SEMINAR PAPER

ON

SMART MEMORIES

BY

OGUNTOLU IRETIOLA DEBORAH

MATRIC NO: NCSF/15/0055

SUBMITTED TO SCHOOL OF PURE AND APPLIED SCIENCE

OGUN STATE INSTITUTE OF TECHNOLOGY, IGBESA.

IN PARTIAL FULFILMENT FOR THE AWARD OF

NATIONAL DIPLOMA IN THE DEPARTMENT OF COMPUTER SCIENCE.

MAY, 2017

CERTIFICATION

This is to certify that this research work has been carried out by **OGUNTOLU IRETIOLA DEBORAH** with the matric number NCSF/15/0055 in computer science department, **OGUN STATE INSTITUTE OF TECHNOLOGY, IGBESA OGUN STATE.**

…… ……

Oguntolu Iretiola. D Date

Student

…… ……

Mr Akinola Date

Supervisor Signature

…… …….

Mrs. Oladejo. R Date

Head of Department Signature

DEDICATION

This project work is dedicated to Almighty God. Also to my parent Mr & Mrs OGUNTOLU IRETIOLA DEBORAH for giving me a formal education at all cost.

ACKNOWLEDGEMENT

My utmost gratitude, thanksgiving and appreciation go to the almighty God; I also give thanks to my parent Mr. & Mrs. OGUNTOLU for their moral and financial support for this study to come to completion. I would like to thank my supervisor Mr AKINOLA and the HOD computer science department Mrs OLADEJO RACHEAL for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present a seminar report. It helped me a lot to realize of what we study for.

TABLE OF CONTENTS

ABSTRACT

Trends in VLSI technology scaling demand that future computing devices be narrowly focused to achieve high performance and high efficiency, yet also target the high volumes and low costs of widely applicable general purpose designs. To address these conflicting requirements, we propose a modular reconfigurable architecture called Smart Memories, targeted at computing needs in the 0.1mm technology generation. A Smart Memories chip is made up of many processing tiles, each containing local memory, local interconnect, and a processor core. For efficient computation under a wide class of possible applications, the memories, the wires, and the computational model can all be altered to match the applications. To show the applicability of this design, two very different machines at opposite ends of the architectural spectrum, the Imagine stream processor and the Hydra speculative multiprocessor, are mapped onto the Smart Memories computing substrate. Simulations of the mappings show that the Smart Memories architecture can successfully map these architectures with only modest performance degradation.

A Smart Memories chip is made up of many processing tiles, each containing local memory, local interconnect, and a processor core. For efficient computation under a wide class of possible applications, the memories, the wires, and the computational model can all be altered to match the applications. To show the applicability of this design, two very different machines at opposite ends of the architectural spectrum, the Imagine stream processor and the Hydra speculative multiprocessor, are mapped onto the Smart Memories computing substrate. Simulations of the mappings show that the Smart Memories architecture can successfully map these architectures with only modest performance degradation.

CHAPTER ONE

1.0 INTRODUCTION

The continued scaling of integrated circuit fabrication technology will dramatically affect the architecture of future computing systems. Scaling will make computation cheaper, smaller, and lower power, thus enabling more sophisticated computation in a growing number of embedded applications. This spread of low-cost, low power computing can easily be seen in today's wired (e.g. gigabit Ethernet or DSL) and wireless communication devices, gaming consoles, and handheld PDAs. These new applications have different characteristics from today's standard workloads, often containing highly data- parallel streaming behaviour. While the applications will demand ever-growing compute performance, power (ops/W) and computational efficiency are also paramount; therefore, designers have created narrowly focused custom silicon solutions to meet these needs.

However, the scaling of process technologies makes the construction of custom solutions increasingly difficult due to the increasing complexity of the desired devices. While designer productivity has improved over time, and technologies like system-on-a-chip help to manage complexity, each generation of complex machines is more expensive to design than the previous one. High non-recurring fabrication costs (e.g. mask generation) and long chip manufacturing delays mean that designs must be all the more carefully validated, further increasing the design costs. Thus, these large complex chips are only cost-effective if they can be sold in large volumes. This need for a large market runs counter to the drive for efficient, narrowly- focused, custom hardware solutions.

To fill the need for widely applicable computing designs, a number of more general-purpose processors are targeted at a class of problems, rather than at specific applications. Tri-media, Equator, Mpact, IRAM, and many other projects are all attempts to create general purpose computing engine for multimedia applications. However, these attempts to create more universal computing elements have some limitations. First, these machines have been optimized for applications where the parallelism can be expressed at the instruction level using either VLIW or vector engines. However, they would not be very efficient for applications that lacked parallelism at this level, but had, for example, thread level parallelism. Second, their globally shared resource models (shared multi-ported registers and memory) will be increasingly difficult

to implement in future technologies in which on-chip communication costs are appreciable. Finally, since these machines are generally compromise solutions between true signal processing engines and general-purpose processors, their efficiency at doing either task suffers.

On the other hand, the need for scalable architectures has also led to proposals for modular, explicitly parallel architectures that typically consist of a number of processing elements and memories on a die connected together by a network. The modular nature of these designs ensures that wire lengths shrink as technologies improve, allowing wire and gate delays to scale at roughly the same rate. Additionally, the replication consumes the growing number of transistors. The multiple processing elements take advantage of both instruction-level and thread-level parallelism. One of the most prominent architectures in this class is the MIT Raw project, which focuses on the development of compiler technologies that take advantage of exposed low-level hardware.

Smart Memories combines the benefits of both approaches to create a partitioned, explicitly parallel, reconfigurable architecture for use as a future universal computing element. Since different application spaces naturally have different communication patterns and memory needs, finding a single topology that fits well with all applications is very difficult. Rather than trying to find a general solution for all applications, we tailor the appearance of the on-chip memory, interconnection network, and processing elements to better match the application requirements. We leverage the fact that long wires in current (and future) VLSI chips require active repeater insertion for minimum delay.

The presence of repeaters means that adding some reconfigurable logic to these wires will only modestly impact their performance. Reconfiguration at this level leads to coarser-grained configurability than previous reconfigurable architectures, most of which were at least in part based on FPGA implementations. Compared to these systems, Smart Memories trades away some flexibility for lower overheads, more familiar programming models, and higher efficiency.

1.1 SMART MEMORIES OVERVIEW

At the highest level, a Smart Memories chip is a modular computer. It contains an array of processor tiles and on-die DRAM memories connected by a packet-based, dynamically-routed network (Figure 1). The network also connects to high-speed links on the pins of the chip to allow for the construction of multi-chip systems. Most of the initial hardware design work in the

Smart Memories project has been on the processor tile design and evaluation, so this paper focuses on these aspects.

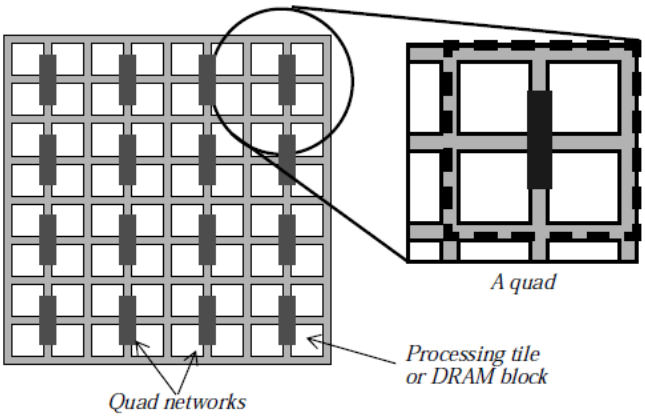


Fig 1.1 Smart memory chip

The organization of a processor tile is a compromise between VLSI wire constraints and computational efficiency. Our initial goal was to make each processor tile small enough so the delay of a repeated wire around the semi-perimeter of the tile would be less then a clock cycle. This leads to a tile edge of around 2.5mm in a 0.1mm technology [7]. This sized tile can contain a processor equivalent to a MIPS R5000 [19], a 64-bit, 2-issue, in-order machine with 64KB of on-die cache. Alternately, this area can contain 2-4MB of embedded DRAM depending on the assumed cell size. A 400mm2 die would then hold about 64 processor tiles, or a lesser number of processor tiles and some DRAM tiles.

CHAPTER TWO

2.0 LITERATURE REVIEW

A Smart Memories chip is made up of many processing tiles, each containing local memory, local interconnect, and a processor core. For efficient computation under a wide class of possible applications, the memories, the wires, and the computational model can all be altered to match the applications. To show the applicability of this design, two very different machines at opposite ends of the architectural spectrum, the Imagine stream processor and the Hydra speculative multiprocessor, are mapped onto the Smart Memories computing substrate. Simulations of the mappings show that the Smart Memories architecture can successfully map these architectures with only modest performance degradation.

2.1 MEMORY

Memory refers to the computer hardware devices involved to store information for immediate use in a computer; it is synonymous with the term "primary storage". Computer memory operates at a high speed, for example random-access memory (RAM), as a distinction from storage that provides slow-to-access program and data storage but offers higher capacities. If needed, contents of the computer memory can be transferred to secondary storage, through a memory management technique called "virtual memory". An archaic synonym for memory is store.

The term "memory", meaning "primary storage" or "main memory", is often associated with addressable semiconductor memory, i.e. integrated circuits consisting of silicon-based transistors, used for example as primary storage but also other purposes in computers and other digital electronic devices. There are two main kinds of semiconductor memory, volatile and non-volatile. Examples of non-volatile memory are flash memory (used as secondary memory) and ROM, PROM, EPROM and EEPROM memory (used for storing firmware such as BIOS). Examples of volatile memory are primary storage, which is typically dynamic random-access memory(DRAM), and fast CPU cache memory, which is typically static random-access memory (SRAM) that is fast but energy-consuming, offering lower memory areal density than DRAM. Most semiconductor memory is organized into memory cells or bistable flip-flops, each storing one bit (0 or 1). Flash memory organization

includes both one bit per memory cell and multiple bits per cell (called MLC, Multiple Level Cell). The memory cells are grouped into words of fixed word length, for example 1, 2, 4, 8, 16, 32, 64 or 128 bit. Each word can be accessed by a binary address of N bit, making it possible to store 2 raised by N words in the memory. This implies that processor registers normally are not considered as memory, since they only store one word and do not include an addressing mechanism. Typical secondary storage devices are hard disk drives and solid-state drives. In the early 1940s, memory technology often permit a capacity of a few bytes. The first electronic programmable digital computer, the ENIAC, using thousands of octal-base radio vacuum tubes, could perform simple calculations involving 20 numbers of ten decimal digits which were held in the vacuum tube accumulators. The next significant advance in computer memory came with acoustic delay line memory, developed by J. Presper Eckert in the early 1940s. Through the construction of a glass tube filled with mercury and plugged at each end with a quartz crystal, delay lines could store bits of information in the form of sound waves propagating through mercury, with the quartz crystals acting as transducers to read and write bits. Delay line memory would be limited to a capacity of up to a few hundred thousand bits to remain efficient.

Two alternatives to the delay line, the Williams tube and Selectron tube, originated in 1946, both using electron beams in glass tubes as means of storage. Using cathode ray tubes, Fred Williams would invent the Williams tube, which would be the first random-access computer memory. The Williams tube would prove more capacious than the Selectron tube (the Selectron was limited to 256 bits, while the Williams tube could store thousands) and less expensive. The Williams tube would nevertheless prove to be frustratingly sensitive to environmental disturbances. Efforts began in the late 1940s to find non-volatile memory. Jay Forrester, Jan A. Rajchman and An Wang developed magnetic core memory, which allowed for recall of memory after power loss. Magnetic core memory would become the dominant form of memory until the development of transistor-based memory in the late 1960s. Developments in technology and economies of scale have made possible so-called Very Large Memory (VLM) computers. The term "memory" when used with reference to computers generally refers to Random Access Memory or RAM.

2.2 VOLATILE MEMORY

Volatile memory is computer memory that requires power to maintain the stored information. Most modern semiconductor volatile memory is either static RAM (SRAM) or dynamic RAM (DRAM). SRAM retains its contents as long as the power is connected and is easy for interfacing, but uses six transistors per bit. Dynamic RAM is more complicated for interfacing and control, needing regular refresh cycles to prevent losing its contents, but uses only one transistor and one capacitor per bit, allowing it to reach much higher densities and much cheaper per-bit costs. SRAM is not worthwhile for desktop system memory, where DRAM dominates, but is used for their cache memories. SRAM is commonplace in small embedded systems, which might only need tens of kilobytes or less. Forthcoming volatile memory technologies that aim at replacing or competing with SRAM and DRAM include Z-RAM and A-RAM.

2.3 NON-VOLATILE MEMORY

Non-volatile memory is computer memory that can retain the stored information even when not powered. Examples of non-volatile memory include read-only memory (see ROM), flash memory, most types of magnetic computer storage devices (e.g. hard disk drives, floppy disksand magnetic tape), optical discs, and early computer storage methods such as paper tape and punched cards. Forthcoming non-volatile memory technologies include FeRAM, CBRAM, PRAM, SONOS, RRAM, racetrack memory, NRAM, 3D XPoint, andmillipede memory.

2.4 EARLY COMPUTER SYSTEMS

In early computer systems, programs typically specified the location to write memory and what data to put there. This location was a physical location on the actual memory hardware. The slow processing of such computers did not allow for the complex memory management systems used today. Also, as most such systems were single-task, sophisticated systems were not required as much. This approach has its pitfalls. If the location specified is incorrect, this will cause the computer to write the data to some other part of the program. The results of an error like this are unpredictable. In some cases, the incorrect data might overwrite memory used by the operating system. Computer crackers can take advantage of this to create viruses and malware.

2.5 VIRTUAL MEMORY

Virtual memory is a system where all physical memory is controlled by the operating system. When a program needs memory, it requests it from the operating system. The operating system then decides what physical location to place the memory in. This offers several advantages. Computer programmers no longer need to worry about where the memory is physically stored or whether the user's computer will have enough memory. It also allows multiple types of memory to be used. For example, some memory can be stored in physical RAM chips while other memory is stored on a hard drive (e.g. in a swapfile), functioning as an extension of the cache hierarchy. This drastically increases the amount of memory available to programs. The operating system will place actively used memory in physical RAM, which is much faster than hard disks. When the amount of RAM is not sufficient to run all the current programs, it can result in a situation where the computer spends more time moving memory from RAM to disk and back than it does accomplishing tasks; this is known as thrashing. Virtual memory systems usually include protected memory, but this is not always the case.

2.6 PROTECTED MEMORY

Protected memory is a system where each program is given an area of memory to use and is not permitted to go outside that range. Use of protected memory greatly enhances both the reliability and security of a computer system. Without protected memory, it is possible that a bug in one program will alter the memory used by another program. This will cause that other program to run off of corrupted memory with unpredictable results. If the operating system's memory is corrupted, the entire computer system may crash and need to be rebooted. At times programs intentionally alter the memory used by other programs. This is done by viruses and malware to take over computers.

Protected memory assigns programs their own areas of memory. If the operating system detects that a program has tried to alter memory that does not belong to it, the program is terminated. This way, only the offending program crashes, and other programs are not affected by the error. Protected memory systems almost always include virtual memory as well.

2.7 SENSORY MEMORY

Sensory memory holds sensory information less than one second after an item is perceived. The ability to look at an item and remember what it looked like with just a split second of observation, or memorization, is the example of sensory memory. It is out of cognitive control and is an automatic response. With very short presentations, participants often report that they seem to "see" more than they can actually report. The first experiments exploring this form of sensory memory were precisely conducted by George Sperling(1963)[2] using the "partial report paradigm". Subjects were presented with a grid of 12 letters, arranged into three rows of four. After a brief presentation, subjects were then played either a high, medium or low tone, cuing them which of the rows to report. Based on these partial report experiments, Sperling was able to show that the capacity of sensory memory was approximately 12 items, but that it degraded very quickly (within a few hundred milliseconds). Because this form of memory degrades so quickly, participants would see the display but be unable to report all of the items (12 in the "whole report" procedure) before they decayed. This type of memory cannot be prolonged via rehearsal.

Three types of sensory memories exist. Iconic memory is a fast decaying store of visual information; a type of sensory memory that briefly stores an image which has been perceived for a small duration. Echoic memory is a fast decaying store of auditory information, another type of sensory memory that briefly stores sounds that have been perceived for short durations. Haptic memory is a type of sensory memory that represents a database for touch stimuli.

2.8 SHORT-TERM MEMORY

Short-term memory is also known as working memory. Short-term memory allows recall for a period of several seconds to a minute without rehearsal. Its capacity is also very limited: George A. Miller (1956), when working at Bell Laboratories, conducted experiments showing that the store of short-term memory was 7±2 items (the title of his famous paper, "The magical number 7±2"). Modern estimates of the capacity of short-term memory are lower, typically of the order of 4–5 items;[4] however, memory capacity can be increased through a process called chunking.[5] For example, in recalling a ten-digit telephone number, a person could chunk the digits into three groups: first, the area code (such as 123), then a three-digit chunk (456) and lastly a four-digit chunk (7890). This method of remembering telephone numbers is far more effective than attempting to remember a string of 10 digits; this is because we are able to chunk

the information into meaningful groups of numbers. This may be reflected in some countries in the tendency to display telephone numbers as several chunks of two to four numbers.

Short-term memory is believed to rely mostly on an acoustic code for storing information, and to a lesser extent a visual code. Conrad (1964)[6] found that test subjects had more difficulty recalling collections of letters that were acoustically similar (e.g. E, P, D). Confusion with recalling acoustically similar letters rather than visually similar letters implies that the letters were encoded acoustically. Conrad's (1964) study, however, deals with the encoding of written text; thus, while memory of written language may rely on acoustic components, generalizations to all forms of memory cannot be made.

2.9 LONG-TERM MEMORY

The storage in sensory memory and short-term memory generally has a strictly limited capacity and duration, which means that information, is not retained indefinitely. By contrast, long-term memory can store much larger quantities of information for potentially unlimited duration (sometimes a whole life span). Its capacity is immeasurable. For example, given a random seven-digit number we may remember it for only a few seconds before forgetting, suggesting it was stored in our short-term memory. On the other hand, we can remember telephone numbers for many years through repetition; this information is said to be stored in long-term memory. While short-term memory encodes information acoustically, long-term memory encodes it semantically: Baddeley (1966) discovered that, after 20 minutes, test subjects had the most difficulty recalling a collection of words that had similar meanings (e.g. big, large, great, huge) long-term. Another part of long-term memory is episodic memory, "which attempts to capture information such as 'what', 'when' and 'where'". With episodic memory, individuals are able to recall specific events such as birthday parties and weddings.

Short-term memory is supported by transient patterns of neuronal communication, dependent on regions of the frontal lobe (especially dorsolateral prefrontal) and the parietal lobe. Long-term memory, on the other hand, is maintained by more stable and permanent changes in neural connections widely spread throughout the brain. The hippocampus is essential (for learning new information) to the consolidation of information from short-term to long-term memory, although it does not seem to store information itself. It was thought that without the hippocampus new memories were unable to be stored into long-term memory and that there would be a very

short attention span, as first gleaned from patient Henry Molaison [9] after what was thought to be the full removal of both his hippocampi. More recent examination of his brain, post-mortem, shows that the hippocampus was more intact than first thought, throwing theories drawn from the initial data into question. The hippocampus may be involved in changing neural connections for a period of three months or more after the initial learning. Research has suggested that long-term memory storage in humans may be maintained by DNA methylation,[10] or prions.[11]

CHAPTER THREE

ARCHITECTURE/METHODOLOGY

3.0 TILE ARCHITECTURE

A Smart Memories tile consists of a reconfigurable memory system; a crossbar interconnection network; a processor core; and a quad network interface (Figure 2). To balance computation, communication, and storage, we allocated equal portions of the tile to the processor, interconnect, and memory.

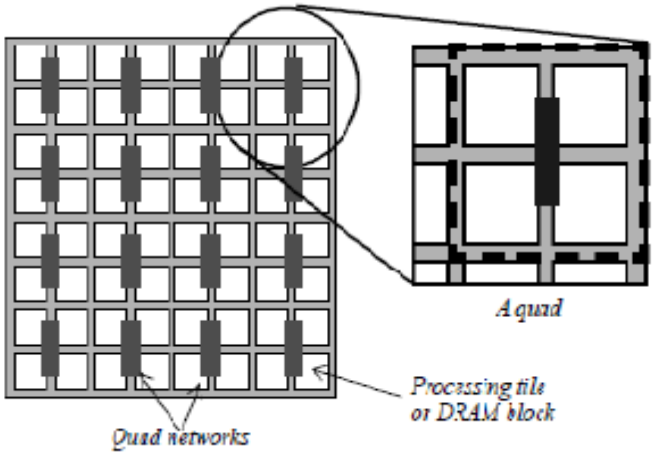


Fig 3.1 Smart Memory Chips

3.1 MEMORY SYSTEM

The memory system is of growing importance in processor design. Different applications have different memory access patterns and thus require different memory configurations to optimize performance. Often these different memory structures require different control logic and status bits. Therefore, a memory system that can be configured to closely match the appA recent study of SRAM design shows that the optimal block size for building large SRAMs is small, around a few KB. Large SRAMs are then made up of many of these smaller SRAM blocks. We leverage this naturally hierarchical design to provide low overhead reconfigurability. The basic memory mat size of 8KB is chosen based on a study of decoder and I/O overheads and an architectural study of the smallest memory granularity needed. Allocating a third of the tile area to memory allows for 16indent 8KB memory mats, a total of 128KB per tile. Each mat is a 1024x64b logical

memory array that can perform reads, writes, compares, and read-modify-writes. All operations are byte maskable depenlication demands is desirable. In addition to the memory array, there is configurable logic in the address and data paths. In the address path, the mats take in a 10-bit address and a 4-bit opcode to determine what operation is to be performed. The opcode is decoded using a reconfigurable logic block that is set up during the hardware configuration. The memory address decoder can use the address input directly or can be set in Auto-increment/decrement streaming mode. In this mode, the mat stores the starting index, stream count, and stride. On each streaming mode request, the mat accesses the next word of the stream until reaching the end of the stream.

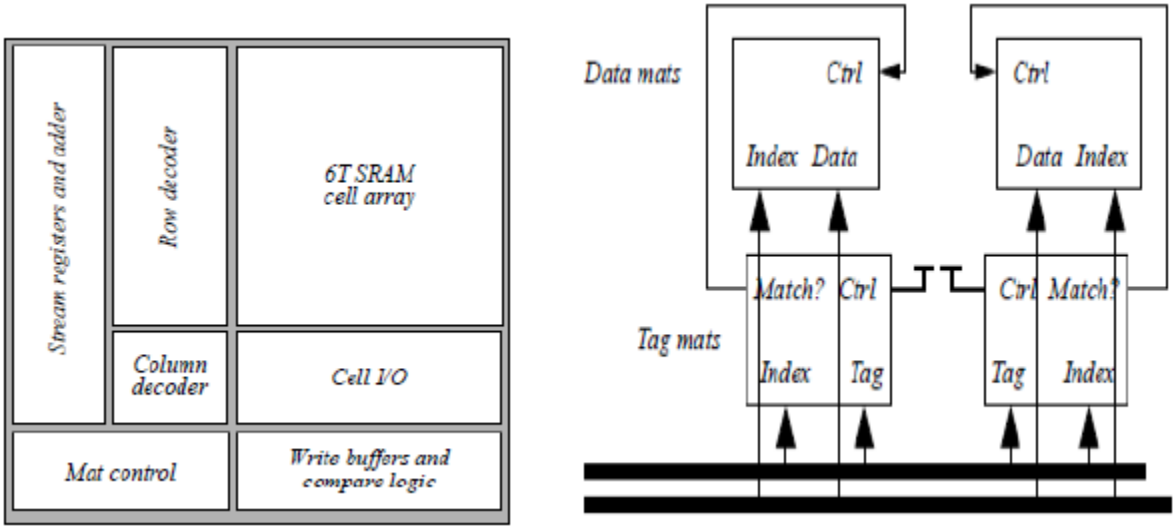


Figure 3.1.1 Memory mat detail Figure 4. Mats configured as 2-way set-associative cache

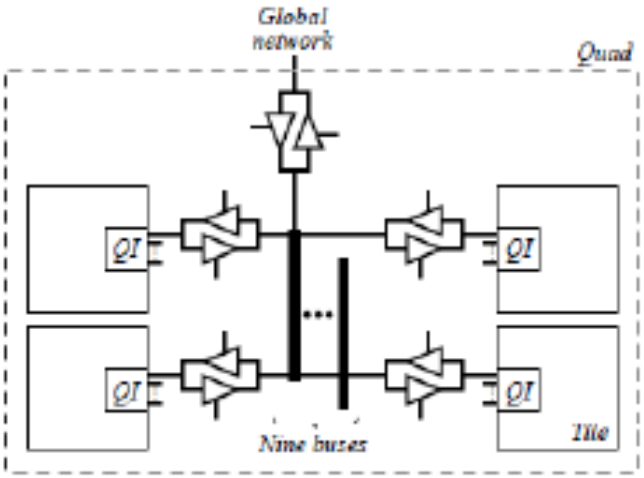
In the data path, each 64-bit word is associated with a valid bit and a 4-bit configurable control field. These bits can be used for storing data state such as cache LRU or coherence bits. They are dual ported to allow read-modify-write operations each cycle and can be flash cleared via special opcodes. Each mat has a write buffer to support pipelined writes and to enable conditional write operations (e.g. in the case of a cache write). Mats also contain logic in the output read path for comparisons, so they can be used as cache tag memory The Smart Memories mats can be configured to implement a wide variety of caches, from simple, single-ported, direct-mapped structures to set associative, multi-banked designs. Figure 4 gives an example of four memory mats configured as a two-way set associative cache with two of the mats acting as the tag

memories and two other mats acting as the data memories. The mats can also be configured as local scratchpad memories or as vector/stream register files. These simpler configurations have higher efficiency and can support higher total memory bandwidth at a lower energy cost per access. Associated with the memory, but located in the two load-store units of the processor, are direct-memory access (DMA) engines that generate memory requests to the quad and global interconnection networks. When the memory mats are configured as caches, the DMA engines generate cache fill/spill requests. When the mats are configured for streaming or vector memories, the DMA engines generate the needed gather/scatter requests to fill the memory with the desired data.

3.2 INTERCONNECT

To connect the different memory mats to the desired processor or quad interface port, the tile contains a dynamically routed crossbar which supports up to 8 concurrent references. The processor and quad interface generate requests for data, and the quad interface and memories service those requests. The crossbar does not interconnect different units of the same type (e.g. memory mat to memory mat communication is not supported in the crossbar).The quad interconnection network, shown in Figure 5, connects the four tiles in a quad together. The network consists of 9 64-bit multicast buses on which any of the 4 tiles or the global network can send or receive data. These buses may also be configured as half word buses. In addition to these buses, a small number of control bits are broadcast to update state, atomically stall the processors, and arbitrate for the buses.

The quad interface on each tile connects the internal tile crossbar to the quad network, thus mediating all communication to and from the tile.



3.3 PROCESSOR

The processor portion of a Smart Memories tile is 64-bit processing engines with reconfigurable instruction format/decode. The computation resources of the tile consist of two integer clusters and one floating point (FP) cluster. The arrangement of these units and the FP cluster unit mix are shown in Figure 6. Each integer cluster consists of an ALU, register file, and load/store unit. This arithmetic unit mix reflects a trade-off between the resources needed for a wide range of applications and the area constraints of the Smart Memories tile [2-5]. Like current media processors, all 64-bit FP arithmetic units can also perform the corresponding integer operations and all but the divide/sqrt unit perform sub word arithmetic The high operand bandwidth needed in the FP cluster to sustain parallel issue of operations to all functional units is provided by local register files (LRFs) directly feeding the functional units and a shared register file with two read and one write ports. The LRF structure provides the necessary bandwidth more efficiently in terms of area, power, and access time compared to increasing the number of ports to the shared register file. The shared FP register file provides a central register pool for LRF overflows and shared constants. A network of result and operand buses transfers data among functional units and the register files.

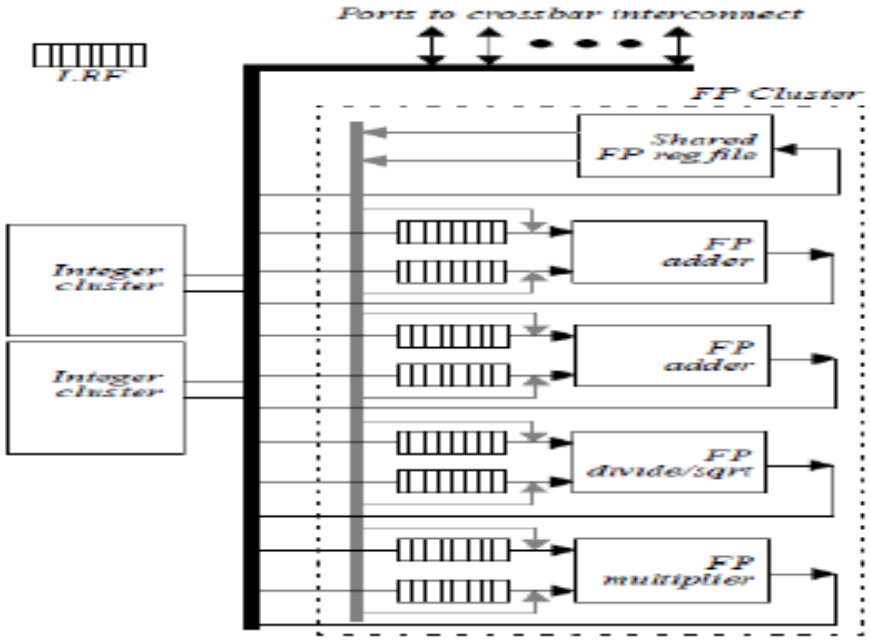


Fig 3.2 Smart memories compute chips

Finally, a 32-bit RISC-style instruction set is available for applications that do not exhibit much ILP. To extract thread-level parallelism of such applications, each tile can sustain two concurrent, independent threads. The two threads on a tile are asymmetric. The primary thread may perform integer or FP computations and can issue up to two instructions per cycle, while the secondary thread is limited to integer operations at single-issue. The secondary thread is intended for light-duty tasks or for system-level support functions. For example, lower communication costs on systems with multiple processing nodes on a chip permit dynamic data and task migration to improve locality and load balance at a much finer grain than is practical in conventional multi-processors. The increased communication volume and resource usage tracking for such operations can easily be delegated to the secondary thread. The two threads are assumed to be independent and any communication must be explicitly synchronized.

3.4 GENERAL ARCHITECTURE

Figure 3-1 shows high-level logical organization of the universal memory system. It consists of distinct memory elements arranged in levels of hierarchy, connected by communication channels. There are three main elements in the memory system: memories as storage locations, their associated controllers, and communication channels connecting the controllers together. In actual implementation, elements might be organized and grouped differently, however the logical view of any implementation is similar to Figure 3-1. Note that in this figure we assume processors are located at the top and main memory at the bottom. Memories and controllers closer to the processors hence are referred to as higher-level memories or controllers and the ones farther from processor are referred to as lower level ones. The execution model of the system is based on exchanging messages between the different components. Operations start by processors emitting memory instructions to their corresponding Load/Store Unit (LSU). At each level of hierarchy, controllers receive and decode messages, then execute a set of operations to handle the received message. Executed operations might include accesses to the local memory as well as composing and sending new messages to other controllers. The combined result of the operations executed by all controllers involved, results in our desired outcome, satisfying a processor's memory request in compliance with the system's memory model.

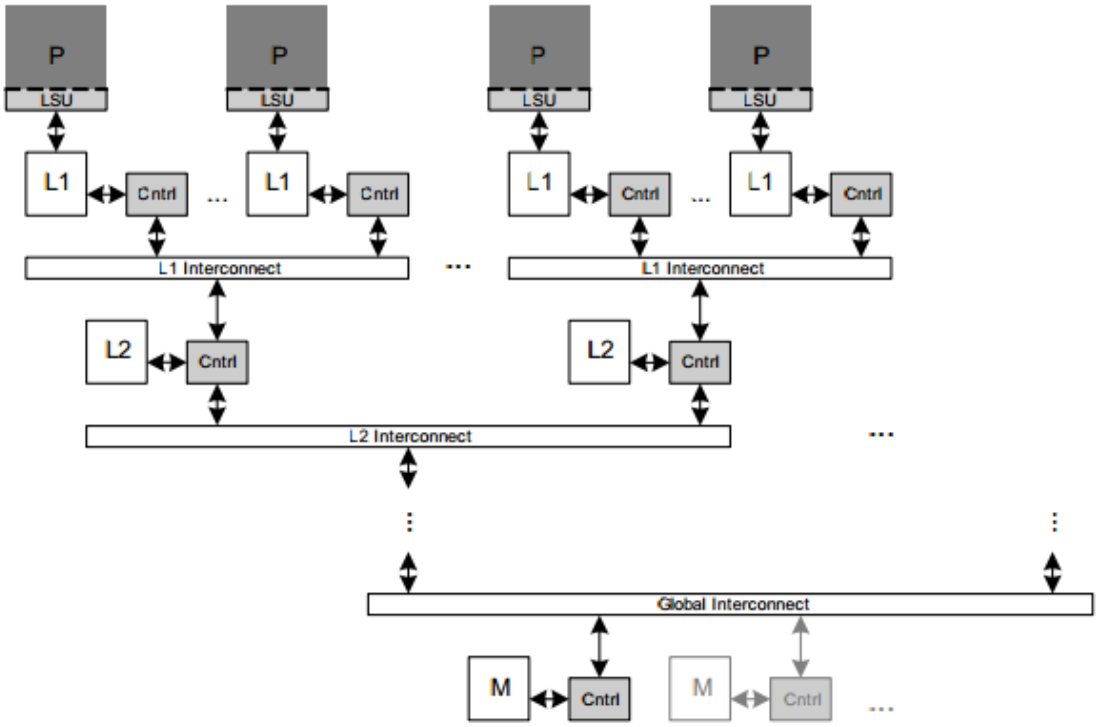


Figure 3.4: High-level architecture of the memory system

Four major categories of messages are recognized in the system:

Data Transfer Requests: Data transfers involve copying a block of data from one memory location to another. Data transfer messages usually travel downwards (towards main memory) in the memory hierarchy, attempting to read/write data blocks from/to larger, slower memories to faster smaller ones. They might also copy data between memories at the same level of hierarchy. Transfer requests can be short messages that attempt to acquire a data block for the local memory, such as cache misses and DMA gather requests, or long messages writing a data block to a remote memory such as write-backs and DMA scatter requests.

Data Transfer Replies: Transfer replies are either long messages carrying requested data block, such as cache refills or short acknowledgement messages indicating that data copy operation is completed (e.g. write-back acknowledgements). State Inquiry/Update Requests: The purpose of these messages is to query and adjust the state information associated with a data blocks. They are usually sent by a controller to the controllers in the same or higher level and travel upward in the hierarchy, where data copies are located. These messages are short, containing no data, but depending on the state information they acquire, their corresponding reply might contain data in addition to the acquired state information. Most common examples of such messages are

coherence requests or bus snoops requests. State Inquiry/Update Replies: Reply messages for state inquiries contain the state information of the target data block. They also might bring back the data portion of the target block depending on the state in which they find it. Examples are replies to coherence messages that carry data and/or ownership information.

In the following, we describe the memory system resources and the capabilities that they should provide in more details.

3.5 STORAGE ELEMENTS

Memories at each level of hierarchy must not only store the application data, but also keep the state information that system associates with data. Our logical model does not make any specific assumptions about organization of the memories at each level, such as granularity of the data storage (word, byte, etc.), size of the memory, number of banks, or even number of state bits associated (However we assume that there are enough state bits available to implement the desired memory model). The only requirement is that all the storage locations have unique addresses across the system and are addressable by each and every processor. If processors only use main memory addresses (e.g. when local memories are used as caches), then at each level of the hierarchy controllers convert the processor generated address to the unique physical address of the local memory they are associated with before attempting to access the local memory. Memories at each level of the hierarchy should support the basic read and write operations on the data and state information they store. As it will be discussed later in this chapter, data accesses in the memories are usually preceded by accesses to their associated state information. This is due to the fact that state information oftentimes protects the data by encoding necessary access permissions. Before attempting the data access, processors and controllers must check the state information to ensure that they have the required permissions. Therefore, as an optimization, the memories can overlap data and state accesses, provided that the data access is conditioned on having correct state information. This necessitates support for conditional operations on data in the memories, as well as the basic means for propagating and exchanging state information between them. Given such optimizations, sequential operations on the state and data can be converted into concurrent operations, reducing the latency of the overall memory access time which is particularly advantageous for L1 memories due to the frequent processors accesses.

3.6 ADVANTAGES AND DISADVANTAGES OF SMART MEMORIES

A**DVANTAGES**

1. Reduced chip I/O bandwidth
2. High performance and low latency
3. Feature rich, flexible and programmable
4. Lower cost
5. One chip for several functions

DISADVANTAGES

1. Packet Processing Bottlenecks
2. Data away from compute
3. I/O and memory bandwidth
4. Smart Memory Keep compute close to data
5. Keep locking close to data
6. Provide inter-memory connect

CHAPTER FOUR

SUMMARY AND CONCLUSION

4.0 SUMMARY

While Smart Memories allows mapping of different memory protocols, the flexible mechanisms added to provide reconfigurability impact both the performance and physical characteristics of the system. Our studies show that while the performance overhead induced by these mechanisms is modest, less than 20% in most of the cases, the increase in the system's area and power is not negligible. However, most of this increase is resulted from our poor implementation of system's configuration storage, using flip-flops for implementing memories and TCAMs. These inefficiencies can be removed and a better implementation of the system can be achieved by using memory macros and custom structures, as shown by Mai et al for the case of a reconfigurable memory mat.

4.1 CONCLUSION

Continued technology scaling causes a dilemma while computation gets cheaper, the design of computing devices becomes more expensive, so new computing devices must have large markets to be successful. Smart Memories addresses this issue by extending the notion of a program. In conventional computing systems the memories and interconnect between the processors and memories is fixed, and what the programmer modifies is the code that runs on the processor. While this model is completely general, for many applications it is not very efficient. In Smart Memories, the user can program the wires and the memory, as well as the processors. This allows the user to configure the computing substrate to better match the structure of the applications, which greatly increases the efficiency of the resulting solution.

REFERENCES

Kaplinsky C (2001). A New Microsystem Architecture for the Internet Era. Presented in Microprocessor Forum, Oct. 1999.Available at www.Wikeipedia.com

Diefendorff and P. Dubey (1999) How Multimedia Workloads Will Change Processor Design. IEEE Computer, pages 43- 45, Sept. 1997.

Slavenberg, S (2000). The Trimedia TM-1 PCI VLIW Media Processor. In Proceedings of Hot Chips 8, 1996.

Lucas L (1999). High Speed Low Cost TM1300 Trimedia Enhanced PCI VLIW Mediaprocessor. In Proceedings of Hotchips 11, pages 111-120, Aug. 1999.

Donnell J (1999). MAP1000A: A 5W, 230MHz VLIW Mediaprocessor.In Proceedings of Hot Chips 11, pages 95-109, Aug. 1999.

Kalapathy P (1997). Hardware-Software Interactions on MPACT. IEEE Micro, pages 20-26, Mar. 1997.

Kozyrakis C (1997) et al. Scalable Processors in the Billion-transistor Era: IRAM. IEEE Computer, pages 75-78, Sept. 1997.

Horowitz M (1999). The Future of Wires. SRC White Paper: Interconnect Technology Beyond the Roadmap, 1999 Available at http://www.src.org/cgibin/deliver.cgi/sarawp.pdf?/areas/nis/sarawp.pdf).

Matzke D (1992). Will Physical Scalability Sabotage PerformanceGains? IEEE Computer, pages 37-9, Sept. 1997.

Waingold W(1997). Baring It All to Software: Raw Machines. IEEE Computer, pages 86-93, Sept.1997.