

# A 0.18 $\mu$ m IMPLEMENTATION OF A FLOATING-POINT UNIT FOR A PROCESSING-IN-MEMORY SYSTEM

Taek-Jun Kwon, Joong-Seok Moon\*, Jeff Sondeen, Jeff Draper

USC Information Sciences Institute  
4676 Admiralty Way, Marina del Rey, CA 90292 U.S.A.  
{tjkwon,sondeen,draper}@ISI.EDU

\*Apple Computer Inc.  
1 Infinite Loop, Cupertino, CA 95014 U.S.A.  
jsmoon@apple.com

## 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 key capability of this architecture is the support of parallel single-precision floating-point operations. Each PIM chip includes eight single-precision FPUs, each of which supports eight basic instructions and IEEE-754 compliant rounding and exceptions. Through block sharing and a hardware-efficient division algorithm, the resulting FPU is well-balanced between area and performance. This paper focuses on the novel divide algorithm implemented and documents the fabrication and testing of a prototype FPU based on standard cell methodology in TSMC 0.18 $\mu$ m CMOS technology.*

## 1. INTRODUCTION

The Data-Intensive Architecture (DIVA) project [1][2] is using embedded memory technology to replace the memory system of a conventional workstation with “smart memories” capable of very large amounts of processing. 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 processing-in-memory (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 not aided by caches in conventional systems due to little spatial or temporal data locality and are thus severely impacted by the processor-memory bottleneck. Based on our first PIM implementation, a PIM system incorporating these devices is projected to achieve speedups ranging from 8.8 to 38.3

over conventional workstations for a number of applications [2].

An important target application of DIVA is multimedia. Multimedia applications perform repeated computations on streams of data, often with little temporal data reuse. As processors exploit increased parallelism, multimedia applications become memory bound. Since many of these applications operate on floating-point data, efficient floating-point computation is crucial for the success of DIVA systems. The DIVA FPU supports the single-precision data type, four rounding modes, and most precise exceptions specified by the IEEE-754 standard. Eight FPUs are integrated in each PIM chip; therefore, a significant design tradeoff effort was conducted to minimize area and maximize performance of the FPU. In particular, the division algorithm is carefully chosen and implemented to minimize the area overhead while achieving high throughput. An earlier paper reported the overall design of the DIVA FPU and provided pre-fabrication estimates of its performance [6]. This paper focuses on details of the novel divide algorithm implemented and reports results of a prototype FPU fabricated using Artisan standard cells and ROM in TSMC 0.18 $\mu$ m CMOS technology. Lab measurements show the FPU is capable of running up to 266MHz with an average power dissipation of 154mW.

The remainder of this paper is organized as follows. Section 2 presents a brief description of the DIVA PIM node architecture followed by an overview of the FPU microarchitecture and detail of the divide algorithm in Section 3. Section 4 presents the implementation details of the prototype FPU. Section 5 presents the lab results followed by a brief summary and conclusion in Section 6.

## 2. DIVA ARCHITECTURE OVERVIEW

Figure 1 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

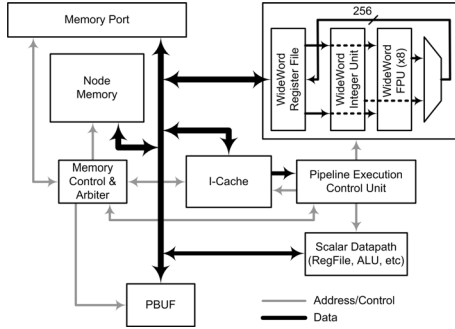


Figure 1: DIVA PIM node organization

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. A more detailed description of the DIVA hardware organization can be found in the literature [3][4].

### 3. DIVA FPU MICROARCHITECTURE

The DIVA FPU implements a subset of the IEEE-754 floating-point standard [5]. Since target applications are mostly from the multimedia realm, only single-precision numbers are supported. To achieve a better area-performance solution, operations on denormalized numbers are not supported, and such operations cause exceptions when attempted. In addition, whenever a result is a denormalized number, an underflow exception is raised and the minimum normalized number is produced for output. The inexact exception flag on division operations is not IEEE-754 compliant, which is common for multiplicative division algorithms. Additional operations are necessary to correct this. Other exception flags – Invalid, Divide by Zero, Overflow, Underflow and

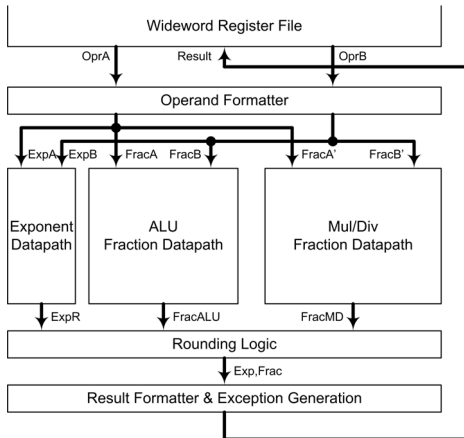


Figure 2: DIVA FPU microarchitecture

Inexact (except divide) – are accurately generated as specified by the IEEE-754 standard. All four rounding modes are implemented. Figure 2 depicts the microarchitecture of the FPU. The FPU has two main blocks: ALU and Mul/Div. Exponent computation functions for both blocks are combined in one datapath to reduce area. Similarly, logic for converting to/from the internal number format and rounding logic are shared for both of the datapaths. Since DIVA execution control is a simple in-order single-issue instruction pipeline, combining common datapaths does not suffer any performance penalty. Input registers for the ALU and the Mul/Div blocks are controlled by separate enable signals so that only one of the datapaths is active for each instruction. Six operations (Add, Subtract, Fp2Int, Int2Fp, Absolute, Negate) are executed by the ALU block while Multiply and Divide operations are executed by the Mul/Div block. For a detailed description of the ALU block and instruction pipelining issues, refer to [6].

#### 3.1. Multiplier/Divider Fused Unit

The block diagram of the Mul/Div block is shown in Figure 3. As described in the previous section, the exponent computation datapath, the operand/result formatter and rounding logic are shared with the ALU block. The multiplier for the fraction datapath is shared between multiply and divide operations to reduce the area. A two-stage pipelined multiplier is used for better synthesis results and stage balance with the ALU block. Since the multiply function implementation is a straightforward 5-stage pipelined design, we focus the rest of this section on the novel divider implementation.

To meet performance requirements of modern scientific applications such as 3D graphics rendering, high

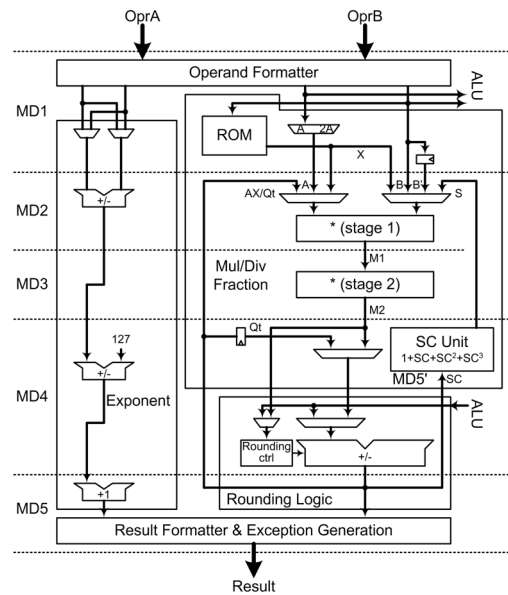


Figure 3: Block diagram of the Mul/Div datapath

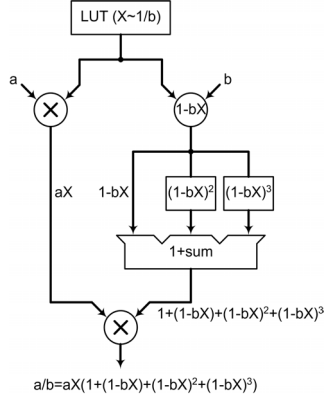


Figure 4: Division algorithm by Liddicoat and Flynn

performance is crucial for division as well as multiplication. High-radix SRT dividers based on the digit recurrence algorithm are widely used for modern microprocessors. However, this type of divider requires a hardware block, which will substantially increase the area for datapath and communication buses. Since eight copies of the DIVA FPU are to be implemented, a good area-performance solution is the primary design goal. To achieve this, we adapted the multiplicative division algorithm proposed by Liddicoat and Flynn [7][8], which is based on Taylor series expansion, as shown in Figure 4. This algorithm achieves fast computation by using parallel powering units such as squaring and cubing units, which compute the higher-order terms significantly faster than traditional multipliers with a relatively small hardware overhead. An example of squaring a 4-bit number is shown in Figure 5. Partial product terms can be combined using the equivalence  $A_i A_j + A_j A_i = 2A_i A_j$  and  $2A_i A_j$  is represented by placing  $A_i A_j$  one column to the left. The square of operand A can be computed with the reduced partial product array shown in the lower portion of Figure 5. The cubing unit can be designed in a similar fashion. Further area and latency optimization can be achieved by truncating the least significant bits of these partial product terms while satisfying the required precision of floating-point operation. Specifically, 69.6% of the partial product terms from the squaring unit and 97.6% of the partial product terms from the cubing unit were truncated. As a

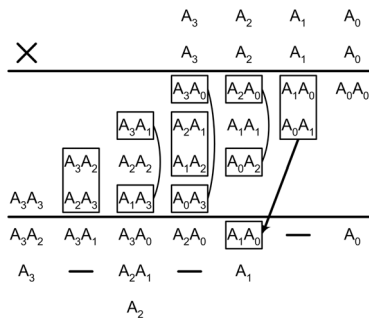


Figure 5: PPA reduction for squaring unit

result, the proposed squaring unit achieves an area reduction of 66.8%, and the proposed cubing unit achieves an area reduction of 89.9%, as compared with a traditional multiplier. There are three major multiply operations to produce a quotient with 0.5 ulp (unit in the last place) error as shown in Figure 4. One additional multiply operation is required for exact rounding. To maximize the area efficiency, all of these multiply operations are executed by one multiplier. By sharing the multiplier, the pipeline latency increases, however, through careful pipeline scheduling, we were able to achieve high throughput for consecutive divide instructions (5 clock cycles). A lookup table for an initial seed value, an approximate value of the reciprocal of the divisor, is implemented using a 128x7-b ROM. Table I summarizes the operations in each cycle for the divide instruction. For a detailed description of the algorithm, refer to [7].

TABLE I  
Steps for divide operation

Steps	Operation	Pipeline Stage
1	$X = \text{ROM}(b)$	MD1
2	$M1 = b * X$ (stage1)	MD2
3	$M2 = b * X$ (stage2), $M1 = a * X$ (stage1)	MD3/MD2
4	$SC = 1 - M2$ , $M2 = a * X$ (stage2)	MD4/MD3
5	$S = 1 + SC + SC^2 + SC^3$ , $AX = M2$	MD5/MD4
6	$M1 = AX * S$ (stage1)	MD2
7	$M2 = AX * S$ (stage2)	MD3
8	$Qt = \text{trunc}(M2) + 1$	MD4
9	$M1 = b * Qt$ (stage1)	MD2
10	$M2 = b * Qt$ (stage2)	MD3
11	$R = \text{round}(Qt)$	MD4
12	$\text{Result} = \text{format}(R)$	MD5

#### 4. IMPLEMENTATION

The FPU design has been described in Verilog with the exception of the two-stage multiplier, where the netlist was generated using Synopsys synthesis tools [9]. These netlists along with the ROM needed for the divider were combined together. For balanced pipeline stages, register retiming techniques have been generally applied in the logic synthesis step. This netlist was then placed and routed to generate a layout using Cadence Silicon Ensemble. The FPU was synthesized under the timing constraint of a 250MHz clock frequency. The layout of the prototype FPU occupies an area of 640μm x 640μm and contains 127,641 transistors. The post-layout result shows that the area of the proposed FPU is 28.6% smaller than the direct implementation of the division algorithm without sharing a multiplier. As the Mul/Div block occupies 54.4% of the total area of the FPU, the increased area for the divide instruction support such as a ROM and SC unit is only 31%. The area overhead to the overall PIM design is also very small as 8 FPUs occupy only 3.6% of the total area of the existing DIVA PIM chip. Since FPUs are implemented in various ways, a detailed comparison with other designs is difficult. The straightforward

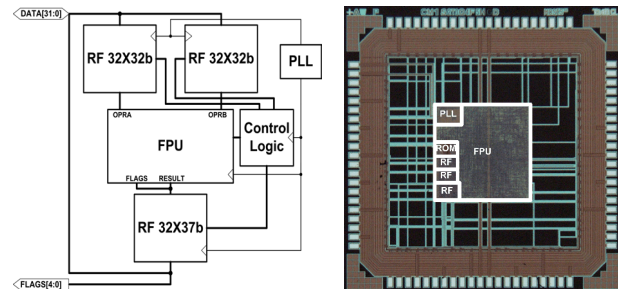


Figure 6: DIVA FPU test chip

standard cell approach described in this paper achieved a portable FPU design in a moderate area with relatively high throughput as compared to other implementations. For more comparison information, refer to [10].

A test chip was fabricated with the proposed FPU. The test chip consists of the FPU, two 32x32-b register files for the FPU inputs, one 32x37-b register file for the 32-b FPU output result and 5-b exception flags, a PLL and a control logic. The block diagram and the die photo of the test chip are presented in Figure 6. Features of the test chip are summarized in Table II.

TABLE II  
Summary of the test chip

Technology	TSMC 0.18 $\mu$ m CMOS
Supply Voltage	1.8V (core) 2.5V (pad)
Transistor count	168,149
Dimension	1062 $\mu$ m x 1062 $\mu$ m (core)
	3016 $\mu$ m x 3016 $\mu$ m (die)
Speed	266MHz @1.8V

## 5. EXPERIMENTAL RESULTS

### 5.1. Testing methodology

The chip has been tested with the use of an HP 16702A logic analysis system. Pattern generator modules were utilized to apply test vectors to the inputs of the chip, and timing/state capture modules were used to sense the outputs of the chip. To test the operation of the FPU at high clock speeds with a limited test setup, the following testing methodology was used. First, the inputs to the FPU were stored into two 32x32-b register files at a slow clock speed, and then the FPU operation on the same instruction was repeated on a random input data stream at the highest throughput rate (5 for divide and 1 for others) for each data point at maximum clock speed. The output results were stored into the 32x37-b register file at the same time and were then read out at the reduced I/O clock speed.

### 5.2. Power dissipation and performance

The test results show that the average power dissipation of the FPU is 154.4mW with a chip clock frequency of

266MHz at 1.8v. The average power dissipation for multiply and divide instructions is 15% more than other ALU instructions, mainly resulting from the larger datapaths of the Mul/Div block. The power dissipation for divide instructions is less than multiply instructions although the ROM is accessed only for divide instructions. This is because 40% of the pipeline stages are idle while consecutive divide instructions are processed. At the nominal supply voltage of 1.8V, the maximum operating clock frequency is 266MHz due to the limit of the PLL.

## 6. CONCLUSION

This paper presented an overview of the DIVA FPU design, detail of the divide algorithm implemented, and results of a prototype chip. A standard cell implementation based on TSMC 0.18 $\mu$ m CMOS technology has shown that an attractive area-performance result was achieved with moderate power dissipation. This balance was attained through an efficient divider implementation and sub-block sharing among different functions. This design has been implemented in a PIM chip currently under test. Preliminary results show the FPU to be fully functional.

## 7. ACKNOWLEDGEMENT

This research was supported by DARPA contract F33615-03-C-4105.

## REFERENCES

- [1] M. Hall and C. Steele, "Memory Management in PIM-based Systems", in *Proc. of the workshop on intelligent memory systems*, Boston, MA, 2000
- [2] Jeff Draper, et al, "The Architecture of the DIVA Processing-In-Memory Chip", in *Proc. of the International Conference on Supercomputing*, June 2002
- [3] 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
- [4] Jeffrey Draper, et al, "Implementation of a 32-bit RISC Processor for the Data-Intensive Architecture Processing-In-Memory Chip", in *Proc. of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors*, July 2002
- [5] "IEEE Standard for Binary Floating-Point Arithmetic", ANSI/IEEE Standard 754, Aug. 1985
- [6] 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
- [7] A. Liddicoat, "High-Performance Arithmetic for Division and the Elementary Function", Ph.D. dissertation, Stanford University, Feb. 2002
- [8] A. Liddicoat and M.J. Flynn, "High-Performance Floating-Point Divide", *Euromicro Symposium on Digital System Design*, Sep. 2001
- [9] *Module Compiler User Guide*, Synopsys Inc.
- [10] Soderquist, P., Leeser M., "Division and Square Root: choosing the right implementation", *Micro*, IEEE, pp.56-66, July-Aug. 1997