POSTER: An Integrated Vector-Scalar Design on an In-order ARM Core

Milan Stanic
Barcelona Supercomputing Center
Barcelona, Spain
milan.stanic@bsc.es

Oscar Palomar
University of Manchester
Manchester, UK
oscar.palomar@manchester.ac.uk

Timothy Hayes
Barcelona Supercomputing Center
Barcelona, Spain
timothy.hayes@bsc.es

Ivan Ratkovic
Barcelona Supercomputing Center
Barcelona, Spain
ivan.ratkovic@bsc.es

Osman Unsal
Barcelona Supercomputing Center
Barcelona, Spain
osman.unsal@bsc.es

Adrian Cristal
Barcelona Supercomputing Center
Barcelona, Spain
adrian.cristal@bsc.es

ABSTRACT

In the low-end mobile processor market, power, energy and area budgets are significantly lower than in other markets (e.g. servers or high-end mobile markets). It has been shown that vector processors are a highly energy-efficient way to increase performance; however adding support for them incurs area and power overheads that would not be acceptable for low-end mobile processors. In this work, we propose an integrated vector-scalar design for the ARM architecture that mostly reuses scalar hardware to support the execution of vector instructions. The key element of the design is our proposed block-based model of execution that groups vector computational instructions together to execute them in a coordinated manner.

CCS Concepts

•Computer systems organization → Single instruction, multiple data;

Keywords

vector processors; low-power; energy efficiency; mobile

1. INTRODUCTION

Vector processors [1] are energy efficient architectures that yield high performance whenever there is enough data-level parallelism (DLP) [7]. Besides the long and successful history of vector processors in supercomputers, vector units have been proposed in microprocessor design [6, 4, 2]. Recent research on vector processors shows that they can be a good match even for applications from domains such as column-store databases [5]. Although vector processors are energy efficient, they still have too high power and area overheads for low-end mobile processors. This is mostly due to their highly restrictive power and area budget.

This paper contributes a method to increase the performance of the low-power low-end embedded systems in an energy-efficient way. The energy efficiency is attained by modifying a scalar core to execute vector instructions on the existing infrastructure. In particular, we propose an integrated vector-scalar design that combines scalar and vector processing mostly using existing resources of an energy-efficient scalar processor (in our evaluation environment it is based on the ARM Cortex A7). In addition to a design that uses a conventional vector execution model, we also contribute a novel block-based model of execution for vector computational instructions.

2. INTEGRATED DESIGN

As a baseline, we use a scalar core based on the highly energy-efficient ARM Cortex-A7. It is an in-order processor that implements the ARM v7 architecture with an 8-stage pipeline (non-highlighted gray blocks in Figure 1).

In our proposed integrated vector-scalar design, we attempt to maximize the reuse of resources already present in the baseline scalar core (white blocks in Figure 1) while adding support for vector instructions. While the front-end of the pipeline is the same (fetch and decode[1] stages), in the back-end we added two structures to support the execution of vector instructions on the scalar core: a vector register file, and a vector memory unit (blue blocks in Figure 1). There is also additional logic that controls the execution of vector instructions. Vector execution control logic (VECL) is added in the issue stage to support the execution of computational vector instructions. Aliasing control logic (ACL) exchanges information between the vector memory and the data cache unit and forces scalar and vector memory instructions to be executed in-order. We implement support for chaining [9], a well-known concept in vector processors. Similar to result forwarding in scalar processors, chaining allows starting the execution of a dependent vector instruction as soon as the first element of the vector is generated

[1]With the obvious exception of the decode logic, which needs to be extended to support the new vector instructions.

PACT '16, September 11–15, 2016, Haifa, Israel.

ACM ISBN 978-1-4503-4121-9/16/09..

DOI: http://dx.doi.org/10.1145/2967938.2974057

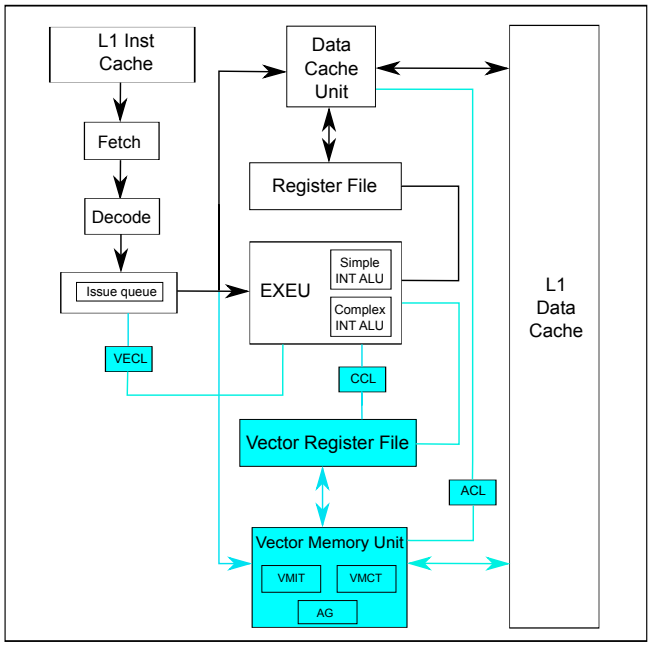

Figure 1: Block diagram of the integrated design.

by the previous computational instruction. Chaining control logic (CCL) is responsible for the execution of chained dependent computational instructions.

2.1 Vector Computational Instructions

For executing the vector computational instructions on the existing scalar FUs, we study two alternatives: 1) the One-By-One model of execution (*OBO*), in essence the classic vector execution model, in which a vector instruction is executed to completion once it starts execution in a functional unit, i.e. for all the operations of the vector; and 2) a novel execution model called Block-Based Execution (*BBE*). In this model, for a block of consecutive vector computational instructions, first all operations on the first element of the vectors are executed, then the operations of the second element, and so on.

2.1.1 Block-Based Execution

In order to support this model of execution, we added simple control logic and a small table that keeps the information of the instructions of the block. In the design presented in this paper, the blocks of vector computational instructions are formed dynamically in a very simple way: once a computational vector instruction is ready for execution, the control logic examines the next instruction in the issue queue and adds it to the block if it is a vector computational instruction. This process stops when the next instruction in the issue queue is of another type (a scalar or vector memory instruction) or the block table is full.

3. INITIAL EVALUATION

We have extended the gem5 simulator [3] and McPAT [8] to evaluate our integrated design. We used eight kernels from various benchmarks and vectorized them. The results show that our integrated design reduces energy over the scalar baseline for most of the kernels with a small area overhead (only 4.7% when using a vector register with 32 elements). We report up to 5x energy reduction for our block-based execution model over the scalar baseline. Additionally, we found that the block-based execution model provides better results (up to 26% of energy saving) than a conventional vector unit with dedicated units. The area overhead of adding the conventional vector unit with a floating-point unit is significant, around 44% with vector registers of 32 elements. Regarding performance gains, we report more than a 6x speed-up compared to the scalar baseline. Moreover, our block-based execution model is up to 1.4x faster than the conventional vector unit for floating-point kernels.

4. CONCLUSION

Power dissipation, energy consumption and area are critical concerns in processor design, especially for embedded systems in the low-end market. In this paper, we propose an integrated vector-scalar design. The integrated design allows for execution of vector computational instructions mostly reusing resources of an ARM in-order core. We implement two models to execute vector computational instructions: one-by-one and block-based execution models. Initial evaluation shows substantial savings.

5. ACKNOWLEDGMENTS

The research leading to these results has received funding from the RoMoL ERC Advanced Grant GA no 321253 and is supported in part by the European Union (FEDER funds) under contract TIN2015-65316-P. This research has been also supported the Agency for Management of University and Research Grants (AGAUR - FI-DGR 2014).

6. ADDITIONAL AUTHORS

Additional authors: Mateo Valero (Barcelona Supercomputing Center, email: `mateo.valero@bsc.es`).

7. REFERENCES

[1] K. Asanovic. *Vector Microprocessors*. PhD thesis, University of California, Berkeley, May, 1998.

[2] C. F. Batten. *Simplified vector-thread architectures for flexible and efficient data-parallel accelerators*. PhD thesis, Cambridge, MA, USA, 2010.

[3] Binkert et al. The gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, aug 2011.

[4] R. Espasa et al. Tarantula: A vector extension to the alpha architecture. ISCA '02, pages 281–292, 2002.

[5] T. Hayes et al. Vector extensions for decision support dbms acceleration. In *MICRO 45*, pages 166–176, 2012.

[6] C. Kozyrakis and D. Patterson. Overcoming the limitations of conventional vector processors. In *ISCA '03*, pages 399–409, 2003.

[7] Y. Lee et al. Exploring the tradeoffs between programmability and efficiency in data-parallel accelerators. ISCA '11, pages 129–140, 2011.

[8] S. Li et al. Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *MICRO 42*, pages 469–480, Dec 2009.

[9] R. M. Russell. The CRAY-1 computer system. *Commun. ACM*, 21:63–72, Jan. 1978.