A Double-Data Rate (DDR) Processing-in-Memory (PIM) Device with WideWord Floating-Point Capability

Tim Barrett, Sumit Mediratta, Taek-Jun Kwon, Ravinder Singh, Sachit Chandra, Jeff Sondeen, Jeffrey Draper

University of Southern California, Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, CA 90292
{jtb, sumitm, tjkwon, ravinder, chandra, sondeen, draper}@isi.edu

Abstract:

The Data-Intensive Architecture (DIVA) system incorporates Processing-In-Memory (PIM) chips as smart-memory coprocessors to a microprocessor. This architecture exploits inherent memory bandwidth both on chip and across the system to target several classes of bandwidth-limited applications. A recently developed PIM chip in TSMC 0.18μm technology incorporates a DDR SDRAM interface for its inclusion in commodity systems, such as the HP zx6000 workstation used on this project. Each PIM chip includes eight single-precision floating-point units (FPU) in the WideWord pipeline, enabling significant speedups in the target system. This paper focuses on the integration of new subcomponents into the PIM chip design, system integration, and measured system results, demonstrating the significant GFLOP/W feature offered by PIM computing.

1. Introduction

The Data-IntensiVe Architecture (DIVA) project [3] explored architecture design based on the underlying technology of embedded DRAM. The DIVA approach replaces the memory system of a conventional workstation with "smart memories" using the concept of processing-in-memory (PIM). The goal of the project is to significantly reduce the ever-increasing processor-memory bandwidth bottleneck in conventional systems. System bandwidth limitations are overcome in a number of ways, most notably: (1) the tight coupling of a single PIM processor with an on-chip memory bank; and (2) the use of a separate chip-to-chip interconnect for direct communication between nodes on different chips that bypasses the host system bus. This combination of features targets applications that are severely impacted by the processor-memory bottleneck in conventional systems, mostly due to little spatial or temporal data locality resulting in poor cache utility.

This paper reports the implementation and experiments performed for a PIM device integrated into a commodity system, namely a HP zx6000 workstation. Given the PIM's first-order goal of appearing as a memory device in a standard memory system of commodity computers, the PIM device must contain a standard memory interface. An initial PIM prototype developed earlier [3,4] used a synchronous DRAM (SDRAM) interface. To track developments in memory technology, a double-data rate (DDR) SDRAM interface was developed and incorporated into the PIM device discussed in this paper. To provide even further flexibility, an SDRAM interface was also maintained in the chip implementation, which allows a system to select either an SDRAM or DDR SDRAM interface for the PIM device to be compatible with the resident memory system. Additionally, the PIM device discussed in this paper greatly improves over earlier prototype work with the incorporation of eight IEEE-754 single-precision floating-point units (FPU) in the WideWord pipeline. The design of this area-efficient, technology-portable FPU has appeared in earlier work [10,11]; its integration with the WideWord pipeline is reported in this paper. The PIM device reported in this paper also improves over the earlier prototype in that it incorporates an address translation unit [2], enabling in-memory virtual addressing, a key feature of the DIVA concept. Finally, this paper reports system integration issues and experiment results, proving not only the speedup offered by PIM computing but also the power-efficient manner in which the speedup is achieved.

The remainder of this paper is organized as follows. Section 2 presents a brief background of the DIVA concept followed by a description of the PIM chip implementation in Section 3. Section 4 presents the details of integrating these PIM devices into an Itanium2-based HP workstation. Section 5 reports the results of lab experiments followed by future work and concluding remarks in Section 6.

2. PIM Architecture Overview

A driving principle of the DIVA system architecture is efficient use of PIM technology while requiring a smooth migration path for software. This principle demands integration of PIM features into conventional systems as seamlessly as possible. As a result, DIVA PIM chips are designed to resemble commercial DRAMs, enabling PIM devices to be accessed by host software as if it were conventional memory. In Figure 1, we show a small set of PIMs connected to a single host processor through conventional memory control logic.

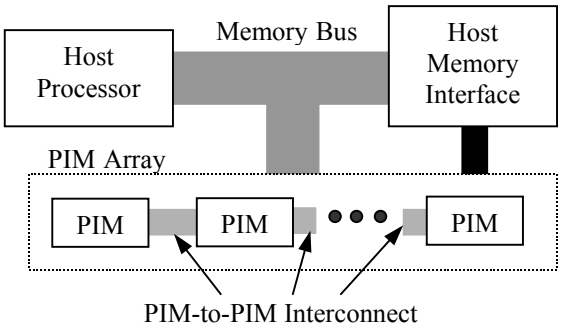

Figure 1. DIVA system architecture

Figure 2 shows the major control and data connections of each DIVA PIM node. The DIVA PIM node processing logic supports single-issue, in-order execution, with 32-bit instructions and 32-bit addresses. There are two datapaths whose actions are coordinated by a single execution control unit: a 32-b scalar datapath and a 256-b WideWord datapath. The WideWord datapath performs integer logic and arithmetic operations on groupings of 8-, 16-, or 32-b operands or single-precision floating-point operations on groupings of 32-b operands. Both datapaths execute from a single instruction stream fetched from a small instruction cache. Each datapath has its own independent general-purpose register file with 32 registers. For a more detailed description of the DIVA microarchitecture, refer to [3,4].

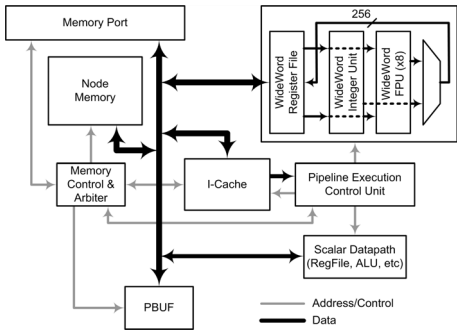

Figure 2. DIVA PIM node organization

3. Chip Description

As discussed earlier, the most recently developed PIM2 chip greatly improves on an earlier PIM prototype as it incorporates a DDR SDRAM interface, WideWord floating-point capability, and address translation. A brief description of the design and integration of each of these components is given in this section, followed by some chip implementation details.

3.1 Subcomponent Design and Integration

For the PIM device to reside in the memory system of commodity computers, it must use a compatible memory interface. A standard DDR SDRAM interface was designed and implemented for this PIM chip. It supports almost all modes allowed by the standard, such as sequential and interleaved accesses, burst mode lengths, etc. It acknowledges all transactions allowed by the standard, simply ignoring those transactions which require no internal action in the PIM chip. To maintain backward compatibility with older systems, a simple SDRAM interface was also maintained, and the PIM chip contains a 1-bit input pin so the system can select which memory interface to make operational in the resident system. This capability proved useful in chip check-out as our testing facility could not support DDR triggering of input stimuli, so the older SDRAM interface was used for checkout of all internal components, with the DDR interface verification taking place in-system.

A major enhancement to the PIM chip architecture was the inclusion of eight single-precision floating-point units in the WideWord datapath, as shown in Figure 2. The integration of this FPU [10,11] into the WideWord pipeline presented several challenges. The FPU is a 5-stage pipeline, with most operations requiring 5 cycles. This FPU was merged into the execute stage of the WideWord pipeline, which is itself a 5-stage pipeline based on the DLX design [6] where integer operations require 1 cycle in the execute stage of the pipeline. A fairly complex pipeline controller was designed and implemented to handle instruction issue/retirement sequencing and data hazards associated with FPU operation. For example, since the FPU itself is a 5-stage pipeline, if a source register specifier of an FPU instruction matches the destination register specifier of a previous FPU instruction that is still in progress in the FPU pipeline, the instruction issue logic must hold up the instruction until the result is ready for forwarding.

Another enhancement was the incorporation of an address translation unit that provides a virtual to physical address mapping for both instructions and data. A description of this unit is found in [2] with other PIM memory management issues described in [5]. The integration of this unit required sequencing with instruction cache look-ups to minimize the total time required for instruction fetches. Address translation for data loads/stores was not as problematic, as there was ample timing margin in the memory request cycle to allow address translation.

3.2 Chip Implementation

The subcomponents mentioned above were specified in RTL VHDL, with some structural VHDL for timing-critical blocks. These subcomponents as well as the integrated PIM chip were synthesized using Synopsys Design Compiler and Automatic Chip Synthesis tools. Floorplanning, clock-tree routing, and place-and-route were performed with Cadence First Encounter tools. Design checks were performed using Mentor Calibre.

The resulting SRAM-based PIM chip is shown in Figure 3 and implements all features of the PIM architecture concept described above. It was fabricated through MOSIS in TSMC 0.18µm technology, and the silicon die measures 10.5mm X 11.5mm. The chip contains approximately 56.6 million transistors and 420 pads, and it is packaged in a 35mm TBGA.

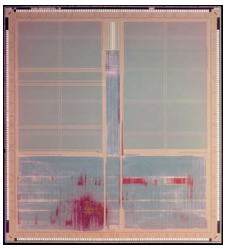

Figure 3. Die Microphotograph of PIM2 Chip

4. System Integration

Inserting the PIM chip in the HP zx6000 workstation required integration on many levels including: hardware, firmware, operating system and applications.

A printed circuit board was fabricated to match JEDEC MO-206 and No. 21-C for PC2100 DDR DIMM Design Standard [9] as shown in Figure 4. Stacking connectors are used for the PIM network to allow communication between boards with a minimum number of cables.

Figure 4. Two PIM DIMM.

At boot-time, firmware attempts to optimize the memory system performance using address line and bank interleaving which runs counter to a coherent view of the PIM memory space on the host/application side. The Serial Presence Detect (SPD) ROM on the DIMM is not populated, and the memory controller settings are modified between firmware and operating system boot to place the PIMs in the memory system without interleave.

The chip was implemented without Error Correction Code (ECC) bits on the data lines so ECC detection and correction throughout the machine was disabled in the memory controller. The common memory system feature Chipkill is enabled by data bit spreading. This requires the host to rearrange data in the PIM memory system to obtain proper alignment with the PIM processor. Future memory systems must be tracked and influenced to achieve a more PIM-friendly memory organization.

Applications interact with the PIM via a standard Linux kernel module loaded into the operating system. The Itanium2 used in the HP zx6000 has a three-level cache that cannot be disabled with current firmware. This issue is mitigated by declaring the PIM memory non-cacheable in the kernel module; which forces reads and writes out to the PIM memory system. Additional tools include: a gcc compiler backend that supports the instruction set architecture, a parallelizing front-end for the C compiler based on SUIF[12] and user monitor code providing code loading, word reads/writes and controlling PIM run/halt state.

The end result of the system integration is up to eight PIMs controllable via standard programming languages running under Linux in the memory system of an HP zx6000 workstation as shown in Figure 5.

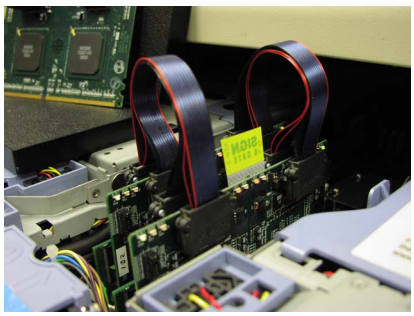

Figure 5. PIM DIMMs in HP zx6000 memory bay.

5. Results

The results generated thus far are both interesting and promising. The HPC Challenge [7] benchmarks provide one startpoint for the work while stressmarks from the DIVA project provide another.

Several benchmarks have been demonstrated and produce correct results. NAS Conjugate Gradient (cg) was the first stressmark to take advantage of the eight parallel floating-point units. Transitive Closure also from the DIVA project was ported to the system. In this stressmark the dataset is partitioned in as many pieces as processing elements and they exchange rows performing computation to determine the minimal transitive matrix. The two PIM Transitive Closure code also served as a first proof of communication. Two kernels of artificial intelligence code, a mutual information algorithm and a graph-clustering algorithm have also been ported and shown to produce the correct result. Instrumented timing measurements are now in progress

Streamadd, a subset of STREAM from the HPC benchmarks, provided two families of interesting results. It is a very simple subroutine to test sustained memory bandwidth and is summed up as: For i = 1 to Large {c[i] = a[i] + b[i]}. First the integer implementation of streamadd on the PIM produces a monotonic increase in execution time as the array sizes increase to the right on the constant (pim) line in Figure 6.

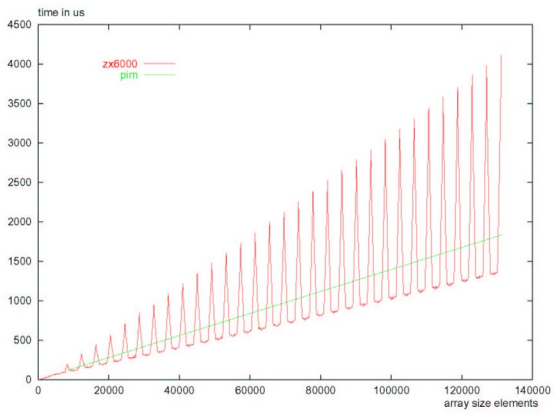

Figure 6. Integer Streamadd IT2 & PIM

The Itanium2 on the other hand shows a regular repeating pattern of execution time spikes that grow up to 3x the rising baseline execution time (zx6000).

The wide variation in execution time over extended problem size were found to be Level 1 Data Translation Lookaside Buffer (L1DTLB) misses forcing conflicting transfer attempts from Level 2 (which quickly fill the L2 cache transaction queue L2_OzQ) caused by data array offsets [8]. The PIM, with cache-speed access to all data without a data cache, suffered no such effects.

The second case is a floating-point implementation of the same streamadd operation. The PIM results are also as expected, deterministic performance monotonically increasing as the problem size increases. The Itanium2 floating point result shows much less variation than the integer result since floating-point values bypass the L1 cache and go straight through the L2 and L3 to main memory. As the array sizes increase their offsets always conflict somewhat and the single PIM execution time (pim) is comparable to Itanium2 execution time (zx6000) as shown in Figure 7.

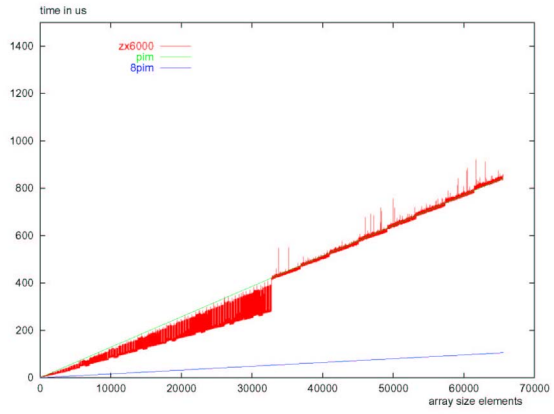

Figure 7. Floating Point Streamadd IT2 & PIM

With eight PIMs running in the memory system there is an 8x speedup over the single Itanium2 processor (8pim). This result is significant on several counts. First, the PIM processor is running at 140 MHz while the Itanium2 is running at 900MHz yielding a nearly 6.5 speedup in flops/MHz for a single PIM. Second, the Itanium2 is burning a maximum of 130 watts while the PIM is running at about one watt. So a single PIM has an over 100x power consumption advantage. In addition, the past several years have seen the advent of cycle skipping and frequency throttling in high-performance processors to keep the temperature within safe limits. If such trends continue PIMs will provide even further performance improvements.

Random Access, another HPC Benchmark [7] that measures memory bandwidth, was implemented for PIMs in the zx6000. The result was encouraging in that a single PIM was measured and matches the Itanium2 at 0.0044 GUPs including interface overhead, while eight PIMs in parallel filling the memory bus width would yield 0.036 GUPs. This measurement is remarkable as it is on par with a 512 node Cray T3E, which is noted for good performance on such benchmarks [7].

6. Conclusion and Future Work

We are confident that PIM is of great use in the battle for better processor–memory bandwidth via lower latency and for much greater power efficiency. Results show a 6.5 speedup over Itanium2 in flops/MHz on a single PIM and more than 100x power consumption advantage. Our PIM produces 1.12 Gflops/watt while the Itanium2 produces 0.036 Gflops/watt giving the PIM a 30x advantage in peak performance per watt. Additionally, without the cache hierarchy it is much more straightforward to approach peak performance with real-world sustained performance. Both the bandwidth gap and power dissipation issues continue to grow with no change in sight. With appropriate influence on memory space organization PIMs can be used to great benefit.

Going forward, in addition to porting other HPC Challenge benchmarks, we are collaborating with our artificial intelligence group implementing a Link Discovery application. Its primary concern is to identify strong links and discover hidden relationships among entities and organizations based on low-level, incomplete and noisy evidence data [1]. We are also exploring the use of PIM for emulating associative memories to support cognitive information processing.

Acknowledgments

This research was supported in part by DARPA contracts F30602-98-2-0180 and F33615-03-C-4105. Application software assistance was provided by Spundun Bhatt, Jacqueline Chame and Mary Hall.

References:

[1] J. Adibi, H. Chalupsky, A. Valente, and E. Melz. The kojak group finder: Connecting the dots via integrated knowledge-based and statistical reasoning. *In Innovative Applications of Artificial Intelligence IAAI 2004*. American Association for Artificial Intelligence, 2004

[2] Herming Chiueh, Jeffrey Draper, Sumit Mediratta, Jeff Sondeen, "The Address Translation Unit of the Data-Intensive Architecture (DIVA) System", *Proceedings of the 28th European Solid-State Circuit Conference* , September 2002

[3] Jeff Draper, et al, "The Architecture of the DIVA Processing-In-Memory Chip", in *Proc. of the International Conference on Supercomputing*, June 2002

[4] Jeffrey Draper, Jeff Sondeen, Chang Woo Kang, "Implementation of a 256-bit WideWord Processor for the Data-Intensive Architecture (DIVA) Processing-In-Memory (PIM) Chip", in *Proc. of the 28th European Solid-State Circuit Conference*, Sep. 2002

[5] M. Hall and C. Steele. "Memory Management in a PIM-Based Architecture," in *Proc. of the Workshop on Intelligent Memory Systems*, October 2000

[6] J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufman, Second Edition, 1996

[7] HPC Challenge Benchmarks, http://icl.cs.utk.edu/hpcc/index.html

[8] Intel Developer Documentation, http://www.developer.intel.com

[9] JEDEC, http://www.jedec.org

[10] Taek-Jun Kwon, Joong-Seok Moon, Jeff Sondeen, Jeff Draper, "An 0.18µm Implementation of a Floating-Point Unit for a Processing-In-Memory System", in *Proc. of the International Symposium on Circuits and Systems*, May 2004

[11] Joong-Seok Moon, Taek-Jun Kwon, Jeff Sondeen, Jeff Draper, "An Area-Efficient Standard-Cell Floating-Point Unit Design for a Processing-In-Memory System", in *Proc. of the 29th European Solid-State Circuit Conference*, Sep. 2003

[12] SUIF Compiler System, http://suif.stanford.edu