Vectorization for Digital Signal Processors via Equality Saturation Extended Abstract

Alexa VanHattum
Cornell University

Rachit Nigam
Cornell University

Vincent T. Lee
Facebook Reality Labs Research

James Bornholt
The University of Texas at Austin

Adrian Sampson
Cornell University

1. Motivation

Digital signal processors (DSPs) are ubiquitous and unmatched in their efficiency for embedded sensing applications, but they are difficult to program. Their simple in-order pipelines, their exotic VLIW and vector instruction sets, and their per-deployment hardware variability means that traditional compilers generate code that is often far from optimal. Instead, DSP engineers typically hand-tune implementations of critical, fixed-size kernels to extract the best performance from a specific DSP target. Applications such as such as simultaneous localization and mapping (SLAM) [12, 13, 19, 20] and structure from motion [21] rely on components dominated by a variety of small-scale DSP kernels, so this kind of manual, kernel-by-kernel optimization can pay off.

Expert tuning, however, is difficult to scale to the diversity of DSP hardware. DSPs often offer per-application hardware customization, where device makers can select a subset of an instruction set tailored for their particular application and even add custom proprietary instructions [7]. Even worse, applications often need size-specific, specialized variants of DSP kernels: for example, products and convolutions of small 3×3 and 4×4 matrices are commonplace in various machine perception applications. Manually optimizing each kernel size for each possible DSP target represents an enormous engineering cost.

This paper designs a compiler, Diospyros, that aims to compete with manual tuning by DSP experts. Diospyros frames compilation as a search problem in a space of candidate programs. It uses a system of rewrite rules to define a search space that encompasses both high-level functional specifications and low-level device-specific instructions. Crucially, rewrite rules in Diospyros can perform complex data movement to enable efficient use of the fixed-width vector SIMD units common in DSP architectures. Unlike traditional approaches to general-purpose vectorization [9], Diospyros generates irregular shuffle operations that pack as much work as possible into vector lanes. This focus on data movement allows Diospyros to effectively optimize specialized small-scale signal processing kernels that dominate many DSP applications, and that existing DSP compilers struggle with.

To identify the most efficient vectorized compilation of a scalar input program, Diospyros exhaustively searches the space of candidates using *equality saturation* [8, 22, 25]. Equality saturation lets Diospyros explore all possible applications of its rewrite rules in any order by representing the search space as an equality graph (E-graph) [14]. From this saturated E-graph, Diospyros extracts the most efficient program according to a cost model and lowers it to C code with target-specific vector intrinsics that can be passed through a vendor-supplied DSP compiler for code generation.

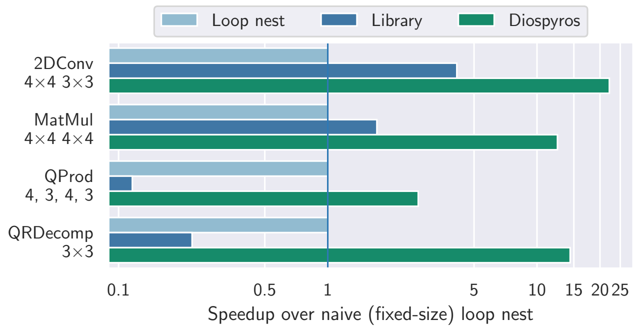

Figure 1: Speedup for a sample of kernels compiled with Diospyros, compared to naive loop nests and optimized library routines. This figure highlights kernels where Diospyros does well; the paper's Figure 5 shows full results.

2. Limitations of the State of the Art

Optimizations for automatic vectorization Classical vectorization techniques include loop-based vectorizers [1], superword level parallelism (SLP) optimizations [9], and modern descendants [11, 15]. These optimizations typically do not attempt to aggressively shuffle data to fit it into fixed-size vector units, which is Diospyros's main goal. Their focus is fundamentally different. Classic vectorization passes aim to cover large codebases quickly and heuristically; Diospyros focuses on individual, high-value kernels and searches for an optimal implementation. Classic approaches work best on regular loops over large arrays that allow the computation to reach a "steady state"; Diospyros focuses on the small, fixed-size kernels that often arise in DSP applications, where it is critical to rearrange data to pack it into vector lanes.

Program synthesis for high-performance kernels Other systems have applied synthesis techniques to find efficient kernel implementations [2, 16, 26]. However, these systems face scalability challenges common to program synthesis; Diospyros can synthesize kernels several times larger than previous efforts focused on DSPs or other embedded applications [4, 23]. Diospyros's abstract vector DSL is also more portable than most synthesis-based compilation techniques, which require a detailed semantics for the target architecture.

Optimizing linear algebra kernels There is a long line of work on efficient compilation for DSP code, including vectorization [5, 10, 24]. These techniques can often generate target-specific shuffle code to implement pre-specified permutation patterns, but they do not search for kernel-specific data movement strategies themselves. These approaches rely on domain expertise and hardware-specific engineering to generate fast code. Diospyros avoids baking in any specific data movement strategies and instead expends computation time to automate the search for optimized implementations.

The SPIRAL project [6, 17], and particularly the SLinGen tool for small fixed-size linear algebra kernels [18], proposes a range of hand-tuned compilation strategies to optimize DSP applications. Like SLinGen, Diospyros works at a higher abstraction level to enable optimizations that assembly would obscure. However, equality saturation allows Diospyros to both avoid hand-crafting specific optimization patterns and cover a larger search space than SLinGen's autotuning.

3. Key Insights

- Performance on DSP hardware depends on generating irregular "shuffle" instructions to keep vector units busy.
- Equality saturation can be applied to vectorization and, without prioritizing target-specific heuristics, yield better performance than an existing vectorizing DSP compiler.
- To make equality saturation scale to realistic DSP kernels without running out of memory, it helps to limit the application of rewrite rules that can blow up the size of the E-graph, by applying them only speculatively and re-computing them when necessary.

4. Main Artifacts

- Diospyros, an open-source vectorizing compiler for DSPs based on rewrite rules with equality saturation. Diospyros currently targets the Tensilica Fusion G3 family of digital signal processors [3].
- A methodology for designing rewrite rules that allow for flexibility in matching vector operations while avoiding the exponential blowup caused by incorporating unrestricted operator associativity and commutativity.

5. Key Results and Contributions

Empirical results:

- Kernels compiled with Diospyros outperform the best non-hand written alternative (usually the optimized Nature math library shipped with the Tensilica DSP SDK) by 3.0× on average, as Figure 1 shows. Compared to an expert-written kernel hand-tuned for a single fixed matrix size, Diospyros produces a kernel with performance within 17% in 2.7 seconds of compilation time.
- As a case study, we integrate a Diospyros-generated kernel into an open-source computer vision library and demonstrate an end-to-end speedup of 2.1× compared to the baseline.

Contributions:

- We distill the challenges of programming high-performance kernels on DSP hardware: simple hardware puts the burden of optimization on the programmer; data sizes close to the machine vector width create a need for irregular and unintuitive data movement code; and machine-specific vector instructions limit code portability.
- We demonstrate that equality saturation can apply to vectorization and can automatically generate critical data shuffling strategies, outperforming an existing vectorizing compiler for a DSP architecture.
- We describe a speculation strategy for limiting the application of common rewrite rules in equality saturation, such as commutativity and the additive identity, in order to avoid an exponential blowup in memory requirements.

Advantages over past work:

- Unlike standard compiler passes for auto-vectorization, Diospyros can identify novel data movement strategies that are critical for DSP performance on common kernels.
- Unlike prior systems that use program synthesis for high-performance code generation, Diospyros uses rewrite rules over an abstract vector DSL to enable portability and to scale better than synthesizers that require detailed machine-level models.
- Unlike prior compiler infrastructure that targets DSPs, Diospyros uses a generic rewriting approach and mostly avoids baking in heuristics that tie it to a particular DSP architecture or problem domain.

6. Why ASPLOS

This paper is about extracting performance from a kind of architecture that relies on software optimization for efficiency. It applies ideas from the programming languages and formal methods worlds—term rewriting systems, equality saturation, and symbolic evaluation—to expose instruction- and data-level parallelism that neither standard compilers nor DSP hardware can discover on their own.

7. Citation for Most Influential Paper Award

The Diospyros paper pioneered the use of equality saturation to implement compilers for niche, domain-specific, and "moving target" hardware. Hand-engineered compiler heuristics and expert-written kernel libraries sufficed in the era when CPU ISAs remained stable for decades, but the early 2020s saw simultaneous explosions in domain-specific processors and in linear algebra applications that necessitated a new approach. By showing that a search-based strategy based on saturating a system of rewrite rules could replace target-specific tuning for digital signal processors (DSPs), the paper initiated a line of work on building flexible compilers that can adapt to rapid changes in the hardware they target.

References

[1] Randy Allen and Ken Kennedy. Automatic translation of FORTRAN programs to vector form. In *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 1987.

[2] Gilles Barthe, Juan Manuel Crespo, Sumit Gulwani, César Kunz, and Mark Marron. From relational verification to SIMD loop synthesis. In *ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*, 2013.

[3] Cadence Design Systems, Inc. Tensilica customizable cores, 2020. `https://ip.cadence.com/ipportfolio/tensilica-ip/xtensa-customizable`.

[4] Meghan Cowan, Thierry Moreau, Tianqi Chen, James Bornholt, and Luis Ceze. Automatic generation of high-performance quantized machine learning kernels. In *ACM/IEEE International Symposium on Code Generation and Optimization (CGO)*, 2020.

[5] Franz Franchetti and Markus Püschel. Generating SIMD vectorized permutations. In *Proceedings of the International Conference on Compiler Construction*, 2008.

[6] Franz Franchetti, Yevgen Voronenko, and Markus Püschel. A rewriting system for the vectorization of signal transforms. In *International Conference on High Performance Computing for Computational Science (VECPAR)*, 2006.

[7] Ricardo E Gonzalez. Xtensa: A configurable and extensible processor. *IEEE Micro*, 20(2):60–70, 2000.

[8] Rajeev Joshi, Greg Nelson, and Keith H. Randall. Denali: A goal-directed superoptimizer. In *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, 2002.

[9] Samuel Larsen and Saman Amarasinghe. Exploiting superword level parallelism with multimedia instruction sets. In *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, 2000.

[10] Daniel S. McFarlin, Volodymyr Arbatov, Franz Franchetti, and Markus Püschel. Automatic SIMD vectorization of fast fourier transforms for the larrabee and AVX instruction sets. In *Proceedings of the International Conference on Supercomputing*, 2011.

[11] Charith Mendis and Saman Amarasinghe. GoSLP: Globally optimized superword level parallelism framework. In *ACM SIGPLAN Conference on Object Oriented Programming, Systems, Languages and Applications (OOPSLA)*, 2018.

[12] Raul Mur-Artal and Juan D Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5): 1255–1262, 2017.

[13] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5): 1147–1163, 2015.

[14] Greg Nelson. *Techniques for program verification*. PhD thesis, Stanford University, 1980.

[15] Dorit Nuzman, Ira Rosen, and Ayal Zaks. Auto-vectorization of interleaved data for SIMD. In *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, 2006.

[16] Phitchaya Mangpo Phothilimthana, Archibald Samuel Elliott, An Wang, Abhinav Jangda, Bastian Hagedorn, Henrik Barthels, Samuel J. Kaufman, Vinod Grover, Emina Torlak, and Rastislav Bodík. Swizzle Inventor: Data movement synthesis for GPU kernels. In *ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2019.

[17] Markus Puschel, José MF Moura, Jeremy R Johnson, David Padua, Manuela M Veloso, Bryan W Singer, Jianxin Xiong, Franz Franchetti, Aca Gacic, Yevgen Voronenko, et al. SPIRAL: Code generation for DSP transforms. *Proceedings of the IEEE*, 93(2):232–275, 2005.

[18] Daniele G. Spampinato, Diego Fabregat-Traver, Paolo Bientinesi, and Markus Püschel. Program generation for small-scale linear algebra applications. In *ACM/IEEE International Symposium on Code Generation and Optimization (CGO)*, 2018.

[19] Hauke Strasdat, Andrew J Davison, JM Martìnez Montiel, and Kurt Konolige. Double window optimisation for constant time visual slam. In *2011 international conference on computer vision*, pages 2352–2359. IEEE, 2011.

[20] Shinya Sumikura, Mikiya Shibuya, and Ken Sakurada. OpenVSLAM: A Versatile Visual SLAM Framework. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, pages 2292–2295, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6889-6. doi: 10.1145/3343031.3350539. URL `http://doi.acm.org/10.1145/3343031.3350539`.

[21] Chris Sweeney. Theia multiview geometry library: Tutorial & reference. `http://theia-sfm.org`.

[22] Ross Tate, Michael Stepp, Zachary Tatlock, and Sorin Lerner. Equality saturation: a new approach to optimization. In *ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, 2009.

[23] Alexa VanHattum, Rachit Nigam, Vincent T. Lee, James Bornholt, and Adrian Sampson. A synthesis-aided compiler for DSP architectures (wip paper). In *ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES)*, 2020.

[24] Sander Vocke, Henk Corporaal, Roel Jordans, Rosilde Corvino, and Rick Nas. Extending halide to improve software development for imaging DSPs. In *ACM Transactions on Architecture and Code Optimization (TACO)*, 2017.

[25] Max Willsey, Yisu Remy Wang, Oliver Flatt, Chandrakana Nandi, Pavel Panchekha, and Zachary Tatlock. egg: Fast and extensible e-graphs, July 2020. `https://arxiv.org/abs/2004.03082`.

[26] Zhilei Xu, Shoaib Kamil, and Armando Solar-Lezama. MSL: A synthesis enabled language for distributed implementations. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2014.