UNIVERSITY OF JOENSUU
DEPARTMENT OF COMPUTER SCIENCE
Report Series A

Are Multiport Memories Physically Feasible?

Martti Forsell

Report A-1993-1

ACM B.3.2
UDK 681.3.07
ISSN 0789-7316
ISBN 951-708-187-1

Are multiport memories physically feasible?

Martti J. Forsell

Department of Computer Science, University of Joensuu, PB 111, SF-80101 Joensuu, Finland

October 27, 1993

Abstract

A Parallel Random Access Machine (PRAM) is a popular model of parallel computation that promises easy programmability and great parallel performance, but only if efficient shared main memories can be built. This would not be easy, because the complexity of shared memories leads to difficult technical problems. In this paper we consider the idea of true multiport memory that can be used as a building block for efficient PRAM-style shared main memory. Two possible structures for multiport memory chips are presented. We shall also give a preliminary cost-effectivity and performance analysis for memory systems using proposed multiport RAMs. Results are encouraging: At least small size multiport memories seem to be physically feasible. Also the power of PRAM model can be fully exploited by computer systems with multiport memories.

Keywords: PRAM, parallelism, shared memories, VLSI

---

1. Introduction

Efficient parallel computers are hard to manufacture because of difficult technical problems. The most important is: how to arrange efficient communication between processors?

Theoretically this problem was best solved by using a shared main memory between processors [Schwarz66, Karp69, Fortune78]. In the 50's, 60's and 70's however, the memory manufacturing technology was so underdeveloped that the idea of building true shared memories was considered impossible. Instead several other possibilities, like memory interleaving and memory distribution were investigated [Burnett70, Enslow74, Schwarz80].

The use of true shared main memory continued among algorithm writers, because of the simplicity of programming. Even several abstract parallel programming models using shared memory were designed. The most widely used model is a Parallel Random Access Machine [Fortune78, Leighton91, McColl92] (see Section 2 for a brief explanation).

Email: mforsell@cs.joensuu.fi

During the 80's one began to use message passing processor networks to simulate the PRAM model [Schwatz80, Gottlieb83, Gajski83, Hillis85, Pfister85, Ranade87, Abolhas91], because the manufacturing of shared memories was still considered impossible. Despite of those efforts, no serious commercial products are available yet.

In this paper we return to the original idea of a shared memory of PRAM model and consider it in the light of current VLSI technology. We claim that the progress in the VLSI area during the last two decades has made it possible to consider manufacturing the main memory of a PRAM-style computer of shared semiconductor memories.

The number of ports and size of multiport memories will be quite limited, but it will be shown that even with small port count computers with multiport memories will be faster than other systems with remarkably greater number of processors.

Two possible structures of multiport memories will be presented. We will also estimate the complexity, cost-effectiveness and performance of proposed multiport memories.

2. Parallel random access machine

The *Parallel Random Access Machine* (PRAM) model is a logical extension of the Random Access Machine (RAM) model. It is a widely used abstract model of parallel computation [Fortune78, Leighton91, McColl92].

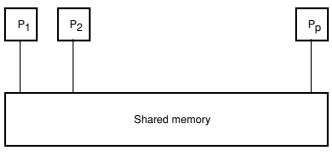

Figure 1. The PRAM model. $P_1,...,P_P$ are processors.

A PRAM consists of P processors each having a similar register architecture. All processors are connected to a shared memory (see Figure 1). All processors run the same program synchronously, but a processor may branch within the program independently of other processors. That is, each processor has its own Program Counter.

The model has several different variants according to level of shared memory access. We refer to two of them in this paper (most of the other models can be treated analogously):

CRCW Common

- Processors can read and write a memory cell simultaneously. In the case of writing all processors storing to the same cell must have identical data.

CRCW Priority

- Processors can read and write a memory cell simultaneously. In the case of writing to the same cell the processor with the lowest index succeeds.

The Recent PRAM algorithm research is valuable because it promises that parallel processing power of multiprocessors can be exploited in very large range of applications if PRAM-style computers can be built.

3. Proposed implementations of shared memories

Except multiport memories there are at least three different methods to implement a shared memory: One can use an ordinary memory, memory interleaving or shared memory simulation by distributed memories.

3.1 Ordinary memory

Ordinary memories or single port memories are used to implement shared memories in small multiprocessor systems. An ordinary memory module can be connected to several processors through a time-shared bus [Enslow74] (see Figure 2).

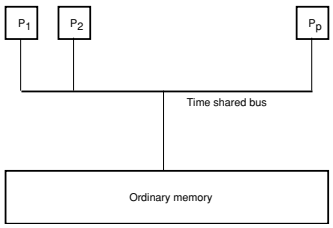

Figure 2. A multiprocessor system using an ordinary memory.

By using this method, we however face serious bandwidth problems: If, for example, *P* processors are simultaneously accessing the memory, this will take *P* times longer than in the case of a single processor. That is because only one memory access can be done through a single port at the same time. This problem can be partially solved by using snooping caches [Tomasevic93], but they will not help us in the case of multiple simultaneous writes.

In order to use an ordinary memory in a multiprocessor system the memory must be superfast, otherwise the performance of the system will not be much greater than the performance of a single processor system. Because the current access times of ordinary memory chips are apparently equal to processor clock cycle times, the existence of a superfast memory is considered unrealistic. Therefore, there is almost no possibility for implementing any serious parallel computer system with ordinary memories.

3.2 Memory interleaving

Memory interleaving is a memory system implementation method that is used for example in current supercomputers. The memory of an interleaved system is divided into several independent banks that are connected to processors through a crossbar switch [Burnett70, Enslow74] (see Figure 3).

Information is stored in the memory with sequential items residing in banks that are consecutive, modulo the number of memory banks. A *memory bank* is a memory block that is built as an ordinary memory. Thus, it is not possible that several processors simultaneously access a memory within the same bank. However, several processors can access several banks simultaneously.

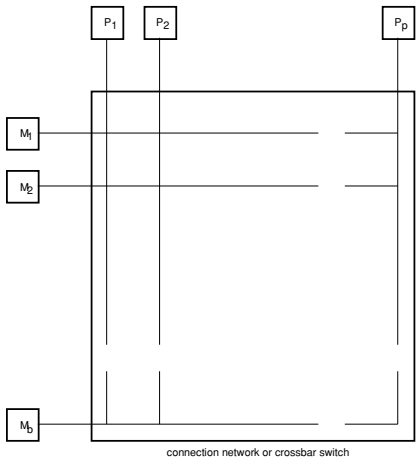

Figure 3. A multiprocessor system using an interleaved memory. $P_1,...,P_P$ are processors and $M_1,...,M_b$ are memory banks.

The advantage of the memory interleaving is obvious with vectorizable algorithms, because subsequent vector elements can be placed in subsequent banks and accessed simultaneously. Thus parts of vectors can be fetched in one memory cycle.

However interleaved memory systems fail to provide the ease of programming and the performance of ideal shared memories. Constantly it happens that several processors try to access simultaneously the same bank and some waiting is needed. This is very usual especially in efficient parallel PRAM algorithms, like parallel sorting and prefix calculation.

3.3 Simulation of a shared memory by a distributed memory

So far, the most promising method for implementing a shared memory has been a *simulation of a shared memory by a distributed memory*. In this solution an

ordinary memory is distributed over a processor network (see Figure 4).

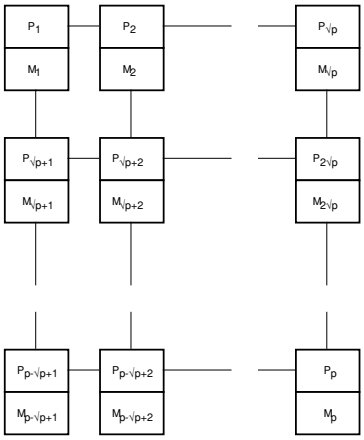

Figure 4. A multiprocessor system using a shared memory simulation by a distributed memory. Processors $P_1,...,P_p$ with their local memories $M_1,...,M_p$ are organized here as a 2-dimensional mesh.

Each network node consists of both a processor and an ordinary local memory. Processors use message passing to access memory locations that do not lie in their local memory.

According to investigations such a processor networks can be used to simulate a PRAM or some other parallel processing model [Schwatz80, Gottlieb83, Hillis85, Pfister85, Alt87, Bilardi88, Ranade87, Valiant90, Feldman92, Prechelt93]. The suggested network topologies range from hypercubes to butterflies and meshes.

It has turned out that the proposed shared memory machines have poor performance in relation to the ideal PRAM model [Ranade87, Blelloch89, Prechelt93]. That is because the simulation of a shared memory takes so much time.

Table 1 shows estimated average routing times of message permutations in three architectures. As can be seen, hundreds or even thousands of steps may be needed for simulating one permutation. As a result one PRAM step would execute in several hundreds of steps. Thus, we may need hundreds of processors even to get the performance of single processor.

Machine	Steps	#Proc
Fluent Machine [Ranade87]	3380	112k
PRAM on CM2 [Blelloch89]	600	64k
MasPar MP-1216A [Prechelt93]	4800	16k

Table 1. Latencies for permutation routing in memory cycles on certain parallel machines.

4. A natural solution for shared memories: multiport semiconductor RAM

The fourth and the most obvious way to implement shared memories is to use multiport RAMs as building blocks for a true shared memory.

A *Multiport Random Access Memory* (multiport RAM) is a memory having multiple ports that are to be used to access memory cells simultaneously and independently of each other.

In parallel computers one processor is usually connected to one port. From the processor point of view there exists a uniform shared memory connected to it. Other processors do not affect the operation of a processor; only the contents of a memory may be changed by instructions executed by other processors.

Note that the term multiport memory is used here in different meaning than in the 70's, when multiport memories were actually interleaved memories in which connection networks were closely integrated into the memory systems [Enslow74].

Small and fast multiport *Static Random Access Memories* (SRAMs) are currently commercially available, for example, from IDT. Sizes of memories vary from 8 to 128 kbit and number of ports vary from 2 to 4.

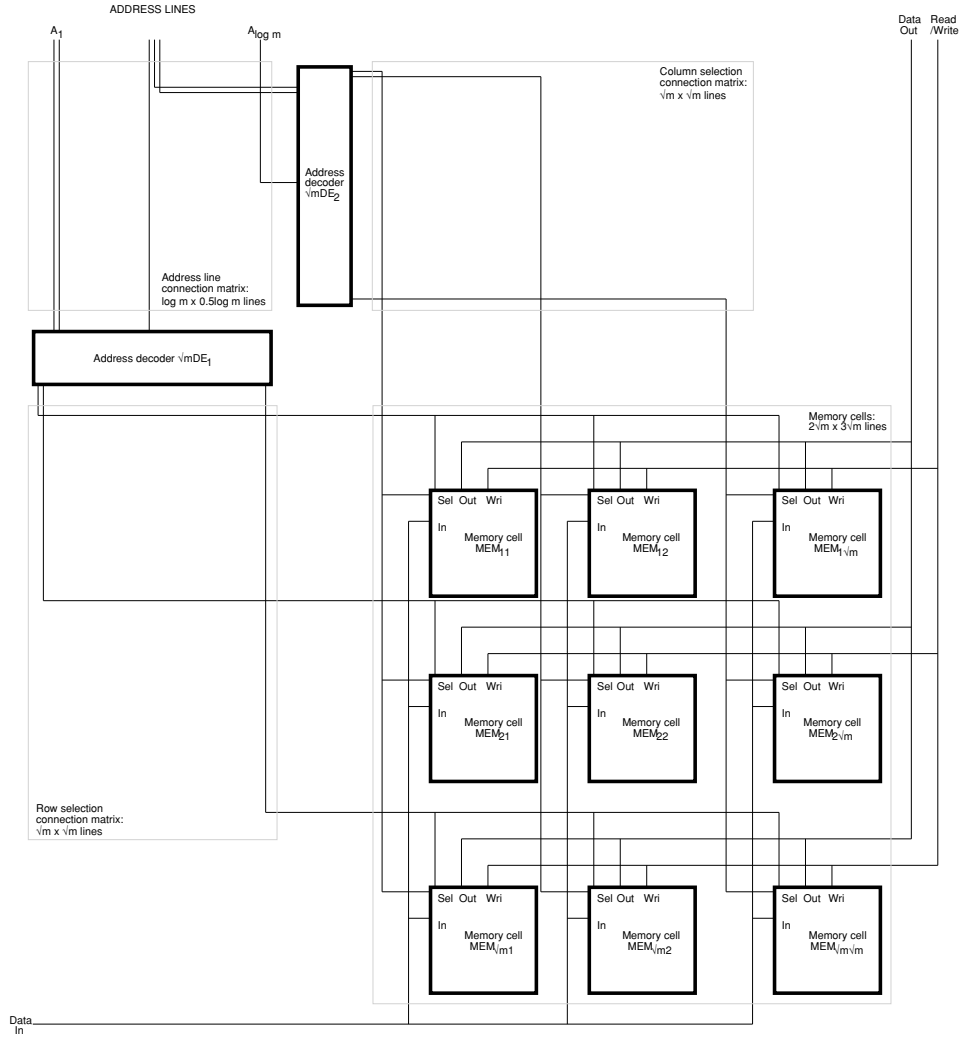

Figure 5. The logical structure of an ordinary single port m cell static RAM chip. Chip select and power lines are not shown. The physical lay-out of components does not reflect the actual placement of semiconductor elements and is meant only for comparing single port and multi port versions of memory chips. For the same reason the connection matrices are rounded with dotted lines and labelled with complexity numbers.

It is possible to use these memories to build *Concurrent Read Exclusive Write* (CREW) PRAM-style shared memories, but the limited size and port count badly affect the performance of such memory systems in real world applications. Also the structure of these memories does not allow simultaneous writes that is needed in CRCW PRAM-style memories.

To estimate the theoretical feasibility of multiport main memories in general, two more powerful structures of multiport RAMs are presented and analyzed.

4.1 The structure of a multiport RAM

Let us first take a look at the logical structure of an ordinary RAM. A typical SRAM chip consists of two address decoders, memory cell array, bus interface and some wiring (see Figure 5).

The memory chip is connected to other devices through address lines, data lines, read/write line, chip select line and power lines. For the well known complexity reasons memory cells are arranged into the form of a square array instead of line array in current memory chips. The address must therefore be divided into two parts for decoding.

To select an appropriate memory cell, appropriate row and column select lines must be activated. Address decoders are used to select the right memory cell by activating the right selection lines according to the address information provided in address lines. The read operation (write operation) is selected by activating (deactivating) read/write line.

A memory cell consists of few gates that are connected to each others (see Figure 6). The cell is connected to other elements in the chip through row and column select lines, read/write line, input line and output line.

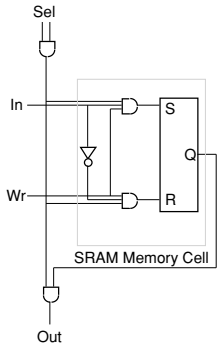

Figure 6. An example of a typical SRAM cell.

To transform a single port memory to a multiport memory ports are added by adding new address lines, data lines, chip select lines and decoders for each port. Inside the chip the selection circuit of memory cells is duplicated for each port and the memory cells are made multiported (see Figure 7).

A memory chip now consists of two address decoders per port, a common memory cell array and multiple wiring circuitries (see Figure 8). The multiport chip is connected to outside world through multiple address lines, multiple data lines, multiple read/write lines, multiple chip select lines and power lines.

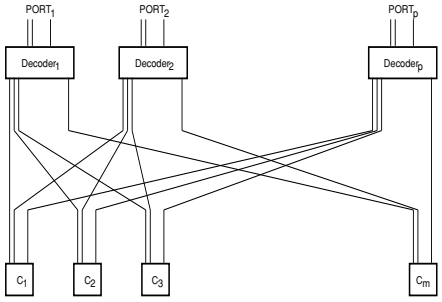

Figure 7. An ordinary memory is made parallel by adding new lines for each port and duplicating the cell selection circuitry for each port. $C_1,...,C_m$ are multiported cells.

The memory operations are done in the similar manner to that in the ordinary memory except now multiple ports control multiple operations in one or more cells. Address information provided in a memory port is decoded in appropriate address decoders to select appropriate memory cell by activating right row and column select lines of the cell.

The structure of a multiport memory cell is naturally more complex than the structure of an ordinary cell. We present here the logical structure of a CRCW Common cell and a CRCW Priority cell.

In CRCW Common-style memories several processors can simultaneously write the same data to a memory cell. The correct operation is obtained by ORing the input values together and then feeding the result to the original cell (see Figure 9). A read/write operation is determined by ORing read/write lines and cell select is determined by ORing the select lines. Finally, the output circuitry is duplicated for each port.

In CRCW Priority-style memories several processors can simultaneously write

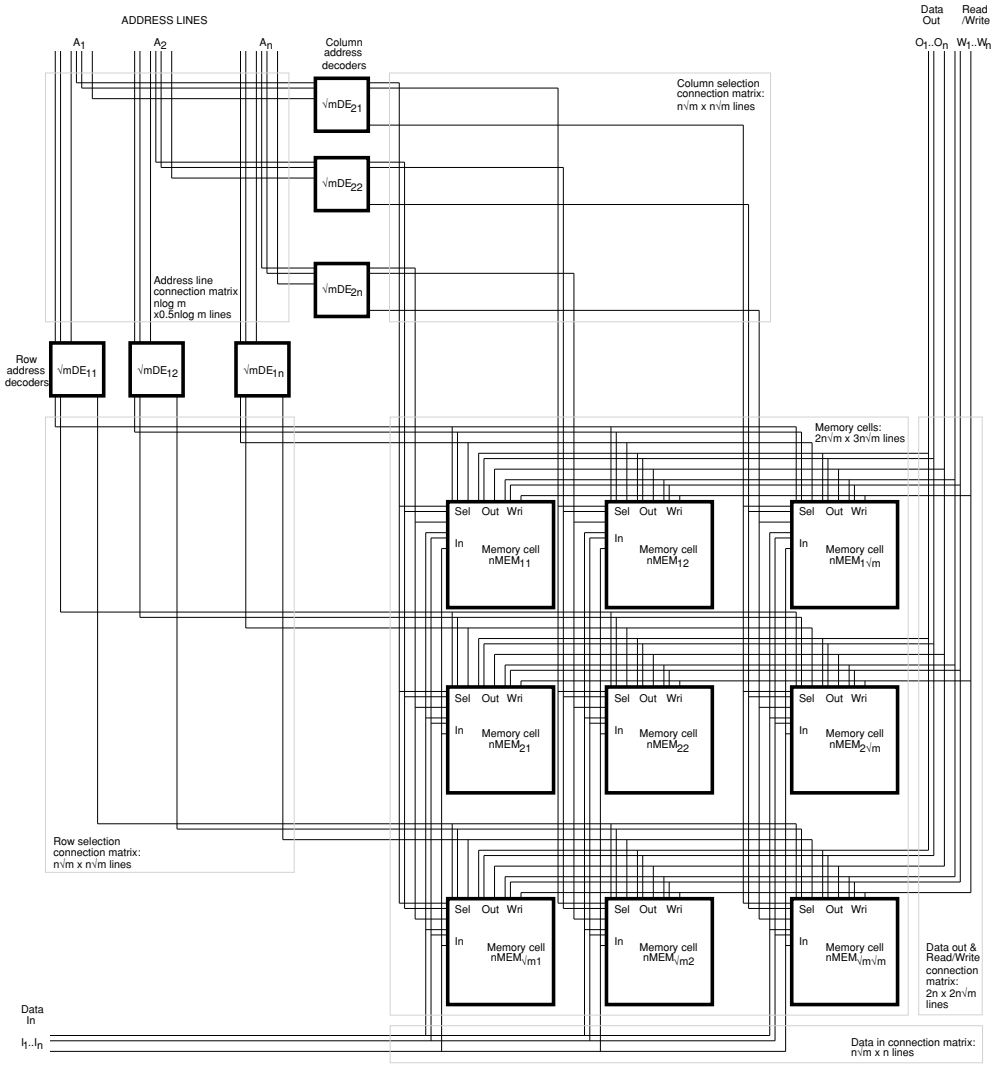

Figure 8. The logical structure of proposed n port m cell static RAM chip. Chip select and power lines are not shown. The physical lay-out of components does not reflect the actual placement of semiconductor elements and is meant only for comparing singleport and multiport versions of memory chips. For the same reason the connection matrices are rounded with dotted lines and labelled with complexity numbers.

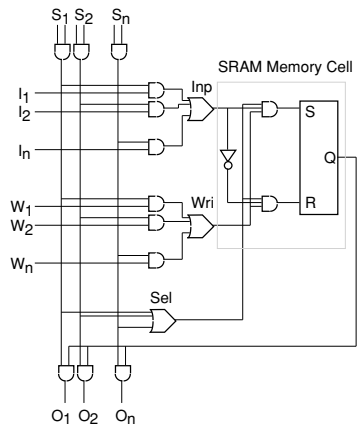

Figure 9. An example of a CRCW Common SRAM cell.

different data to the same memory cell. The processor with the lowest index succeeds. The correct operation is obtained by select-

ing the first inactive read/write line among all inactive read/write lines (see Figure 10). Write data is selected by a parallel priority encoder which activates all the read/write lines except the first inactive line.

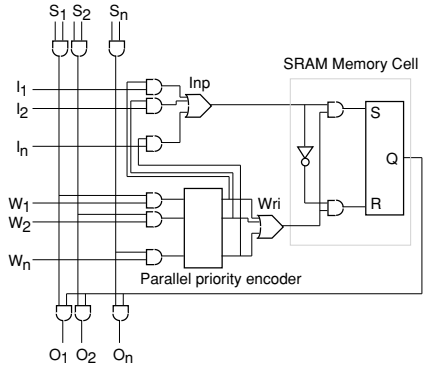

Figure 10. An example of a CRCW Priority SRAM cell.

A read/write operation is determined by ORing these lines together. The right input is selected among all the inputs by ANDing corresponding input line and output of parallel priority encoder. Obtained input lines can then be ORed together to form the input of the original cell. Finally, the output circuitry is duplicated for each port.

The word length of a memory chip can be increased by duplicating data-in lines, data-out lines and memory cells for each bit of memory word. No changes are needed for cell selection circuitry, because a memory word is selected simultaneously. This simplifies the memory cell array relatively.

4.2 The cost-effectiveness of a multiport RAM

The relative cost-effectiveness of the multiport RAM is estimated by calculating wiring and component count complexity factors for ordinary and multiport memories. It is shown that multiport semiconductor RAMs look adequate in the sense of cost-effectiveness.

Suppose we are comparing an m location w-bit word SRAM and a P port m location w-bit SRAM. Knowing that a log n-line to n-line parallel decoder has $n\log n + \log n$ components and a $2\log n \times 2n$ line matrix we get: Ignoring memory cells a P-port SRAM chip has P times more components ($0.5P(1 + \sqrt{m})\log m$ compared to $0.5(1 + \sqrt{m})\log m$) and P times more lines than a SRAM chip of the same size ($P\log m \times 2\sqrt{m}$ compared to $\log m \times 2\sqrt{m}$).

Both P-port and single port chips have the same count of memory cells, but in the multiport case the cells are more complex: Table 2 shows the complexity of memory cells in units of cell level gate inputs (reflecting the number of transistors in cells). The complexity of parallel N line priority encoder found in a CRCW Priority cell is assumed to be $N+2N \log N$ inputs rather than N^2+2N-1 inputs of unoptimized parallel priority encoder.

Type of memory	Cell level input count
Ordinary SRAM	$13w+2$
CRCW Common	$5Pw+6P+11w$
CRCW Priority	$5Pw+5P+2P\log P +9w$

Table 2. The complexity of RAMs (in units of cell level gate inputs)

Table 3 shows the complexity of memory cell array wiring in units of lines.

Type of memory	Cell array lines
Ordinary SRAM	$(w+1)\sqrt{m} \times (w+2)\sqrt{m}$
CRCW Common	$P(w+1)\sqrt{m} \times P(w+2)\sqrt{m}$
CRCW Priority	$P(w+1)\sqrt{m} \times P(w+2)\sqrt{m}$

Table 3. The complexity of RAMs (in units of connection matrix lines)

Simple calculations show now that P-port CRCW Common-style RAM requires P times more components and slightly less than P^2 times more silicon area in relation to single port RAM of same size. Assuming $P \leq 1024$, P-port CRCW Priority-style RAM requires $2P$ times more components and slightly less than P^2 times more silicon area. From the complexity numbers we conclude that CRCW Common-style and CRCW

Priority-style memories are slightly less than P^2 times less cost effective than ordinary memories of same size.

It is mainly the increase of wiring rather than the increase in the number of components that explodes the complexity of multiport RAMs when adding more ports. This makes the use of complex and powerful CRCW PRAM-style memories desirable over simpler and less powerful CREW PRAM-style memories currently available.

The size and port count of multiport memory chips in real multiprocessors will be quite limited, maybe within range of 4 kbit to 4 Mbit and 4 to 32 ports, because of the complexity of wiring. These estimations are based on the state of current VLSI technology.

4.3 The performance of the multiport RAM system

The performance calculations are favorable for PRAMs using multiport RAMs even with small count of processors. We will estimate the relative memory cycle time of a multiport RAM and the performance of multiprocessor using multiport RAMs.

If we do not count memory cells, the chip level delays are at least 3 gate delays for both ordinary RAMs and the proposed multiport RAMs. (One unit for chip selection and two units for address decoding.) The cell level write operation delays are shown in Table 4. The cell level read operation delay is one for both single port and multiport versions.

Type of memory	Cell level	Chip level
Ordinary SRAM	4	7
CRCW Common	6	9
CRCW Priority	7+log P	10+log P

Table 4. The length of cell level write operation datapaths and total chip level datapaths of RAM's (in units of gate delays)

The access times of multiport RAMs can likely be made small enough to fit in one clock cycle of a fast processor, because the total datapath of multiport RAM is only slightly longer than in ordinary RAM. If the port count is high, however, the CRCW Priority chips may be a little slower.

Relatively short cycle times of proposed memories have a dramatic effect on the memory system performance: A 16 processor PRAM-style computer using proposed chips (referred as 16-PRAM) would be as fast as a 10 000 processor network assuming the simulation of one PRAM step requires approximately 600 memory cycles on network (see Table 1).

In the case of sequential algorithms the difference is even greater: A 16-PRAM would run sequential algorithms even 600 times faster than 10 000 processor network. That is because 16-PRAM can run sequential algorithms as a fast as sequential machine, but global memory access takes so much time in machines with distributed memory.

On the other hand, there exist a lot of algorithms that do not require extensive communication and can therefore be executed efficiently on distributed machines. In these cases a PRAM style computer with a shared memory and small number of processors provides performance comparable to distributed machine with the same number of processors.

4.4 An example of a multiport RAM system within the limits of current technology

To give a rough idea of the complexity of multiport memories and the possibilities of the current technology, suppose for example, that we want to manufacture a 16 port 16kb x 1 CRCW Common-style static RAMs with access time of 20 ns. That should not be unrealistic because such chips are equally complex as currently available single port 4096kb x 1 static RAMs with access time of

15 ns. In fact, 7 to 128 decoders of a 16-port RAM are 50 times less complex than the 11 to 2048 decoders of a single port RAM.

The pin count of 16 port chips would be 290 instead of 28 in single port case. (The pin count of general w-bit word P port m location memory is $P(w + 2 + \log m) + 2$ pins assuming w bi-directional data lines.)

To build a true 16 port 1 MB shared main memory one needs 512 such 16 port 16kb x 1 CRCW Common-style chips. To reduce the complexity of the motherboard of parallel computers maybe SIMM-style packages could be used. The SIMM count of 1 MB memory would be 64 assuming one SIMM can hold 8 chips. The 1 MB 16 port memory bus would be at least 848 lines wide assuming byte addressing and 32-bit data bus. (The general P port M location W-bit word memory system bus is $P(W + 1 + \log M)$ lines wide assuming word addressing.)

The feasibility of multiport RAMs allows the arrival of high performance PRAM-style computers. Because of the limited port count and only medium memory size, multiport RAMs are likely to lead to a nonscalable coarse grain parallel architecture, in which processing elements are fast and the processor count is small.

The development of vector processors, superscalar processors and VLIW processors also gets significant boost from the arrival multiport memories. More instruction level parallelism can be exploited and complex snooping cache memory solutions can be replaced with true multiport memories or maybe with multiport caches.

5. Summary

We have proposed two possibilities to implement true PRAM-style shared main memories. The logical structure of building blocks, multiport RAMs is shown to be a simple extension of ordinary RAMs.

We have compared the relative cost-effectiveness of ordinary memories and multiport memories. The results are only adequate for multiport memories. P-port CRCW Common-style memories and CRCW Priority-style memories are slightly less than P^2 times less cost-effective than ordinary memories.

It was already known that multiport memories were expensive and difficult to built. However with current VLSI technology manufacturing is not impossible. We believe that semiconductor producers will be able to manufacture such multiport RAMs within the next few years.

We have also compared the performance of some known message passing systems and the proposed multiport memory system. The results are favorable for multiport memories. The power of PRAM model can be fully exploited with computer systems with multiport RAMs.

Computers with multiport memories will be even hundreds of times faster than computers with message passing and distributed memory. The complexity of multiport memories will prevent the manufacturing of massively parallel PRAMs with multiport memory technology, but even with small port count computers with multiport memories will provide performance comparable to distributed memory systems with remarkably greater processor count.

The challenge for the 90's is: will the multiport memory technology provide a real alternative to the message passing technology? Current research in the areas of wafer scale integration, multichip module production and optical communication may turn multiport memories even more attractive alternatives for distributed memories.

Our results concerning multiport memories are quite theoretical and there is lot of practical work to be done before we can take advantage of actual products. Anyway the

challenge to build multiport memories is given. It is up to chip manufacturers to do appropriate development decisions.

References

[Abolhas91]
F. Abolhassan, J. Keller, W. J. Paul, On the Cost-effectiveness and Realization of the Theoretical PRAM Model, FB 14 Informatik, *SFB-Report 09/1991*, Universität des Saarlandes, Saarbrucken, 1991.

[Alt87]
H. Alt, T. Hagerup, K. Mehlhorn, F. P. Preparata, Deterministic simulation of idealized parallel computers on more realistic ones, *SIAM J. Comput.* **16**, 5 (1987) 808-835.

[Bilardi88]
G. Bilardi and K. T. Herley, Deterministic simulations of PRAMs on bounded degree networks, *Proceedings of 26th Annual Allerton Conference on Communication, Control and Computation* (1988).

[Blelloch89]
Guy E. Blelloch, Scans as Primitive Parallel Operations, *IEEE Transactions on Computers* **38**, 11 (1989) 1526-1538.

[Burnett70]
G. J. Burnett and E. G. Coffman, A Study of Interleaved Memory Systems, *AFIPS Conference Proceedings SJCC*, 36 (1970), 467-474.

[Enslow74]
P. H. Enslow, *Multiprocessors and parallel processing*, John Wiley&Sons, New York, 1974.

[Feldman92]
Y. Feldman, E. Shapiro, Spatial Machines: A More Realistic Approach to Parallel Computation, *Communications of the ACM* **35**, 10 (1992) 61-73.

[Fortune78]
S. Fortune and J. Wyllie, Parallelism in Random Access Machines, *Proceedings of 10th ACM STOC* (1978), 114-118.

[Gajski83]
D. Gajski, D. Kuck, D. Lawrie and A. Sameh, CEDAR-A Large Scale Multiprocessor, *Proceedings of International Conference on Parallel Processing* (1983), 524-529.

[Gottlieb83]
A. Gottlieb, R. Grishman, C. P. Kruskal, K. P. McAuliffe, L. Rudolph and M. Snir, The NYU Ultracomputer - Designing a MIMD, shared-memory parallel machine, *IEEE Transactions on Computers* **C-32**, (1983) 175-189.

[Hillis85]
W. D. Hillis, The Connection Machine, *The MIT Press*, 1985.

[Karp69]
R. M. Karp and R. E. Miller, Parallel Program Schemata, *Journal of Computer and System Sciences* **3**, 2 (1969) 147-195.

[Leighton91]
T. F. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*, Morgan Kaufmann, San Mateo, 1992.

[McColl92]
W. F. McColl, General Purpose Parallel Computing, *Lectures on Parallel Computation Proceedings 1991 ALCOM Spring School on Parallel Computation (Editors: A. M. Gibbons and P. Spirakis)*, Cambridge University Press, Cambridge, 1992, pp. 333-387.

[Pfister85]
G. F. Pfister, W. C. Brantley, D. A. George, S. L. Harvey, W. J. Kleinfelder, K. P. McAuliffe, E.A. Melton, V. A. Norton and J. Weiss, The IBM Research Parallel Processor Prototype (RP3): Introduction and Architecture, *Proceedings of International Conference on Parallel Processing* (1985), 764-771.

[Prechelt93]
L. Prechelt, Measurements of MasPar MP-1216A Communication Operations, *Technical Report 01/93*, Universität Karlsruhe, Karlsruhe, 1993.

[Ranade87]
A. G. Ranade, S. N. Bhatt, S. L. Johnson, The Fluent Abstract Machine, *Technical Report Series BA87-3*, Thinking Machines Corporation, 1987.

[Schwarz66]
J. T. Schwarz, Large Parallel Computers, *Journal of the ACM* **13**, 1 (1966) 25-32.

[Schwarz80]
J. T. Schwarz, Ultracomputers, *ACM Transactions on Programming Languages and Systems* **2**, 4 (1980) 484-521.

[Tomasevic93]
M. Tomasevic and V. Milutinovic, A Survey of Maintenance of Cache Coherence in Shared Memory Multiprocessors, *Proceedings of the Hawaii International Conference on System Sciences*, Koloa, January 5-8, 1993.

[Valiant90]
L. G. Valiant, A Bridging Model for Parallel Computation, *Communications of the ACM* **33**, 8 (1990) 103-111.