Implementation of a 256-bit WideWord Processor for the Data-Intensive Architecture (DIVA) Processing-In-Memory (PIM) Chip

Jeffrey Draper, Jeff Sondeen
USC Information Sciences Institute
draper@isi.edu, sondeen@isi.edu

Chang Woo Kang
University of Southern California
ckang@usc.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, including multimedia, pointer-based, and sparse-matrix applications. The DIVA project is building a prototype workstation-class system using PIM chips in place of standard DRAMs to demonstrate these concepts.

A key component of this architecture is the WideWord Processor, which is a 5-stage pipelined 256-bit datapath, complete with register file and ALU blocks. This component offers fine-grained data parallelism resulting in significant speedups. This paper details the design and implementation of this WideWord Processor in TSMC 0.18μm technology.

1. Introduction

The increasing gap between processor and memory speeds is a well-known problem in computer architecture, with peak processor performance increasing at a rate of 50-60% per year while memory access times improve at merely 5-7%. Furthermore, techniques designed to hide memory latency, such as multithreading and prefetching, actually increase the memory bandwidth requirements [3]. A recent VLSI technology trend, embedded DRAM, offers a promising solution to bridging the processor-memory gap [9]. One application of this technology integrates logic with high-density memory in a processing-in-memory (PIM) chip. Because PIM internal processors can be directly connected to the memory banks, the memory bandwidth is dramatically increased (with hundreds of gigabit/second aggregate bandwidth available on a chip--up to 2 orders of magnitude over conventional DRAM). Latency to on-chip logic is also reduced, down to as little as one half that of a conventional memory system, because internal memory accesses avoid the delays associated with communicating off chip.

The Data-Intensive Architecture (DIVA) project uses PIM technology to replace or augment the memory system of a conventional workstation with "smart memories" capable of very large amounts of processing. System bandwidth limitations are thus overcome in three ways: (1) tight coupling of a single PIM processor with an on-chip memory bank; (2) distributing multiple processor-memory "nodes" per PIM chip; and, (3) utilizing a separate chip-to-chip interconnect, for direct communication between nodes on different chips that bypasses the host system bus. The system architecture of DIVA is focused on achieving the following four goals: (1) developing PIMs that can serve as the only memory in the system, assuming the dual roles of "smart memories" and conventional memory; (2) supporting a wide range of familiar programming paradigms, closely related to parallel computing; (3) targeting applications that are severely impacted by the processor-memory bottlenecks in conventional systems: sparse-matrix and pointer-based applications with irregular memory access patterns, and image and video applications with large working sets; and, (4) developing a VLSI device to exploit memory and communications bandwidth in PIM-based systems while making efficient use of on-chip resources for target applications.

This paper focuses on the microarchitecture design and implementation of the WideWord Processor component of the PIM processing logic. Similar in style to vector extensions like AltiVec [1], the DIVA WideWord Processor uses a 256-bit datapath that enables significant processing speedups through the use of data parallelism. The WideWord Processor was fabricated as part of a DIVA prototype chip in TSMC 0.18μm technology and is currently in test. The remainder of the paper is organized as follows. Sections 2 and 3 present an overview of the DIVA system architecture and microarchitecture, to put the WideWord Processor design into its proper context. Section 4 describes the WideWord microarchitecture in detail. Section 5 presents details of the fabrication and testing of the WideWord Processor as part of a PIM chip, and Section 6 concludes the paper.

2. System 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 memory 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.

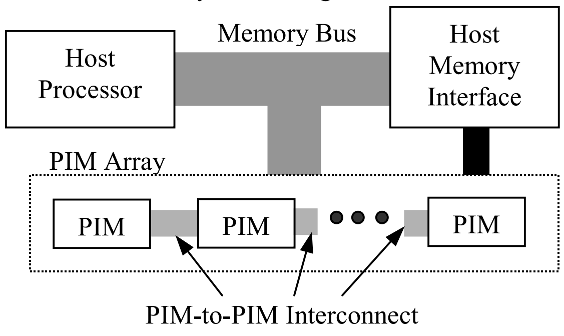

Figure 1. DIVA system architecture

Spawning computation, gathering results, synchronizing activity, or simply accessing non-local data is accomplished via parcels. A parcel is similar to an active message, as it is a relatively lightweight communication mechanism containing a reference to a function to be invoked when the parcel is received [12]. From a programmer's view, parcels, together with the global address space supported in DIVA, provide a compromise between the ease of programming a shared-memory system and the architectural simplicity of pure message passing. Parcels utilize a separate PIM-to-PIM interconnect to enable communication without interfering with host-memory traffic, as shown in Figure 1. Details of this interconnect can be found in [10], and more detail about the DIVA system architecture can be found in [2][4][6][7].

3. Microarchitecture overview

Each DIVA PIM chip is a VLSI memory device augmented with general-purpose computing and networking/communication hardware. Although a PIM may consist of multiple nodes, each of which are primarily comprised of a few megabytes of memory and a node processor, Figure 2 shows a PIM with a single node, which reflects the focus of the initial research that is being conducted. Nodes on a PIM chip share a single PIM Routing Component (PiRC) and a host interface. The PiRC is responsible for routing parcels on and off chip. The host interface implements the JEDEC standard SDRAM protocol so that memory accesses as well as parcel activity initiated by the host appear as conventional memory accesses from the host perspective.

Figure 2 also shows two interconnects that span a PIM chip for information flow between nodes, the host interface, and the PiRC. Each interconnect is distinguished by the type of information it carries. The PIM memory bus is used for conventional memory accesses from the host processor. The parcel interconnect allows parcels to transit between the host interface, the nodes, and the PiRC. Within the host interface, a parcel buffer (PBUF) is a buffer that is memory-mapped into the host processor's address space, permitting application-level communication through parcels. Each PIM node also has a PBUF, memory-mapped into the node's local address space.

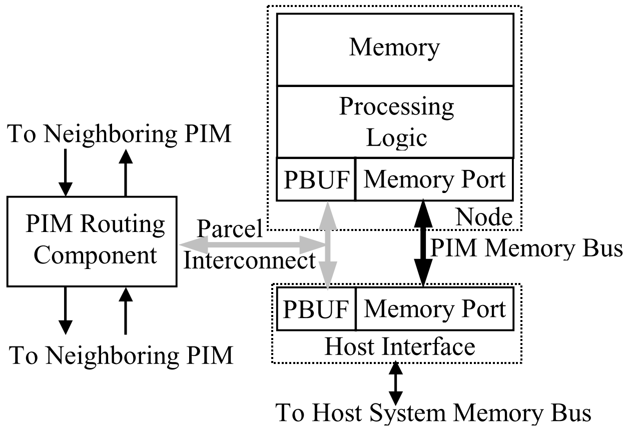

Figure 2. DIVA PIM chip organization

Figure 3 shows the major control and data connections within a node, with the 256-bit memory data bus as the centerpiece. 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 scalar datapath that performs sequential operations on 32-bit operands, and a WideWord datapath that performs fine-grain parallel operations on 256-bit operands. Both datapaths execute from a single instruction stream under the control of a single 5-stage DLX-like pipeline [8]. The instruction set has been designed so both datapaths can, for the most part, use the same opcodes and condition codes, generating a large functional overlap.

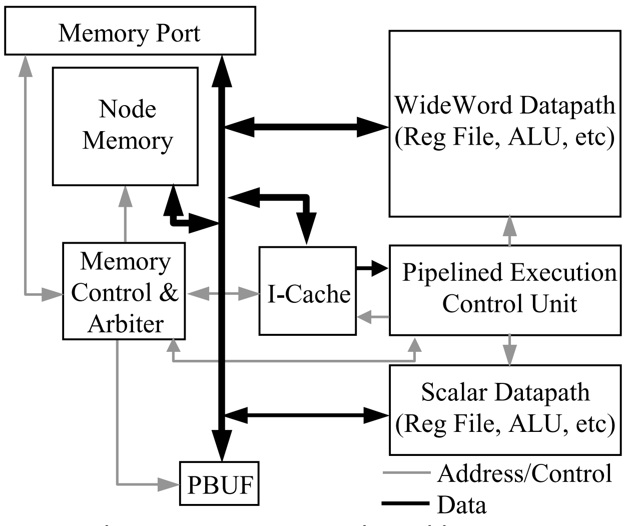
Memory Port
Node Memory
WideWord Datapath (Reg File, ALU, etc)
Memory Control & Arbiter
I-Cache
Pipelined Execution Control Unit
Scalar Datapath (Reg File, ALU, etc)
PBUF
Address/Control
Data

Figure 3. DIVA PIM node architecture

Each datapath has its own independent general-purpose register file, 32 32-bit registers for the scalar datapath and 32 256-bit registers for the WideWord datapath, but special instructions permit direct transfers between datapaths without going through memory. Although not supported in the initial DIVA prototype, floating-point extensions to the WideWord datapath will be provided in future implementations. In addition to the execution units, each DIVA PIM node contains other essential components of note. These components are described in [5].

4. Microarchitecture details of the DIVA WideWord Processor

The combination of the execution control unit and WideWord datapath is regarded as the WideWord Processor. This component enables superword-level parallelism [11] on wide words of 256 bits, similar to multimedia extensions such as MMX and AltiVec. This fine-grain parallelism offers additional opportunity for exploiting the increased processor-memory bandwidth available in a PIM. Selective execution, direct transfers to/from other register files, integration with communication, as well as the ability to access main memory at very low latency, distinguish the DIVA WideWord capabilities from MMX and AltiVec. This section details the microarchitecture of this component by first presenting an overview of the instruction set architecture, followed by a description of the pipeline.

4.1. Instruction set architecture

opcode	rD	rA	rB	C	P	W	funct
6	5	5	5	1	2	2	6

Field Bit Widths
(32 bits total)

Figure 4. WideWord instruction format

As shown in Figure 4, most DIVA WideWord instructions use a three-operand format to specify two 256-bit source registers and a 256-bit destination register. The opcode generally denotes a class of operations, such as arithmetic, and the function denotes a specific operation, such as add or subtract. The **C** bit indicates whether the operation performed by the instruction execution updates condition codes. The **W** field indicates the operand width, allowing WideWord data to be treated as a packed array of objects of eight, sixteen, or thirty-two bits in size. This characteristic means the WideWord ALU can be represented as a number of variable-width parallel ALUs. The **P** field indicates the participation mode, a form of selective subfield execution that depends on the state of local and neighboring condition codes. Under selective execution, only the results corresponding to the subfields that participate in the computation are written back, or committed, to the instruction's destination register. The subfields that participate in the conditional execution of a given instruction are derived from the condition codes or a mask register, plus the instruction's 2-bit participation field. For more details, see [2].

The WideWord instruction set consists of roughly 30 instructions implementing typical arithmetic instructions like add, subtract, and multiply; logical functions like AND, OR, NOT, XOR; and logical/arithmetic shift operations. In addition, there are load/store and transfer instructions that provide for rich interactions between the scalar and WideWord datapaths.

Some special instructions include permutation, merge, and pack/unpack. The WideWord permutation network supports fast alignment and reorganization of data in wide registers. The permutation network enables any 8-bit data field of the source register to be moved into any 8-bit data field of the destination register. A permutation is specified by a permutation vector, which contains 32 indices corresponding to the 32 8-bit subfields of a WideWord destination register. A WideWord permutation instruction selects a permutation vector by either specifying an index into a small set of hard-wired commonly used permutations or a WideWord register whose contents are the desired permutation vector. The merge instruction allows a WideWord destination to be constructed from the intermixing of subfields from two source operands, where the source for each destination subfield is selected by a condition specified in the instruction. This merge instruction effects efficient sorting. The pack/unpack instructions allow the truncation/elevation of data types and are especially useful in pixel processing.

4.2. Pipeline description

The WideWord Processor pipeline is a standard DLX-like 5-stage pipeline, with the following stages: (1) instruction fetch; (2) decode and register read; (3) execute; (4) memory; and, (5) writeback. Data hazards occur when there are read-after-write register dependences between instructions that co-exist in the pipeline. The controller and datapath contain the necessary forwarding, or bypass, logic to allow pipeline execution to proceed without stalling in most data dependence cases. Register forwarding is complicated somewhat by the participation capability. Participation status must be forwarded along with each subfield to effect correct forwarding.

5. Implementation and testing of the DIVA WideWord Processor

The DIVA WideWord Processor specification required on the order of 25,000 lines of VHDL code, consisting of a mix of RTL-level behavioral and gate-level structural code. A preliminary, unoptimized stand-alone layout of the WideWord Processor used 100,000 standard cells (approximately one million transistors) and occupied 10 sq mm in 0.18μm technology, projected to operate at 300MHz while dissipating 500mW.

Although the WideWord Processor is suitable for stand-alone implementations, the DIVA project employs it as part of a tightly integrated node design, as discussed in Section 3. The WideWord Processor VHDL specification was included as part of a DIVA PIM prototype specification, which was synthesized using Synopsys Design Analyzer. The entire chip was placed and routed with Cadence Silicon Ensemble, and physical verification, such as DRC and LVS, was performed with Mentor Calibre. The intellectual property building blocks used in the chip include Virage Logic SRAM, a NurLogic PLL clock multiplier, and Artisan standard cells, pads, and register files.

The first DIVA PIM prototype, shown in Figure 5, is a single-node implementation of the DIVA PIM chip architecture. Due to challenges in gaining access to embedded DRAM fabrication lines, this first prototype is SRAM-based. This chip implements all features of the DIVA PIM architecture except address translation and floating-point capabilities. A second version of a PIM chip, which not only integrates these functions but achieves a faster clock rate, is due to tape out in the second half of 2002. The chip shown in Figure 5 was fabricated through MOSIS in TSMC 0.18μm technology, and the silicon die measures 9.8mm on a side. It contains approximately 2 million logic transistors in addition to the 53 million transistors that implement 8 Mbits of SRAM. The chip also contains 352 pads, of which 240 are signal I/O, and is packaged in a 35mm TBGA.

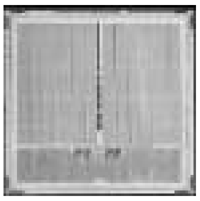

Figure 5. DIVA PIM prototype chip

The chip is being tested with the use of an HP 16702A logic analysis system. Pattern generator modules are utilized to apply test vectors to the inputs of the chip, and timing/state capture modules are used to sense the outputs of the chip. The chip is currently being tested for functionality at a testbench speed of 80MHz. Although exhaustive testing has not yet been completed, the chip is running a demonstration application of matrix transpose that exercises all major control and datapaths within the scalar processor, including the permutation network highlighted in Section 4.1. Even in this limited test setup, the chip is achieving 640MOPS and 2.56Gbytes/s memory bandwidth while dissipating only 800mW. We anticipate even greater achievements with further testing.

6. Conclusion

This paper has presented the design and implementation of the WideWord Processor used in the DIVA system, an integrated hardware and software architecture for exploiting the bandwidth of PIM-based systems. A working implementation of this design, based on TSMC 0.18μm technology, has proven the validity of the design. The workstation system that is currently being developed to use this component is projected to achieve speedups ranging from 8.8 to 38.3 over conventional workstations for a number of applications. These improvements arise mainly from three sources: decreased memory times; coarse-grain parallelism across PIMs to exploit system bandwidth; and, wide on-chip datapaths to exploit fine-grain parallelism, including especially those wide datapaths within the WideWord Processor.

Acknowledgments

This research was supported by DARPA contract F30602-98-2-0180.

References

[1] http://www.altivec.org

[2] J. Brockman, et al, "Microservers: A New Memory Semantics for Massively Parallel Computing", in *Proc. of the International Conference on Supercomputing*, June 1999, pp. 454 - 463.

[3] D. Burger, J. Goodman and A. Kagi. "Memory Bandwidth Limitations of Future Microprocessors," In *Proc. of the 23rd International Symposium on Computer Architecture*, May 1996.

[4] J. Chame, M. Hall and J. Shin, "Code Transformations for Exploiting Bandwidth in PIM-Based Systems," In *Proc. of the Workshop on Solving the Memory Wall Problem*, ISCA Workshop, June 2000.

[5] J. Draper, et al, "The Architecture of the DIVA Processing-in-Memory Chip", to appear at International Conference on Supercomputing, June 2002.

[6] M. Hall, et al, "Mapping Irregular Applications to DIVA: A Data-Intensive Architecture," In *Proc. of SC '99*, November 1999.

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

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

[9] S. Iyer and H. Kalter, "Embedded DRAM Technology: Opportunities and Challenges," *IEEE Spectrum*, April 1999, pp. 56 - 64.

[10] C. Kang, J. Draper, "A Fast, Simple Router for the Data-Intensive Architecture (DIVA) System," In *Proc. of the IEEE Midwest Symposium on Circuits and Systems*, August 2000.

[11] R. Lee, "Subword Parallelism with MAX-2," *IEEE Micro* 16(4), Aug. 1996, pp. 51 - 59.

[12] T. von Eicken, D. Culler, S. C. Goldstein, and K. Schauser, "Active Messages: a Mechanism for Integrated Communication and Computation", In *Proc. of the 19th International Symposium on Computer Architecture*, May 1992.