Design Trade-Offs in Floating-Point Unit Implementation for Embedded and Processing-In-Memory Systems

Taek-Jun Kwon, Jeff Sondeen, Jeff Draper
USC Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292 U.S.A.
{tjkwon,sondeen,draper}@ISI.EDU

Abstract—**Hardware support for floating-point (FP) arithmetic is a mandatory feature of modern microprocessor design. There are many alternatives in floating-point unit (FPU) design, and overall performance can be greatly affected by the organization of a floating-point unit. In this paper, design considerations and trade-off factors are evaluated for two types of floating-point unit architecture and implementation optimized under different design goals. The implementation results of the proposed FPUs based on standard cell methodology in TSMC 0.18μm technology exhibit that both designs are well optimized for their target applications. A single-instruction issue design is implemented in very small area; however, a design capable of concurrently executing FP add and multiply instructions is achievable with only a modest 24% area increase.**

I. INTRODUCTION

Due to the constant advances in VLSI technology and the prevalence of business, technical, and recreational applications that use floating-point operations, floating-point computational logic has long been an essential component of high-performance computer systems as well as embedded systems and mobile applications. Floating-point units (FPU) can be implemented in various ways, and the architecture of an FPU has a great affect on its overall performance, area, and power dissipation. This paper explores the trade-off space with respect to two FPU architectures that are optimized for different design goals. These two architectures were driven by the differing requirements of the Data-Intensive Architecture (DIVA) and Morphable Networked Micro-Architecture (MONARCH) projects. Although the cornerstone of both projects is a high-density VLSI device including FPU capability, the two projects differ in area and performance goals.

DIVA [1][2] uses embedded memory technology as processing-in-memory (PIM) to replace the memory system of a conventional workstation with "smart memories" capable of very large amounts of processing. DIVA 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]. Since DIVA PIM chips serve primarily as memory components, it is important to preserve a large majority of the die area for memory, so the processing logic for such PIM chips should be compacted as much as possible. Hence, the FPU architecture design for DIVA should occupy minimal area when implemented, perhaps even at the expense of performance and power.

MONARCH [3] targets real-time embedded applications that involve both high-speed signal processing and also data-dependent decision-oriented computing. At an architectural level, the MONARCH chip contains functional units that may serve as the central elements in a dataflow architecture for highly efficient stream computing or through morphing they may become the basis of vector extension units controlled by embedded threaded processors, such as a simple RISC design. In the latter mode, the configuration of the computational elements strongly resembles the WideWord operation of DIVA [4]. To achieve high-performance stream processing capability in MONARCH, FPU throughput should be maximized, even at the expense of area.

The remainder of the paper explores trade-offs in FPU design as motivated by the DIVA and the MONARCH projects. Section 2 presents the description of basic FPU blocks followed by a detailed description of two types of FPU architectures. Implementation details are presented in Section 3. Section 4 presents the simulation and comparison results followed by a brief summary and conclusion in Section 5.

0-7803-8834-8/05/$20.00 ©2005 IEEE.

II. DIVA FPU AND MONARCH FPU

A. Basic Blocks: ALU and Multiplier/Divider Fused Unit

Both the DIVA and the MONARCH FPUs implement a subset of the IEEE-754 floating-point standard [5]. Since target applications of both architectures are mostly from the multimedia realm, only single-precision numbers are supported. A multiplicative division algorithm is carefully chosen and implemented to minimize the area overhead while achieving high throughput. 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 Inexact (except divide) – are accurately generated as specified by the IEEE-754 standard. All four rounding modes are implemented.

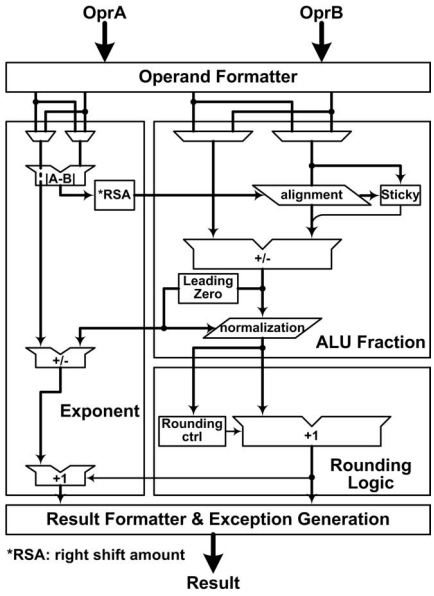

Figure 1. Block diagram of the ALU block

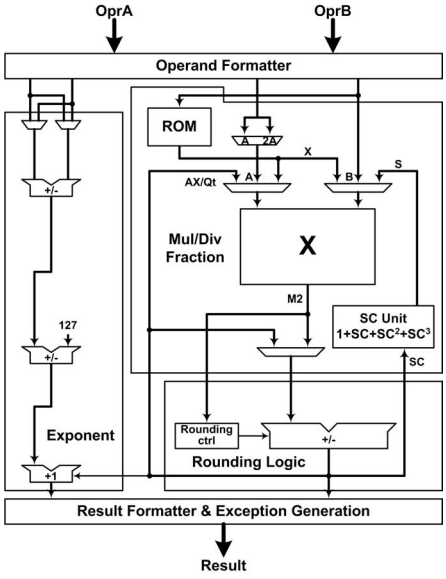

Figure 2. Block diagram of the Mul/Div block

Block diagrams of the ALU block and the Mul/Div block are shown in Fig. 1 and Fig. 2. Addition (which subsumes subtraction) and multiplication are not only the most frequently occurring floating-point arithmetic operations, but together they can support all other operations required by the IEEE 754 floating-point standard. We can regard all other functions, including division, as additions to or enhancements of these basic blocks. For these reasons, the implementation of addition and multiplication largely determines the overall performance of an FPU. 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.

To meet performance requirements of modern scientific applications such as 3D graphics rendering, high performance is crucial for division as well as multiplication. Since eight copies of an FPU are to be implemented in the case of DIVA, a good area-performance solution was one of the primary design goals. To achieve this, we adapted the multiplicative division algorithm proposed by Liddicoat and Flynn [6][7], which computes the quotient significantly faster than other division algorithms with a relatively small hardware overhead. The multiplier in the Mu/lDiv fraction datapath is shared between multiply and divide operations and several multiply operations in the division algorithm are executed by this multiplier to reduce area.

B. MONARCH FPU (Add-Multiply Configuration)

The organization of the MONARCH FPU is shown in Fig. 3. MONARCH targets real-time embedded dataflow applications that require highly efficient stream processing capability. Therefore, the overall FPU architecture should be optimized to achieve higher performance, such as low latency and high throughput. To maximize the throughput of floating-point operations, a high issue rate is also an essential

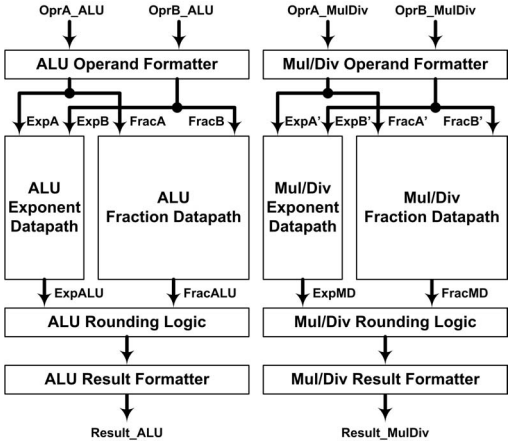

Figure 3. MONARCH FPU organization

point of design consideration, especially the ability to concurrently execute FP add and multiply operations.

To achieve such requirements, we adapted the Add-Multiply configuration, which is a commonly used FPU architecture found in most modern microprocessors. This configuration consists of separate ALU and MulDiv blocks, and there is no common datapath component shared between these two blocks. Both blocks have separate inputs and outputs and they operate independently. Therefore, a high throughput can be achieved by issuing up to two floating-point instructions at the same clock cycle, assuming one instruction is an ALU type and the other is a Mul/Div type. Since both basic FPU blocks do not share any component, each block can be optimized separately to have a reduced number of stages to achieve low instruction latency. As a result, the ALU block has a 3-stage pipelined architecture, and the MulDiv block has a 4-stage pipelined architecture. The latency of all ALU operations and multiply operation is 3 clock cycles while the latency of division operation is 9 clock cycles. (Note that the actual implementation of the MONARCH FPU for the streaming processing component does not support division operations. The floating-point divider was included for the purpose of a fair comparison for this paper).

C. DIVA FPU (Fused Configuration)

Fig. 4 depicts the organization of the DIVA FPU. Since we need to preserve as much area as possible for memory in DIVA, several design considerations have been made. The 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 between both datapaths. DIVA execution control is a simple in-order single-issue instruction pipeline [4][8], therefore combining common datapaths does not suffer any performance penalty. The pipeline 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. A 2-stage pipelined fraction multiplier is used for better synthesis results and stage balance between the ALU and Mul/Div blocks. As a result, the proposed FPU for DIVA has a 5-stage pipelined architecture. The latency of all operations is 5 clock cycles except division, for which the latency is 12 clock cycles. For a more detailed description of the DIVA FPU, refer to [9][10].

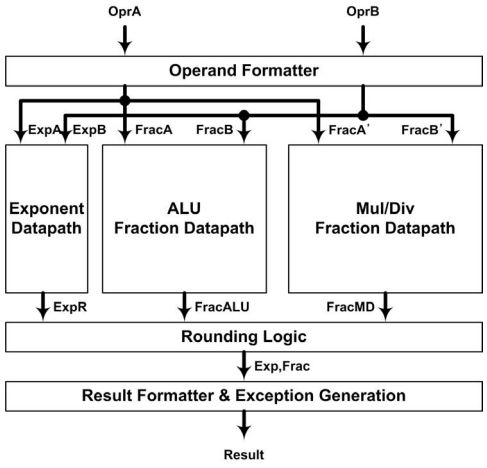

Figure 4. DIVA FPU organization

III. IMPLEMENTATION

Both FPU designs have been described in Verilog with the exception of the two-stage multiplier, where the netlist was generated using Synopsys synthesis tools. These netlists along with the ROM needed for the divider were combined together. For balanced pipeline stages and generating a 2-stage multiplier, register retiming techniques have been generally applied in the logic synthesis step. The FPUs were synthesized to 0.18µm technology under the timing constraint of a 266MHz clock frequency, and the resulting netlists were then placed and routed to generate a layout using Cadence Silicon Ensemble. The layouts and features of both FPUs are presented in Fig. 5 and Table I, respectively.

IV. COMPARISON RESULTS

A. DIVA FPU vs. MONARCH FPU

The implementation results show that the area of the DIVA FPU is 19.7% smaller than the area of the MONARCH FPU. The area overhead of the DIVA FPU 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, preserving the majority of the die area for memory. This area optimization has been achieved through sub-block sharing among different functions. Another factor of area reduction results from the architectural difference between the two designs. Since the DIVA FPU is a 5-stage pipelined architecture and the MONARCH FPU is a 3-stage pipelined architecture, a smaller design is generated in the case of the DIVA FPU in the synthesis step, when the same timing constraint is applied, as less logic per stage and overall is

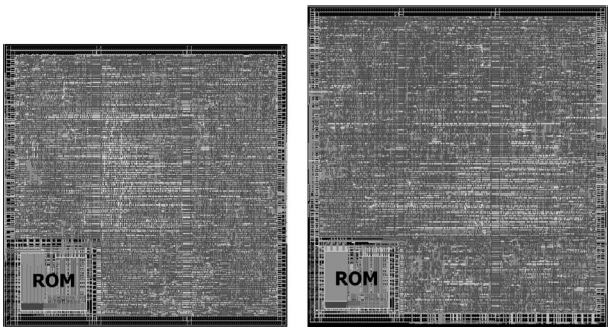

Figure 5. DIVA FPU layout and MONARCH FPU layout

TABLE I. Summary of prototype FPUs

	DIVA FPU	MONARCH FPU
Technology	TSMC 0.18µm CMOS	
Supply Voltage	1.8V	
Dimension	695µm x 693µm	750µm x 800µm
Gate Count	9,467	12,539
Transistor Count	105,539	119,777
Speed	266MHz @1.8V	

needed to meet the timing requirement. This architectural difference also shows that the MONARCH FPU exhibits superior performance to the DIVA FPU. In addition to supporting concurrent add/multiply operations, the latency of the MONARCH FPU is 3 clock cycles, while the latency of the DIVA FPU is 5 clock cycles (for all operations except division in both cases, as noted earlier).

B. 1-stage multiplier vs. 2-stage multiplier

A 2-stage pipelined multiplier was used in the DIVA FPU fraction datapath after several design considerations. By using a 2-stage multiplier, the number of stages is larger, therefore increasing the latency. However, the area of the overall synthesized design is smaller than one using a 1-stage multiplier because of the smaller size of the synthesized fraction multiplier, which occupies a substantial area in an FPU design. Since there are several multiply operations in the division algorithm used for both FPU designs, the multiplier architecture also affects the overall performance of division operations. When a 2-stage multiplier is used, the latency of the division operation increases from 9 clock cycles to 12 clock cycles. However, there is a slight throughput advantage of using a 2-stage multiplier since consecutive division instructions can be issued at every 5 clock cycles, while 6 clock cycles of instruction issue delay are required to ensure in-order completion when a 1-stage multiplier is used.

C. Power Dissipation

Fig. 6 presents simulation results of the power dissipation of each FPU design for each instruction, where the same instruction was repeated on a random input data stream at the highest throughput rate. The results show that the DIVA FPU consumes approximately 41% more power than MONARCH FPU as the average power dissipation of the DIVA FPU is 100.5mW and the average power dissipation of the MONARCH FPU is 71.2mW with a clock frequency of 266MHz at 1.8v. This difference results mainly from the difference between the organizations of the two FPU designs. Even though one of either the ALU or Mul/Div fraction datapath is active by controlling the enable signals of the pipeline registers in the DIVA FPU, there are several components shared between the ALU and the Mul/Div blocks that operate continuously. On the other hand, since the ALU block and the Mul/Div block inside the MONARCH FPU are completely separated, exactly only one of them is active at all times for single-instruction issue. If the MONARCH ALU and Mul/Div blocks are operated concurrently, then obviously the power dissipation of the MONARCH FPU is larger than that of the DIVA FPU. The average power dissipation for multiply and divide instructions is more than other ALU type instructions, mainly resulting from the larger datapath of the Mul/Div block.

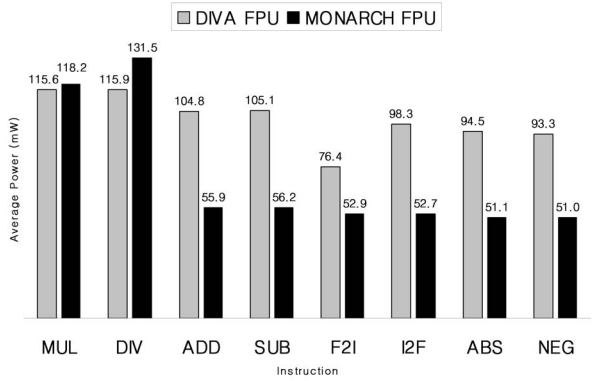

Figure 6. Power dissipation profile of each FPU

V. CONCLUSION

This paper presented two types of FPU architectures optimized for different design goals, their design trade-offs, implementation details and comparison results. Standard cell implementations based on TSMC 0.18μm CMOS technology have shown that each FPU design is well optimized to satisfy the requirements of its applications. A single-instruction issue design is implemented in very small area; however, a design capable of concurrently executing FP add and multiply instructions is achievable with roughly a 24% area increase. The DIVA FPU has been implemented in a PIM chip, and it is fully functional [10]. The MONARCH chip incorporating the other described FPU will tape out in 2005.

ACKNOWLEDGMENT

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] J. Granacki and M. Vahey, "MONARCH: A Morphable Networked micro-ARCHitecture", presentation to *High Performance Embedded Computing Workshop*, October 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] "IEEE Standard for Binary Floating-Point Arithmetic", ANSI/IEEE Standard 754, Aug. 1985

[6] A. Liddicoat, "*High-Performance Arithmetic for Division and the Elementary Function*", Ph.D. dissertation, Stanford University, Feb. 2002

[7] A. Liddicoat and M.J. Flynn, "High-Performance Floating-Point Divide", *Euromicro Symposium on Digital System Design*, Sep. 2001

[8] 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

[9] 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

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