.

1. VERY LARGE SCALE INTEGRATION

Most of the technological achievements of the past decade have depended on microelectronics. Despite this the architecture of microprocessors and microcomputers has hardly changed. In the evolution of microelectronics we are now at very large scale integration (VLSI); in simple terms a chip containing over 100,000 devices.

However, VLSI technology has very different properties from the earlier microelectronic technologies. One important property of VLSI is the smaller feature size, giving benefits in higher density and speed, together with low switching energy and cost. These benefits are well understood [4,19]. In the evolution from 5 micron to 0.25 micron technology (by the end of this decade) we can expect an improvement in the switching elements in the order of two or more [15].

Another important property of VLSI technology is that the cost relationship between communication and switching has reversed [19]. Communication is more expensive and limiting in VLSI in at least four ways, as pointed out by Seitz [24]. Firstly, the wires occupy most of the area on a chip in contrast to transistor switches that rarely occupy more than a few percent of the area. Secondly, the delay in a Metal Oxide Semiconductor (MOS) logic element or driver is principally that induced on the element by the capacitance of the wire to be driven. Thirdly, the energy dissipated is likewise directly related to the capacitance and its dominant parasitic component. Lastly, the product of the resistance and capacitance per unit length of a wire scales so that only fairly short wires on a submicron feature size chip can be realistically regarded as delayless. We must therefore conclude that communication limitations are a fundamental and permanent characteristic of VLSI design and architectures. Thus high bandwidth communication is possible only between parts of a system that are physically close to each other.

In summary, the properties of VLSI are:

1. _Design complexity is critical_ and requires careful design management based on simplicity, regularity and replication.

2. _Wires occupy most space on a circuit_ and therefore equal attention must be given to the design of wiring as to components.

3. _Non-local communication degrades performance_, becoming progressively expensive in chip area, signal energy and propagation.

These properties of VLSI technology, notably the sheer complexity of utilising a hundred thousand logic gates, has led to the Mead-Conway VLSI design and implementation philosophy. Their VLSI philosophy has three important constituents. Firstly, the Mead and Conway VLSI design style documented in "Introduction to VLSI Systems" [19]. Secondly, the Multiproject chip concept [4] which places a number of separate designs on a single chip so as to reduce implementation costs. Lastly, the so-called Silicon Foundry [12] able to fabricate small quantities of designs from different sources, operating VLSI implementation like a photographic film processing service for its customers.

In the context of the above properties of VLSI and the Mead-Conway philosophy, the remainder of this paper examines the special-purpose and general-purpose architectures designed to exploit VLSI.

2. MACHINE ARCHITECTURES

Having briefly examined the properties of VLSI we are now in a position to list the criteria, and their related advantages, of "good" VLSI architectures.

Firstly, the architecture must be implementable by only a few different types of simple cells and simple chips. Since most of the cells and chips are copies of a few basic ones, only a few different simple cells need to be designed and tested. Kung [13] states that exactly how simple a cell should be can only be answered on a case by case basis. For instance, if a whole system is to be implemented on one chip, then each cell would probably contain only simple circuits and a small amount of memory, whereas for board implementations it is likely that each cell may approach the complexity of a simple microcomputer.

Secondly, the data and control paths of the architecture should be simple and regular. Cells may be connected by a network with local and regular interconnections, thus avoiding or minimising long distance or irregular communication. Regular interconnection implies that the design can be made modular and extensible, so a large computer system can be implemented by combining the designs of simple cells and simple chips.

Thirdly, the architecture's communication should be localised. Ideally there will be no global communication, only communication between adjacent cells and chips. This may be viewed as "locality of reference" applied to VLSI.

Fourthly, the architecture must utilise extensive concurrency for performance. High performance is obtained in VLSI architectures from the concurrent operation of a large number of simple (cells and chips) processors rather than from a single large processor. This concurrency may be obtained either by pipelining the stages involved in a computation or by multiprocessing independent computations in parallel.

Lastly, the architecture should make multiple use of each input data item. To balance input/output bandwidth with computation, it is necessary to make multiple use of input data. As Kung [13] has pointed out, this can be achieved either by broadcasting the input data to all cells, or by having the data traverse a regular structure of cells.

A whole spectrum of processor architectures is under development based on this VLSI philosophy of simplicity, regularity, and replication. These architectures range from special-purpose: including dedicated single processors such as Lyon's Optical Mouse chip [17] and adaptable multiple processors such as Kung's Programmable Systolic Array chip

[5], to general-purpose: including single microprocessors such as the Berkeley RISC [20,21] and multi-processors such as the INMOS TRANSPUTERS [1].

An interesting taxonomy [24] for this spectrum of VLSI architectures is proposed by Seitz. This is shown in Figure 1. The horizontal axis shows the "Processor" size which ranges from a replication element as small as a RAM or shift register cell, to a large programmable processor. The vertical axis shows the number of (often identical) processors that make up an architecture.

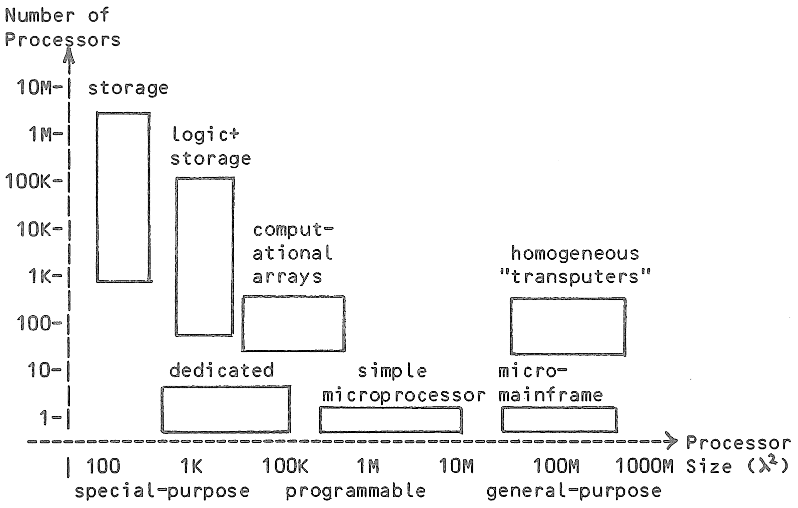

Figure 1: Taxonomy of VLSI Architectures

In a tour of these zones from left to right, one starts with Storage systems, such as random-access memories and shift registers, in which the replication cell is about as small as usefully imaginable. Next are what Seitz calls Logic+storage systems [24], systems in which a small amount of logic is associated with a few hundred bits of storage. A structure made attractive by the logic and storage being implemented in the same technology. Example applications include sorting and association etc.

In the next zone are Dedicated processors, single special-purpose chips built for a specific application such as the RSA cipher chip [23]. These processors have the advantage of a wired-algorithm so-to-speak. Next comes Computational Arrays which perform highly concurrent numerical computations. The processors are connected in regular patterns that match the flows of data and control in the target computation, and usually are capable of operations such as addition and multiplication. Typical applications are signal and image processing.

To the right come the general-purpose processors. Zones with single processors: such as simple microprocessors based on the reduced instruction set concept, and Micromainframes [8]. The distinction between these two zones is partly one of programming style and partly one of engineering. Next comes zones with multiple processors, including networks of "transputer" microcomputers. The expectation for these multi-processor systems is that computations can be distributed across a system by a compiler and loader so that potential concurrencies can be exploited at execution time.

Below we examine examples of these special-purpose and general-purpose VLSI architectures to illustrate the exciting work.

3. SPECIAL-PURPOSE ARCHITECTURES

A dedicated special-purpose VLSI architecture is designed either to form an integral part of a digital system or as a peripheral device to be attached to a conventional host computer. It can be a single processor chip built from a replication of simple cells, or a multi-processor system built from identical simple chips, or even a combination of these two approaches. Special-purpose processors have a number of advantages [26] that help reduce their cost:

1. Only a few different simple cells need to be designed and tested since most of the cells and chips are copies of a few basic models.

2 Regular interconnection implies that the design is modular and extensible, so one can create a large processor by combining the designs of small cells and chips.

3. Many identical cells and chips that use pipelining and multiprocessing can meet the performance requirements of a special-purpose processor.

Besides the excellent special-purpose architectures below, many equally novel processors are found in the literature [7,14,22].

3.1. Single Processors

The two examples of special-purpose single processors briefly examined below are the RSA Cipher chip [23] and the Optical Mouse chip [17].

RSA Cipher Chip

This chip, designed by Ronald Rivest and his colleagues at the Massachusetts Institute of Technology, implements a public-key encryption algorithm. This operation is computationally demanding, requiring up to several hundred multiplications of several hundred bit numbers. The chip was designed as a general-purpose, big-number processor, using a bit-slice architecture for the ALU. It is a good example of a chip design based on the duplication of simple cells.

Externally, the chip is configured as a memory chip that can be read or written at one of four eight-bit word positions. For instance, one of these positions is the "window" for data I/O, while another is for receiving commands such as "encrypt." To support encryption, the RSA chip has a 512-bit ALU organized in a bit-slice manner. It has eight general-purpose 512-bit registers, as well as up-down shifter logic and a multiplier. Other sub-systems include control logic (containing a PLA of 224 72-bit microcode words), a stack/counter array for subroutines and loops, and an array of powerful superbuffers to drive the signals that control the ALU.

Internally, the ALU is only capable of performing the operations $(A * B) \pm C$, shift-left, shift-right, and test least-significant bit. The remaining required operations are implemented by microcode control subroutines. These operations include: RSA encryption/decryption (modular exponentiation), generating a large prime number, generating a complete RSA key-set, greatest common divisor, and input or output of a large number through the eight-bit window.

The "floor plan" of the RSA cipher chip is shown in Figure 2. The left side contains a block of 320 slices of the ALU and the upper-right block contains the remaining 192 slices. The central spine carries control signals to the ALU from the superbuffer driver array at its lower right. The microcode PLA occupies the right-center area of the chip, and the remaining logic (stack, pads, etc.) occupies the lower-right portion. In Figure 2, S denotes a stack, X an eight-bit "window," C a small bus-control PLA, and D some debugging logic.

```
 --------------------------------
|         V D D                  |
|-----------------------------   |
|            |||              |  |
|            |||    A L U     |  |
|            |||  192 cells   |  |
|            |||              |  |
|            |||--------------|  |
|  A L U     |||  ------------|  |
|320 cells   ||| | microcode  |  |
|            ||| |  P L A     |  |
|            ||| |            |  |
|            ||| | --------------|
|            ||| ||D|------------|
|            ||| ||R|  -   -  |  |
|            ||| ||I| |C| |S| |  |
|            ||| ||V|  -   -  |  |
|            ||| ||E| |D| |X| |  |
|--------------- ||R|  -   -  |  |
|         G N D  |------------   |
 --------------------------------
```

Figure 2: RSA Cipher Chip Floor Plan

The entire chip contains about 40,000 MOS transistors, has 18 pins, runs at 4MHz, and uses a little more than one watt of power. Rivest estimated that the project required about five man-months of effort, with the chip finally being fabricated in the fall of 1979. It was primarily a programming project because he and his colleagues almost exclusively wrote programs in LISP, which when executed, created the desired Caltech Intermediate Form output file. Altogether, they wrote about 75 pages of LISP code. The largest pieces specified the final placement and interconnection of the modules (14 pages), the description of the ALU (11 pages), and the microcode assembler and simulator (10 pages). Estimates are that the chip can perform RSA encryption faster than 1200 bps (even faster if shorter keys are used), and the designers predict speeds of 20,000 bps within a few years.

Optical Mouse Chip

Dick Lyon investigated [16, 17] new VLSI architecture methodologies for applications such as signal processing and smart digital sensors. A novel example of the latter is the Optical Mouse chip, designed in the VLSI system design area at Xerox Corporation's Palo Alto Research Centre.

The Optical Mouse is a pointing device for controlling the cursor on a personal workstation display, such as the one found on the Xerox Star 8010 information system. The design was motivated by the desire for a highly reliable mouse with no moving parts except button switches, and it was realised through the innovative use of electro-optics, circuit design, geometric combinatorics, and algorithms - all in a single special-purpose sensor chip.

This chip reports the motion of visible spots relative to its coordinate system by combining two techniques. One is a simple "mostly digital" circuit that produces digital image (bitmap) snapshots of bright features on a dark field using self-timed circuit concepts and mutually inhibiting light sensors. The other technique uses a tracking algorithm with an easy-to-track contrasting pattern, a detector array and inhibition network matched to the pattern, and a digital machine that inputs images of that pattern and track relative image motion. (An especially interesting aspect of the design is the integration of sensors, memory, and logic in a single array using standard MOS technology).

The basis of the sensors is that in nMOS, light striking the circuit side of the chip converts photons to hole-electron pairs; the holes are attracted to negative-biased p-type silicon substrates, while the electrons are attracted to n-type regions. A so-called dynamic node that has been positively charged will detect light by collecting a negative charge (electrons) and "leaking" to a lower voltage. An imager is simply an array of subcircuits, each consisting of a dynamic node, a transistor to reset the node to "high" and isolate it, and an inverter circuit to sense the voltage of the node and communicate it to other circuits.

Figure 3 shows the floor plan of the optical mouse chip. "Mouse cells" represent the four-by-four, two-dimensional sensor array, and "Tracker PLA" tracks spots by comparing images and outputting X and Y movements, as well as outputting "Any-Good" and "Jump" counter control and test signals. The "X counter PLA" and the "Y counter PLA" control up/down counting and transmit "XA XB XL" and "YA YB YL" coordinates to the host computer.

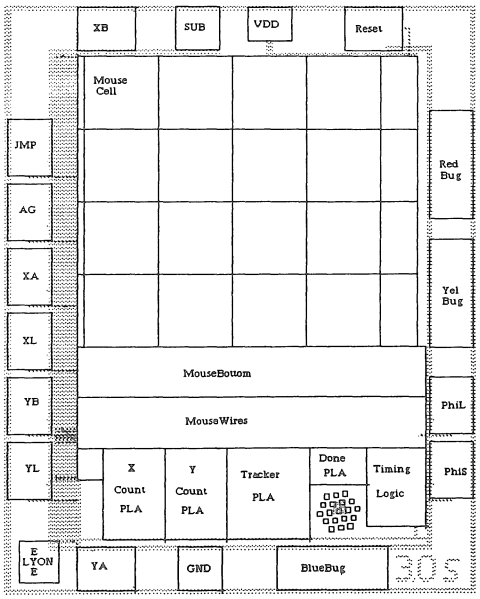

Figure 3: Optical Mouse Chip

The layout style used in this first version of the chip treats a sensor cell with its logic and memory as a low-level cell and constructs the array by selectively programming the cells in different positions. The resulting chip is about 3.5mm x 5.4mm in a typical nMOS process (Lambda=2.5 microns, or five-micron lines). A second version of the chip has also been designed to improve sensitivity.

3.2. Multi-Processors

Special-purpose multi-processor architectures are constructed from identical specialised processors. Some of the processors are dedicated logic, others are programmable to some degree. The processors are typically capable of operations such as addition and multiplication, and are connected in regular patterns that match the stream lines or wave fronts of the calculation. This is one of the most rapidly expanding areas of VLSI architectures because it remains one of the few proven multi-processor approaches for exploiting VLSI. The two examples examined here are the Geometry Engine [3] and the Programmable Systolic Chip [5].

Geometry Engine

This vector function unit, designed by James Clark when at Stanford University, performs three of the common geometric functions of computer graphics: transformation, clipping, and scaling. A single-chip version is used in 12 slightly different configurations to accomplish 4x4 matrix multiplications; line, character, and polygon clipping; and scaling of the clipped results to display device coordinates. This unit is an excellent example of a special-purpose device built from identical chips.

When configured as a four-component unit, the Geometry Engine allows simple operations on floating-point numbers. Each of its four identical function units has an eight-bit characteristic and (currently) a 20-bit mantissa. It operates with a simple structure of five elements: an ALU, three registers, and a stack. This basic unit can perform parallel additions, subtractions, and other similar two-variable operations on either the characteristic or the mantissa. Since one register can shift down and one can shift up, it can also multiply and divide at the rate of one step per microcycle. The 12-chip system consists of 1344 copies of a single bit-slice layout composed of the five elements. Four pins on the chip are wired to indicate to the microcode which of the 12 functions to carry out, according to the chip's position in the subsystem organization.

Figure 4 shows the geometry subsystem described above. The terms "MM", "CLIP", and "SC" denote matrix multiply, clipping, and scaling. Each of the four "MM" blocks, for instance, is a separate Geometry Engine chip that does a four-component vector dot product. Although each multiplication is done at the rate of one partial product per microcycle, the matrix multiplier has 16 of these products simultaneously active. Thus, the total transformation time, which is the bandwidth limiting system operation, is about 12 microseconds.

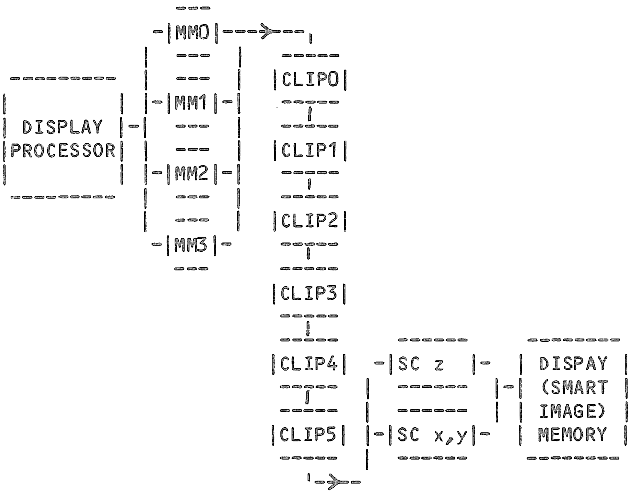

Figure 4: Geometry Engine data flow

In the actual design of the Geometry Engine almost all of the complexity was in the microcode that drives it. This microcode represents the logic equations for its finite-state machine, which are implemented in a PLA. Writing this microcode and making minor additions to the principal bit-slice to accommodate it took up approximately 50 percent of the total design time. Estimations are that the Geometry Engine is capable of performing about four million floating-point operations per second; 48 identical units, four per chip, will each do a floating-point operation in about 12 microseconds.

Programmable Systolic Chip

H. T. Kung and his co-workers at Carnegie-Mellon University are investigating the relationship of algorithm design to special-purpose chip architecture. A notable outcome of Kung's work has been the concept of "systolic arrays" [13]. Recently various computationally demanding problems have been solved using systolic algorithms. These systolic algorithms generally consist of simple processors, or systolic cells, that are connected in a regular pattern and operate on continuous, regular flows of data. To overcome the limited scope and hence the high design cost of a specialised systolic cell, the Programmable Systolic Chip (PSC) [5] has been designed to support many systolic algorithms. The PSC is tailored to the I/O and computational requirements of the family of systolic algorithms, so a large number of these chips can be used to implement a broad spectrum of algorithms.

A PSC processor [5] consists of five functional units (see Figure 5) that operate in parallel and communicate simultaneously in a pipeline fashion over three buses. The five functional units are: a 64x60-bit microcode dynamic RAM and a microsequencer, a 64x9-bit words DRAM register file, an ALU, a multiplier-accumulator (MAC), plus three input and three output ports. Data communication among the units takes place over three independent buses; control and status lines are separate. Each bus can be written by one of eight sources and can be read by any of ten destinations.

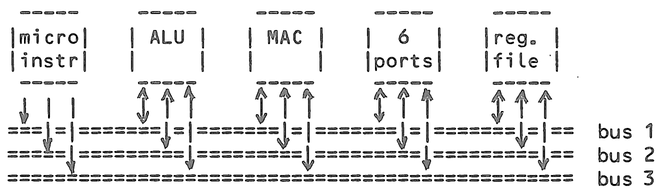

Figure 5: Programmable Systolic Chip

In order to keep the chip small and hence keep pinout and yields reasonable, an eight bit word size was chosen. Most data paths are, however, nine bits. The ninth bit being used to tag data, for control information, or as the most significant bit of a number modula 257 for coding applications. Each of the three input and three output ports are also nine bits; eight for data and the ninth for data or control. To keep the clock period short, interchip communication is pipelined with instruction execution. Thus a value which is computed in a given cycle can be used on the same chip in the next cycle, but not until the cycle after that on a neighbouring chip.

The PSC project, initiated in October 1981 has already led to a number of nMOS PSC processors being fabricated and tested. In the PSC the VLSI architecture problem of parallel programming is cleverly circumvented by limiting the programs supported to systolic algorithms and placing identical microcode in the DRAM of each systolic cell.

We will now consider general-purpose VLSI architectures.

4. GENERAL-PURPOSE ARCHITECTURES

Traditionally, the trend in designing microprocessors, and even mainframe computers, has been towards the use of increasingly complex instruction sets and associated architectures. However, the judicious choice of a simple set of instructions and a corresponding simple machine organisation can achieve such a high instruction rate that the overall processing power can exceed that of processors implementing more complex instructions.

In addition, a recent trend is for these simple processors to be designed so they can not only be used alone, but also can operate together as a parallel system.

Besides the interesting general-purpose architectures below, other novel processors are to be found in the literature [7, 9, 10, 25].

4.1. Single Processors

The two examples of general-purpose single processors briefly examined here are the MIT Scheme Chip [2, 11] and the Berkeley RISC [20, 21].

MIT Scheme-79 Chip

Two Scheme microprocessors namely Scheme-79 [11] and Scheme-81 [2] have been designed and built. Below the Scheme-79 chip which is more readily accessible in the literature is described. Notably, the Scheme-79 chip is a "landmark" of the Mead-Conway VLSI design philosphy, taking just five weeks to design and layout.

The Scheme-79 single-chip microprocessor, developed by Gerry Sussman and his colleagues at MIT directly interprets a typed pointer variant of Scheme [11], a dialect of LISP. To support this interpreter the chip implements an automatic storage allocation system for heap-allocated data and an interrupt facility for user interrupt routines implemented in Scheme. All compound data in the system are built from CAR and CDR pointer list nodes. Each pointer is 32 bits, comprising a 24-bit data field, a 7-bit type field and a 1-bit field used by the storage allocator. The type identifies the object referred to by the data field. This data field is either a literal, or it points to another list node.

The Scheme-79 chip, whose machine organisation is shown in Figure 6 implements a standard von Neumann architecture in which a processor is attached to a memory system. The processor is divided into two parts: the data path and the controller. The data path consists of a set of special-purpose registers, with built-in operators, interconnected by a single 32-bit bus. The controller is a finite-state machine that sequences through the microcode, implementing both the interpreter and garbage collector. At each step it performs an operation on some of the registers (for example, transferring the address in NEWCELL into the STACK register) and selects a next state based on its current state and the conditions developed within the data path.

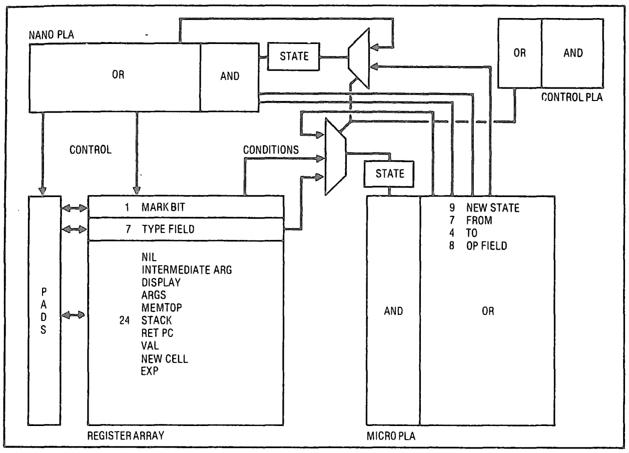

Figure 6: MIT Scheme-79 Machine Organisation

Ten registers in the chip have specialised characteristics. On each cycle, these registers can be controlled so that one of them is gated onto the bus and selected fields of the bus are gated to another register. For example, VAL holds the value of the last expression, EXP is the register for the current expression, and if this current expression is a procedure call, the arguments are evaluated and built into a list kept in the ARGS register.

The finite-state controller for the Scheme-79 chip is a synchronous system composed of a state register and a large piece of combinational logic - the control map. From the current state stored in the state register, the control map develops control signals for the register array and pads, the new state, and controls for selecting the sources for the next sequential state. The Scheme-79 chip interfaces to the external world through a 32-bit bidirectional data bus that specifies addresses, reads and writes (heap) memory, references I/O devices, reads interrupt vectors, and accesses the internal microcode state during debugging.

The Scheme-79 chip, using a process with a minimum line width of five microns, was 5926 microns wide and 7548 microns long, with a total area of 44.74mm^2. The entire project [11] including prototype tool building and chip synthesis was completed in five weeks.

<u>Berkeley Reduced Instruction Set Computer</u>

The reduced instruction set computer (RISC) [20, 21] design philosophy has been pioneered at the University of California, Berkeley. RISC machines combine a small set of often-used instructions with an architecture which is tailored to their efficient execution. In addition, a single-chip implementation of a simpler machine makes more effective use of the limited resources of present-day VLSI chips - such as the number of transistors, area, and power consumption.

The RISC I [21] and II [20] microprocessors are 32-bit, register-oriented machines designed in nMOS. (RISC I has 138 general-purpose registers, whereas RISC II has 198.) RISC I has 31 operation codes and RISC II has 39, they are mostly simple ALU and shift operations on registers. Both machines use 32-bit addresses and support 8-, 16-, and 32-bit data. They implement two simple addressing modes; indexed and PC-relative; synthesized from these are more complicated addressing modes, most instructions are executed in a simple cycle. The LOAD and STORE instruction - the only operations that access memory - violate this single cycle constraint; they add an index register and the immediate offset during the first cycle, performing the memory access during the next cycle to allow enough time for main-memory access.

Figure 7 shows the machine organization of RISC I. The machine naturally subdivides into the following function blocks: the register-file, the ALU, the shifter, a set of program counter (PC) registers, the data I/O latches, the program status word (PSW) register, and the control section (which contains the instruction register, instruction decoder and internal clock circuits). In addition, the register file needs at least two independent buses because two operands are required simultaneously. In Figure 7, buses A and B are read-only and bus C is write-only.

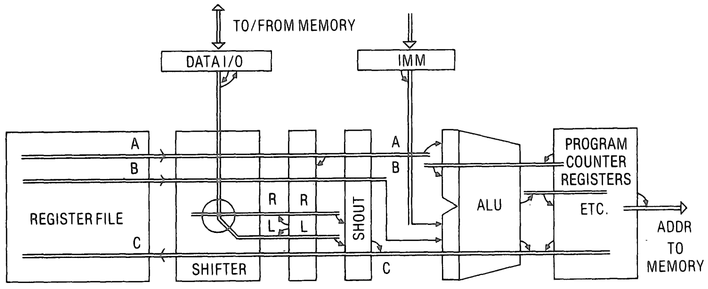

Figure 7: Berkeley RISC Microprocessors

An unusual feature of the two RISC machines is the so-called over lapped window registers, a fast and simple procedure calling mechanism using the general-purpose registers. A procedure has access to 32 register: global (registers 0-9 are shared by every procedure); low (registers 10-15 contain result parameters); local (registers 16-25 are used for local working); and high (registers 26-31 are input parameters). Each time a procedure is called, new registers are allocated in which the low registers of the calling procedure overlap the high registers of the called procedure.

Two different 4-micron nMOS versions of RISC I have been designed; the "gold" version, whose data path is shown in Figure 7, and a "blue" version. Both versions are similar in organization, however a more sophisticated timing scheme in the blue version shortens the machine cycle and reduces chip area. Details of the gold design include a chip size of 406 x 350mm, 44K devices, a design time of 15 man-months, and a layout time of 12 man-months. Power consumption for the chip is estimated at between 1.2 and 1.9 watts. RISC II is 25 percent smaller than RISC I with 41K devices.

4.2. Multi-Processors

General-purpose multi-processor VLSI architectures are constructed from identical simple microcomputers, usually based on the reduced instruction set philosophy. A micro-computer, containing a primitive processor and a small amount of local on-chip RAM, is able to operate alone as a sequential computer or, with others, as a component of a parallel system.

Two examples are examined below the RIMMS [6] and the innovative INMOS TRANSPUTER [1].

Reduced Instruction Set Multi-Microcomputer System

In the reduced instruction set multi-microcomputer system (RIMMS) [6], illustrated by Figure 8, an attempt was made to keep the design of the component microcomputer as conventional as possible.

```
 ------------------------------------------------------------
|                     global address space                   |
|           ----------------------------------------         |
|                                                            |
|     1:|           2:|           3:|           4:|   . . .  |
|    ---------   ---------   ---------   ---------           |
|   |simple   | |simple   | |simple   | |simple   |          |
|   |processor| |processor| |processor| |processor|          |
|   |---------| |---------| |---------| |---------|          |
|   |256 word | |256 word | |256 word | |256 word |          |
|   |memory   | |memory   | |memory   | |memory   |          |
|    ---------   ---------   ---------   ---------           |
 ------------------------------------------------------------
```

Figure 8: Reduced Instruction Set Multi-Microcomputer System

The central idea in RIMMS is that each microcomputer's local memory behaves as if it is a "bank" of a larger memory. To achieve this, each microcomputer forms part of a global (two-level) address space. An address (16-bits) consists of two parts: the high 8 bits define a specific microcomputer/process, while the low 8 bits define a subsidiary location. Each local memory provides storage for a program fragment (i.e. process) and a process can access any location in the global address space. In addition, a location may be used for "shared memory" or "message passing" communication of data. This is achieved by four memory operations: LOAD, STORE, PUT and TAKE. STORE and LOAD have traditional (shared memory) semantics, while PUT and TAKE support message passing. PUT may only overwrite an "empty" location and TAKE may only remove a non-empty value. Both operations "poll" a location until it is in the correct state.

Although any location in the global address space may be accessed, a process is always executed by its local processor and is executed (atomically) to completion. In addition, a process can activate another process, using a FORK instruction, as long as its processor is idle. (This atomic execution removes many of the synchronisation problems typically found in control flow multi-processors.) Finally, the processor implementation has a simple, 16-bit von Neumann data path, with only two visible registers (C - program counter, D - base register) holding the state of a process.

The RIMMS chip [6] is a simple, conventional 2-bus data path consisting of a register file, shifter, ALU, MAR/MDR registers, and a control PLA to implement the instruction set. The register file contains 7x16-bit registers and the ALU has two 16-bit input and two output registers. The CPU part of the chip is estimated to be approximately 8mmx8mm in an nMOS process with Lambda = 3 microns.

TRANSPUTER

INMOS' TRANSPUTERS [1] comprise a family of 16- and 32-bit microcomputers, capable of operating alone as a 10-MIPS (million instructions per second) processor or as a component of a parallel network of TRANSPUTERS.

Each microcomputer, as shown below, consists of four main parts: a reduced instruction set processor, 4K bytes of static RAM, a 32-bit multiplexed memory interface, and four INMOS standard serial links providing concurrent message passing to other TRANSPUTERS. The processor has built in support for multi-processing and parallelism. The execution state of each process is defined by six registers. These registers are a three-register evaluation stack, together with an instruction pointer, a workspace pointer, and an operand register. Instructions are eight bits, comprising a 4-bit function code and a 4-bit data value. Operands longer than four bits are built up four bits at a time in the operand register. Basic arithmetic instructions execute in 50 nsecs and a process switch takes only 600 nsecs.

```
 ------------ 32 bits  ------------
|            |     ||     | PROCESSOR  |
|  MEMORY    |     ||<==>|  32 bits   |
|            |     ||     | (10 MIPS)  |
| 4k bytes   |     ||      ------------
|            |<==>||       ------------
|  50 nsec   |     ||<==>|    LINK     |
|            |     ||      ------------
|            |     ||      ------------
|            |     ||<==>|    LINK     |
|            |     ||      ------------
|            |     ||      ------------
 ------------      ||<==>|    LINK     |
 ------------      ||      ------------
| MEMORY     |<==>||       ------------
| INTERFACE  |     ||<==>|    LINK     |
 ------------      ||      ------------
      ||
```

Figure 9: INMOS TRANSPUTER

Communication between TRANSPUTERS is handled by the links. Each link implements two channels, an output and an input, over which messages are transmitted as a series of bytes.

Finally, parallel programming in TRANSPUTERS and its OCCAM programming language [18] is based on communicating processes and message passing using explicitly defined channels. A network of TRANSPUTERS corresponds directly to a network of processes, with each TRANSPUTER supporting one or more processes in a timeshared fashion.

5. FUTURE DIRECTIONS

VLSI microprocessors containing over 100,000 devices are starting to become commonplace [26]. The advantages of placing a sophisticated computer on a chip are well understood to include improvements in its component cost, performance and reliability, together with its power and size. In addition, as the level of integration increases we can expand the chip boundary to encompass cache and main storage in order to improve the performance of the processor. This is advantageous given the relatively slower improvement in the bandwidth available at the pins compared with that in the processor.

A limitation with this "ever larger single processor" approach is that the communications required to execute each instruction must ordinarily traverse all parts of the processor. In making the microprocessor more powerful, additional logic and storage is added, but this increases the communication costs by increasing the separation of the components. It is clear that such microprocessor designs will reach a point of diminishing cost/performance returns.

In the future improving performance with an ensemble of concurrently operating small processors becomes more attractive than using a large single processor. Thus the INMOS TRANSPUTER and the CMU Programmable Systolic Chip seem to be showing the future direction of VLSI architectures.

REFERENCES

[1] Barron, I., et al, "Transputer does 5 or more MIPS even when not used in parallel," Electronics, vol. 56, no. 23, November 1983, pp. 109-115.

[2] Batali, J., et al, "The SCHEME-81 architecture - system and chip," Proc. Conf. Advanced Research in VLSI , Paul Penfield (ed.), MIT Press, January 1982, pp. 69-77.

[3] Clark, J.H., "A VLSI Geometry Processor for Graphics," COMPUTER, vol. 13, no. 7, July 1980, pp. 59-68.

[4] Conway, L., et al, "MPC79: A Large-Scale Demonstration of a New Way to Create Systems in Silicon," Lambda, 2nd Qtr. 1980, pp. 10-19.

[5] Fisher, A.L., et al, "Architecture of the PSC: A Programmable Systolic Chip," Proc. Tenth Int. Symp. on Computer Architecture, June 1983, pp. 48-53.

[6] Foti, L., et al, "Reduced Instruction Set Multi-Microcomputer System (RIMMS)," Proc. National Computer Conf., July 1984.

[7] Gray, J.P. (ed.), Proc. VLSI 81, Academic Press, New York 1981.

[8] Gupta, A., and Toong, H.D., "An Architecture Comparison of 32-bit Microprocessors," IEEE MICRO, vol. 3, no. 1, February 1983, pp. 9-22.

[9] Guterl, F., "Microprocessors Chip Architecture: a revolution brewing," IEEE Spectrum, July 1983, vol. 20. no. 7, pp. 30-37.

[10] Hennessy, J.L., "VLSI Processor Architecture," IEEE Trans. on Computers, December 1984, vol. C-33, no. 12, pp. 1221-1246.

[11] Holloway, J., et al, "SCHEME-79 - LISP on Chip," COMPUTER, vol. 14, no. 7, July 1981, pp. 10-21.

[12] Jansen, W.D., and Fairbairn, D.G., "The Silicon Foundry: Concepts and Reality," Lambda, 1st Qtr. 1981, pp. 16-26.

[13] Kung, H.T., "Why Systolic Arrays," COMPUTER, vol. 15, no. 1, January 1982, pp. 37-46.

[14] Kung, R.F., et al (ed.), Proc. CMU Conf. VLSI Systems and Computations, Springer-Verlag, New York, October 1981.

[15] Lepselter, M.P., "High Speed Silicon Integrated Circuits Using X-ray Lithography," Conf. Proc. on Advanced Research in VLSI, MIT, January 25-27, 1982.

[16] Lyon, R.F., "A Bit-Serial VLSI Architecture Methodology for Signal Processin," Proc. VLSI 81, August 1981, pp. 131-140.

[17] Lyon, R.F., "The Optical Mouse, and an Architecture Methodology for Smart Digital Sensors," Technical Report VLSI-81-1, Xerox Corporation Palo Alto Research Center, August 1981.

[18] May, D., "OCCAM," ACM SIGPLAN Notices, vol. 18, no. 4, April 1983, pp. 69-79.

[19] Mead, C.A., and Conway, L., "Introduction to VLSI Systems," Addison-Wesley, Reading, Mass., 1980.

[20] Patterson, D.A., "Reduced Instruction Set Computers," Comm. ACM, vol. 28, no. 1, January 1985, pp. 8-21.

[21] Patterson, D., and Sequin, C., "A VLSI RISC," COMPUTER, vol. 15, no. 9, September 1982, pp. 8-21.

[22] Penfield, P. (ed.), 1982 Conf. on Advanced Research in VLSI, MIT January 1982.

[23] Rivest, R.L., "A Description of a Single-Chip Implementation of RSA Cipher," Lambda, 4th Qtr. 1980, pp. 14-18.

[24] Seitz, C., "Ensemble Architectures for VLSI - A Survey and Taxonomy," Proc. 1982 Conf. on Advanced Research in VLSI, P. Penfield (ed.), MIT, January 1982, pp. 33-45.

[25] Seitz, C.L., "Concurrent VLSI Architectures," IEEE Trans. on Computers, December 1984, vol. C-33, no. 12, pp. 1247-1265.

[26] Treleaven, P.C., "VLSI Processor Architectures," COMPUTER, vol. 15, no. 6, June 1982, pp. 33-45.