International Journal of Modern Trends in Engineering and Research
www.ijmter.com
e-ISSN No.:2349-9745, Date: 2-4 July, 2015

Smart Memories

Yogita S. Hon[1], Shweta S. Gosavi[2]
[1]*Department of E&TC Engineering, S.N.J.B's KBJ COE, Chandwad, yogitarathod@gmail.com*
[2]*Department of E&TC Engineering, S.N.J.B's KBJ COE, Chandwad, shwetagosavis2@ymail.com*

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.
Keywords- Hydra speculative multiprocessor, reconfigurable architecture.)

I. INTRODUCTION

Continuous transmission robots are the robots designed to present high quality constraints safely to human users [1]. The initial driving force behind the invention of robots was to meet the need for a robot that could interact safely with automobile assemblers. There are presently several different robots, but the defining feature common to all robots is the continuously variable transmission (CVT) used in place of traditional joints of cobot. The CVTs separates the robot's speed and direction kinematically. For example, the unicycle cobot [1] have one steered wheel rolling on a flat plane. The steered wheel is a CVT that controls the x and y axis transitional velocity ratio as a continuous function of the wheel's heading angle.

The actuator, instead of controlling the rotational velocity of the wheel, is used to control the motion of the wheel. From today's standard workloads, often containing highly data-parallel streaming behavior [1]. While the applications will demand ever-growing compute performance, power (ops/W) and computational efficiency (ops/$) 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 [2, 3], Equator [4], Impact [5], IRAM [6], and many other projects are all attempts to create general purpose computing engine for multi-media 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 [7,8]. Finally, since these machines are generally compromise solutions between true signal processing engines and general-purpose processors, their efficiency at doing either task suffers. 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 [11-18]. Compared to these systems, Smart Memories trades away some flexibility for lower overheads, more familiar programming models, and higher efficiency.

II. 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.

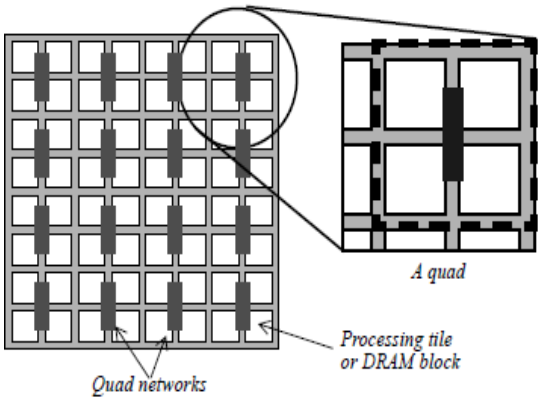

Figure 1. Smart Memories chip

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 than 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, 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 Since large-scale computations may require more computation power than what is contained in a single processing tile, we cluster four processor tiles together into a "quad" and provide a low-overhead, intra-quad, interconnection network. Grouping the tiles into quads also makes the global interconnection network more efficient by reducing the number of global network interfaces and thus the number of hops between processors. Our goal in the tile design is to create a set of components that will span as wide an application set as possible. In current architectures, computational elements are somewhat standardized; today, most processors have multiple segmented functional units to increase efficiency when working on limited precision numbers.

III. 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.

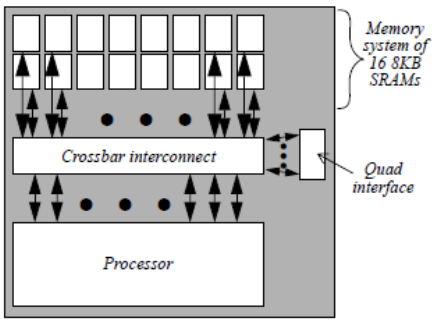

Figure 2. Tile floor plan

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 16 indent 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 depen-lication 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.

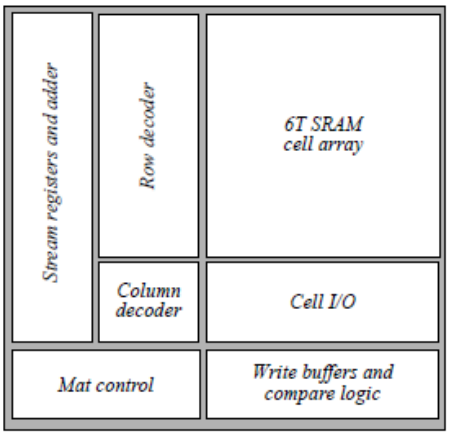

Figure 3. Memory mat detail

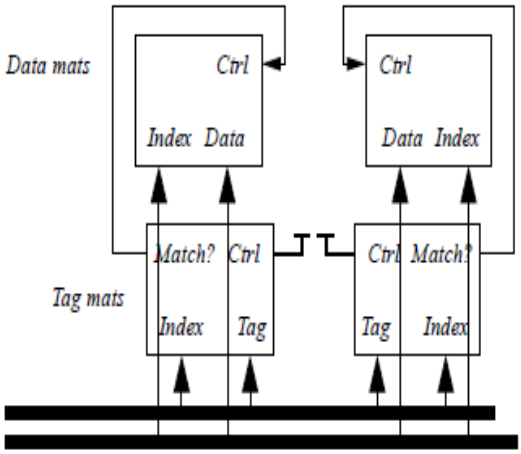

Figure 4. Mats configured as 2-way set-associative cache

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.

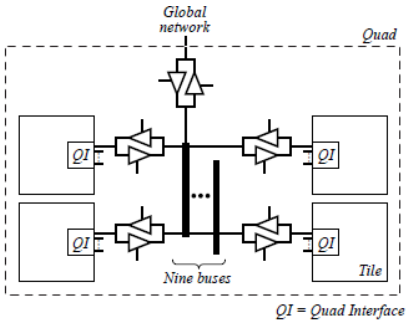

Figure 5. Quad interconnection network

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.

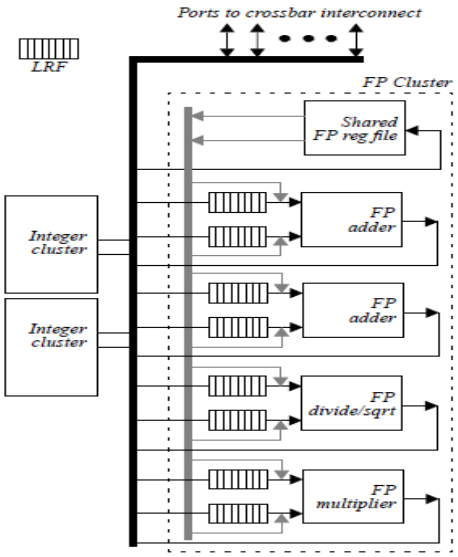

Figure 6. Smart Memories compute resources

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.

IV. 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

[1] Diefendorff and P. Dubey. How Multimedia Workloads Will Change Processor Design. IEEE Computer, pages 43- 45, Sept. 1997.
[2] G. A. Slavenberg, et al. The Trimedia TM-1 PCI VLIW Media Processor. In Proceedings of Hot Chips 8, 1996.
[3] L. Lucas. High Speed Low Cost TM1300 Trimedia Enhanced PCI VLIW Mediaprocessor. In Proceedings of Hotchips 11, pages 111-120, Aug. 1999.
[4] J. O'Donnell. MAP1000A: A 5W, 230MHz VLIW Mediaprocessor.In Proceedings of Hot Chips 11, pages 95-109, Aug. 1999.
[5] P. Kalapathy. Hardware-Software Interactions on MPACT. IEEE Micro, pages 20-26, Mar. 1997.
[6] C. Kozyrakis, et al. Scalable Processors in the Billion-transistor Era: IRAM. IEEE Computer, pages 75-78, Sept. 1997.
[7] M. Horowitz, et al. The Future of Wires. SRC White Paper: Interconnect Technology Beyond the Roadmap, 1999 (http://www.src.org/cgi-bin/deliver.cgi/sarawp.pdf?/areas/nis/sarawp.pdf).
[8] D. Matzke, et al. Will Physical Scalability Sabotage PerformanceGains? IEEE Computer, pages 37-9, Sept. 1997.
[9] C. Kaplinsky. A New Microsystem Architecture for the Internet Era. Presented in Microprocessor Forum, Oct. 1999.
[10] Waingold, et al. Baring It All to Software: Raw Machines. IEEE Computer, pages 86-93, Sept. 1997.

