Hardware/Software Co-design for Energy-Efficient Seismic Modeling

Jens Krueger[*†], David Donofrio[*], John Shalf[*], Marghoob Mohiyuddin[*§],
Samuel Williams[*], Leonid Oliker[*], Franz-Josef Pfreundt[†]

[*] *Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA, USA*
[§] *Computer Science Division, University of California at Berkeley, Berkeley, CA, USA*
[†] *Fraunhofer ITWM, Kaiserslautern, Germany*
{jtkrueger, ddonofrio, jshalf, mmohiyuddin, swwilliams, loliker}@lbl.gov, franz-josef.pfreundt@itwm.fraunhofer.de

ABSTRACT

Reverse Time Migration (RTM) has become the standard for high-quality imaging in the seismic industry. RTM relies on PDE solutions using stencils that are 8^th order or larger, which require large-scale HPC clusters to meet the computational demands. However, the rising power consumption of conventional cluster technology has prompted investigation of architectural alternatives that offer higher computational efficiency. In this work, we compare the performance and energy efficiency of three architectural alternatives – the Intel Nehalem X5530 multicore processor, the NVIDIA Tesla C2050 GPU, and a general-purpose manycore chip design optimized for high-order wave equations called "Green Wave." We have developed an FPGA-accelerated architectural simulation platform to accurately model the power and performance of the Green Wave design. Results show that across highly-tuned high-order RTM stencils, the Green Wave implementation can offer up to 8× and 3.5× energy efficiency improvement per node respectively, compared with the Nehalem and GPU platforms. These results point to the enormous potential energy advantages of our hardware/software co-design methodology.

Keywords

seismic, RTM, stencil, GPU, co-design, manycore

1. INTRODUCTION

In recent years Reverse Time Migration (RTM)[5] has become the high quality standard for seismic imaging for the oil and gas exploration industry. Although the RTM method has been well established for many years, its application has been limited due to the high computational requirements of this method, which require large-scale HPC systems operating for months at a time on a single problem. In the past, the capital expense of acquiring the HPC platform dominated total cost of ownership (TCO), but is rapidly being outstripped by the operational expenses of power and cooling for these systems [22]. As power becomes a primary cost in high-end computing, any effective large-scale solution for next-generation RTM simulations must deliver performance in an energy-efficient manner. The free-lunch of getting performance improvements from ever-increasing clock frequencies is over [2] and more radical approaches to improving the energy efficiency of computer architectures are going to be required to avert a power crisis or catastrophic stall in computing performance [14]. The quest for more energy-efficient approaches has spawned a revolution in multicore technology and renewed interest in alternative architectures such as GPUs (graphics processing units), FPGAs (field-programmable gate arrays) and other novel solutions with a complex tradeoffs between performance, programmability, and energy-efficiency. In this work we examine the performance and power requirements of the high-order wave equation stencils found at the heart of RTM wavefield modeling on modern Nehalem X5530 CPUs and NVIDIA Fermi C2050 GPUs, and an energy-efficient manycore solution based on low-power embedded cores called Green Wave.

The Green Wave design is built upon energy-efficient embedded processor cores [41], and uses a hardware/software co-design methodology to maximize RTM-based stencil performance and efficiency. The design remains fully programmable and general purpose, but uses the co-design process to optimize energy efficiency for the target workload. This is different from a fixed-function logic design that cannot depart from its designed purpose. To facilitate the co-design process, we develop a rapid design prototyping framework named CoDEx [37] that uses the standard toolchain to rapidly synthesize gate-level RTL implementations of the target node design and accurately predict the performance and power characteristics of this chip design for the RTM application using validated cycle-accurate logic and DRAM simulators. Overall, compared to highly optimized Nehalem and Fermi implementations, Green Wave demonstrates up to an order of magnitude improvement in energy efficiency — highlighting the potential of a semi-custom co-design approach as a path towards designing high-performance, energy-efficient systems while maintaining full programmability.

This work makes several contributions. First, it is one of the few studies to present a direct comparisons between a highly optimized CPU and GPU implementations of RTM kernel calculations. We also introduce a manycore-based chip architecture and system design that uses commodity off-the-shelf IP (intellectual property) components from the

SC11, November 12-18, 2011, Seattle, Washington, USA
Copyright 2011 ACM 978-1-4503-0771-0/11/11 ...$10.00.

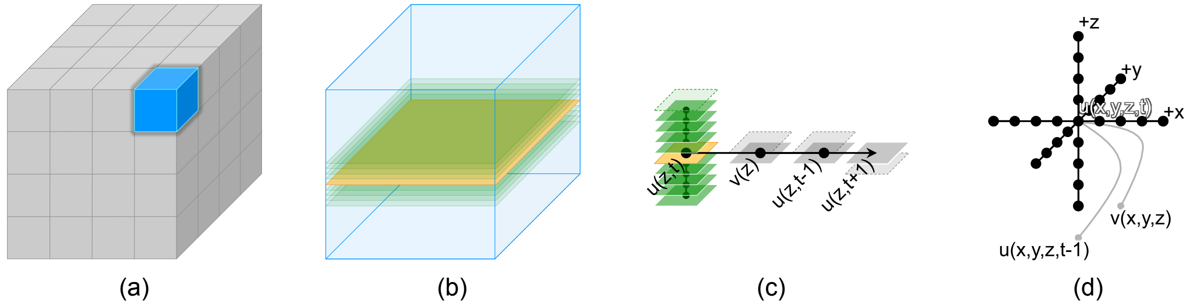

Figure 1: RTM Problem Decomposition and Stencils (a) generic decomposition, (b) streaming planes, (c) multibuffering, (d) wave equation stencil in which black represents the Laplacian Stencil and gray represents the time derivative.

embedded space thereby reducing the cost and risk of producing such an ASIC (application-specific integrated circuit). Additionally, we demonstrate how a hardware/software co-design process that uses cycle-accurate FPGA hardware emulation can iteratively optimize the design for RTM-stencil calculation requirements. Finally, we demonstrate that a co-design process can create a fully-programmable general-purpose design, while offering an order of magnitude better energy efficiency than conventional approaches for the target application. This design process holds promise as a novel approach for creating computationally efficient systems to tackle a wide variety of demanding numerical problems, and reduces the cost and schedule risks of designing a full-custom machine architecture. Overall, our system design enables the seismic industry to take the next step in large-scale high quality survey processing and leads the way towards interactive seismic modeling on mobile platforms.

2. OVERVIEW AND RELATED WORK

In numerous seismological applications it is necessary to create a subsurface reflectivity image to perform various analyses of the Earth's interior including crust development and exploration for hydrocarbons or minerals. This process entails collecting seismic data of the area via an energy source, such as explosion, that generates acoustic waves which are reflected by impedance contrasts of rock layers. Each of these "shots" has an array of receivers that listen to reflection for several seconds, followed by the shot movement in equidistant offsets until the area of interest is covered. To effectively translate this process into a quality image, a sophisticated seismic processing workflow has been developed involving several iterative steps including data acquisition and preprocessing, velocity modeling, wavefield migration and imaging [47]. In this work we employ a hardware/software co-design process focused on reducing the overhead of the seismic wavefield migration phase — the most computational-intensive component of this workflow methodology. This phase is used to correct mispositioned reflectors by cross-correlating the source and receiver wavefield to the desired subsurface image.

2.1 Reverse Time Migration

The RTM algorithm is composed of three main steps. In the first, the source wavefield is propagated forward in time starting from a synthetic simulated source signal at time zero. Next, the receiver wavefield is computed by propagating the receiver data backwards in time. Finally, the imaging condition cross-correlates the source and receiver wavefield to create the final subsurface image. Our study focuses on the wavefield propagation kernel in the first two steps and requires most of the overall RTM computation time. Specifically, the kernel consumes more than 90% of execution time during the forward propagation step based on benchmark timings of current optimized RTM implementations [1].

Recent advances in computer architecture have allowed this numerically intensive technique to be utilized in production seismic computing. However, its large computational demands result in extremely slow processing or limited, low-resolution seismic analysis on large-scale systems. It is therefore critical to improve the performance and energy efficiency of these techniques for next generation processing.

The simulation of the wavefield propagation is performed most commonly with an approximation of the wave equation represented, where c is the velocity function, u is the pressure function at point $(x, y, z) \in \mathbb{R}^3$: $\left(\Delta - \frac{1}{c^2}\frac{\partial^2}{\partial t^2}\right) u = 0$.

The approximation to this equation can be derived using either implicit or explicit techniques. The former leads to large linear systems that can leverage large-scale parallel solvers such as PETSc [3] but can suffer from scalability limitations at high concurrency. The explicit approach, examined here, computes the next timestep of the wavefield propagation via a "stencil" update for each point using fixed timesteps. Note that 32-bit values are standard across the seismic industry, provide sufficient accuracy to receive high quality imaging results, and are used throughout our study. Even lower bit widths have been considered to further increase performance of FPGA-based solutions [16].

2.2 Higher Order Stencils

Stencil computations are used in a broad range of scientific applications involving finite-differencing methods to solve partial differential equations (PDEs) on regular grids. At each point, a nearest-neighbor computation (stencil) is performed, where the value is updated with weighted contributions from a subset of points neighboring in time and space. The stencil "order" is a theoretical bound on the rate at which error decreases relative to increased resolution of pressure field derivation approximation. Here, we also refer to order as the most distant stencil calculations point. Thus, high-order stencils typically generate large working sets.

Figure 1 shows an example of an RTM decomposition and stencil computational structure. Figure 1(a) depicts the large grid associated with each process. Figure 1(b) and

Wave Equation	$\mathbf{8}^{th}$	$\mathbf{12}^{th}$
Points in Laplacian component	25	37
Points in time derivative component	2	2
Velocity component	1	1
Total points accessed per wave equation stencil	27	39
Floating-point operations (flops) per wave equation stencil	26 add, 7 mul	38 add, 9 mul
Compulsory DRAM Bytes per Stencil	16.2	16.3
Typical Bytes per stencil (512^3 with 256 KB local store)	17.6	18.3
Grid Volume (512^3)	2.1 GB	2.1 GB
Ghost Zone Volume (512^3)	24.4 MB	36.9 MB

Table 1: Characteristics of isotropic 8^{th} and 12^{th} order wave equations. Both the Laplacian and time derivatives share a common point, reducing the total number of points per wave equation stencil by one.

(c) visualize optimization techniques detailed in Section 5. As discussed, the wave equation is comprised of a Laplacian stencil and a time derivative. Figure 1(d) visualizes a 8^{th}-order Laplacian stencil (in black) and a 2^{nd}-order time derivative (in gray). The Laplacian component requires accessing 25 points and performing a linear combination using 5 weights (one for each of the four equidistant sextuplets of grid points and one for the center). Thus, the Laplacian performs 5 floating-point multiplies and 24 floating-point additions. The wave equation's time derivative requires accessing not only the spatial grid point at the current and previous time steps, but also the medium's velocity at that point. When the Laplacian and time derivative are combined, we see the complexity of the inhomogeneous isotropic wave equation's stencil. Higher order Laplacian stencils (e.g. 12^{th}-order) will access neighboring points further from the center and induce a corresponding increase in computation.

Table 1 details the computational and bandwidth characteristics of the two wave equation implementations, 8^{th} and 12^{th} order. A 512^3 problem size per node was selected for the cross-comparison between architectures because it presents the best performance on the CPU and GPU and it fully occupies the GPU on-board memory. In this paper we focus on the forward propagating component of RTM, which provides an accurate characterization of the computational requirements of the seismic processing application [1].

2.3 Survey Analysis Resource Requirements

The computational resource decisions of seismic survey analysis are based on today's largest 3D marine seismic studies combined with fine grained resolutions, which result in an upper boundary for migration parameters that have to be handled by the system. The enormous financial expenses of data collection alone makes it mandatory to cover as much area as possible with a single survey. Additionally, new numerical methods offer advances in image fidelity at the cost of increased computational overhead. Thus, there is strong motivation to accelerate processing rates and image quality without increasing power costs, lest the increased energy consumption negate the potential cost savings.

Consider a modern survey size of 30km in streamline (x), 20km in crossline (y) and 10km in depth (z). To conduct this analysis, an exploration ship tows 10 streamer lanes with 1000 receivers (e.g. microphones) each. All receivers have a time sampling interval of 1ms and listen a total of 12 seconds to reflections from the subsurface. Thus, 12,000 timesteps have to be processed in order to receive the wavefield for the next timestep. With shot offsets of 50m × 100m, 120,000 shots are necessary to cover the survey area. The calculation is proceeded on a space grid of 5m in all dimension. With these space sampling intervals a total shot volume of $4000\times4000\times2000$ points has to be processed for each shot for all timesteps (rounded up to $4096\times4096\times2048$ to ease binary handling). On current systems, it takes months to process the sheer scale of such a survey.

Given these requirements we now examine the most time intensive components of RTM: forward and backward RTM wavefield modeling. For the given survey size to reach a computational time of a week, it would require the largest supercomputer in the world containing more than 1M cores and consuming at least 38 MW of power. However, the energy cost alone of operating such a large system is likely prohibitive, as even inexpensive power (7 cents/kwh) translates to approximately \$1M/year per megawatt for a total of \$38M dollars per year. The goal of this project is therefore to demonstrate that using an HPC platform tailored to the requirements of the RTM computation, can enable a one week turn around for seismic image processing with an order of magnitude improvement in the power requirements. This kind of energy efficiency would enable the unprecedented potential of interactive processing, resulting in a transformation improvement in the seismic processing workflow.

2.4 Related Work

There have been numerous studies examining architectural performance on seismic algorithms [8, 25, 33]. Techniques to accelerate these methods have also been explored on FPGA platforms [17, 12, 31]. Recently, researchers have evaluated optimization schemes on GPUs [15, 26]. Various approaches have also been examined to address the bandwidth limitations of seismic migration codes [43, 13, 40].

Reorganizing stencil calculations to take full advantage of memory hierarchies has been the subject of much investigation over the years. These have principally focused on tiling optimizations [10, 36, 35, 24, 9] that attempt to reorganize stencil calculations to exploit locality and reduce capacity misses; such optimizations are effective when the problem size is larger than the cache's ability to exploit temporal locality. Additionally, investigations have explore the potential of local-store based processor technologies on stencil simulations [46] as much of the memory traffic could effectively be hidden via scratchpad double buffering. We have also recently studied the impact of co-tuning strategies in which traditional architecture space exploration is tightly coupled with software auto-tuning for delivering substantial improvements in area and energy efficiency [30]. Recent work in architectural specialization include the Anton molecular dynamics platform that achieved two orders magnitude improvement over existing supercomputing systems at a fraction of the power requirements [38]. Finally, our group has conducted a detailed exploration of Green Flash, a manycore processor design for high-performance systems based on embedded computing low-power architectures, specifically targeted for ultra-high resolution climate simulations [45, 11]. Green Wave and Green Flash are different parameterizations of the same 128-core, local store augmented architecture. Specifically, Green Flash used a generic memory subsystem capable of 50GB/s (rather than quad-channel

DDR3), used 128KB local stores (instead of 256KB), used scalar XTensa cores (instead of VLIW XTensa), and lacked any instruction extensions for address calculations. In this work, the methodology of coupling applications, algorithms, and hardware is applied in the context of seismic modeling and produces the Green Wave design.

3. EVALUATED PLATFORMS

In this section we detail the hardware architecture evaluated in our study. To ensure a fair comparison, we include among the most modern CPUs (Intel's Xeon E5530 Nehalem) and GPUs (NVIDIA's Tesla C2050), and examine highly optimized wave equation implementations on each platform. Performance and energy efficiency is compared against Green Wave: our manycore design that trades high peak flop rates for vastly increased on-chip capacity to maximize on-chip locality. Key features of the three architectural approaches as shown in Table 2.

3.1 Intel Xeon X5530 (Nehalem)

The Intel "Nehalem" X5530 CPU is the built on Intel's "Core" architecture. The architecture, reminiscent of AMD's Opteron processors, integrates memory controllers on-chip and implements a QuickPath Interconnect (QPI) inter-chip network similar to AMD's HyperTransport (HT). QPI provides access to remote memory controllers and I/O devices, while also maintaining cache coherency. Although Nehalem offers two-way simultaneous multithreading (SMT) and TurboMode, both were disabled on our test machines.

The evaluated system is a dual-socket, quad-core 2.40 GHz Xeon X5530 with a total of 16 hardware thread contexts. Each core has a private 32 KB L1 and a 256 KB L2 cache, and each socket instantiates a shared 8 MB L3 cache. Each socket integrates three DDR3 memory controllers operating at 1066 MHz, providing a theoretical DRAM pin bandwidth of 25.6 GB/s to each socket. However, in practice bandwidth is often less than 19 GB/s.

3.2 Fermi

GPUs have recently gained adoption in many non-graphics related fields, including seismic computing through NVIDIA's Compute Unified Device Architecture (CUDA) programming model. GPUs simultaneously execute many threads per core, hiding latency by switching between numerous concurrent threads. Threads are grouped into programmer-defined thread blocks, where threads within a thread block can synchronize and communicate via shared memory.

In this paper we examine NVIDIA's Fermi-based Tesla C2050, a high performance computing (HPC) oriented GPU. The C2050 consists of 14 Streaming Multiprocessors (SMs), 768 KB of L2 cache, and 3 GB of GDDR5 global memory. Each SM consists of 32 cores operating at 1.15 GHz, 32K 32-bit registers, and 64 KB of memory that can be configured in a 1:3 ratio as either shared memory or L1 cache. Aggregate shared memory (local store) bandwidth across the C2050 GPU is 1.03 TB/s, whereas global memory theoretical bandwidth is 144 GB/s and is accessible by all threads as well as host CPU. In our experiments we use the GPU with ECC (error correction code) enabled, which somewhat reduces global memory bandwidth. The L2 cache is coherent across all SMs, whereas L1 caches are not. Since GPUs are accelerator cards they communicate through interconnects like PCIExpress and incur the respective host transfer time. The grid sizes in all our experiments were sufficiently small that they fit in the 3GB of GPU memory. As such, the limited PCIe bandwidth is amortized by the relatively small ghost zone exchanges.

Core Architecture	**Intel Nehalem**	**NVIDIA GF100**	**Tensilica LX2**
Type	superscalar out-of-order SIMD	dual-warp in-order SIMT	VLIW in-order custom
Clock (GHz)	2.40	1.15	1.00
SP GFlop/s	19.2	73.6	2.00
L1 Data $	32 KB	16 KB	8 KB
L2 Data $/LS	256 KB	48 KB	256 KB
SMP Architecture	**Xeon E5530 (Gainestown)**	**Tesla C2050 (Fermi)**	**Green Wave**
Threads/core	2	48 (max)	1
Cores/socket	4	$14^{\dagger}$	128
Sockets/SMP	2	1	1
Shared Last $	8 MB/socket	768 KB	—
memory parallelism	HW prefetch	Multithreading	DMA
On-chip RAM	18.3 MB	3.4 MB	32 MB
DRAM Pin GB/s	51.2	144 (no ecc)	51.2
SP GFlop/s	153.6	1030.4	256
Power under RTM load	298W	390W (System) 214W (GPU-only)	$66\text{W}^{\ddagger}$
Die Area	263mm^2	576mm^2	294mm^2
Process	45nm	40nm	45nm

Table 2: Details of the evaluated architectures. †We call each shared multiprocessor on a GPU a "core". All bandwidths and flop rates are peak theoretical. ‡All power is measured using an inline meter except Green Wave which is derived via modeling tools.

3.3 Green Wave Architecture

The Green Wave architecture is optimized for energy efficiency (sustained flops per Watt) for stencil-based codes. Our approach is motivated by the stagnation of CPU clock rates, which is driving future performance to be extracted from explicit parallelism. In combination with the increasing cost of data movement, large arrays of simple, low-power cores are set to offer the best performance per Watt and greatest scalability [2]. Additionally, it is generally agreed that application specificity offers greater efficiency. Thus the goal of our application-driven co-design Green Wave approach, is to enable full programmability while allowing greater computational efficiency than general-purpose processors and even GPUs by offering custom ISA extensions, and optimally sizing software managed memories and the on-chip network interconnect. This semi-custom design approach offers the advantage of serving the needs of a broad variety of scientific codes in contrast to full custom designs.

The lowest-level building blocks of the Green Wave design are pre-verified IP components from the embedded space. We then layer novel processor extensions and communication services for greater performance and efficiency. This approach minimizes the amount of design custom logic, which in turn reduces verification costs and design uncertainty. Extensive exploration of our Green Flash climate-simulation design have shown this to be an effective approach [45, 11].

Our hardware architecture is built upon the highly energy-efficient Tensilica LX2 core [41], a single-issue in-order instruction processor combined with a floating point unit that can be customized in several dimensions. Tensilica's Xtensa Processor Generator (XPG) toolchain enables rapid prototyping of custom microprocessor cores. We assume a 45nm chip lithography technology to be consistent with the tech-

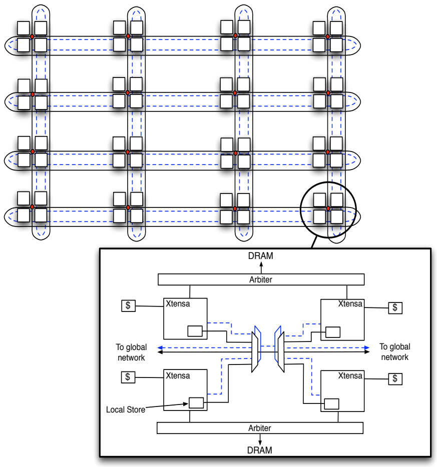

Figure 2: Green Wave CMP Architecture - Lightweight Tensilica LX2 embedded cores are interconnected a scalable 2D concentrated torus on-chip communication fabric for peripheral devices such as memory controllers and off-chip IO.

nology scale used for the other processors in this study (Table 2). At 45nm, the LX2 can achieve a clock rate of 1GHz The XPG tool, allows the addition of new instructions to the base LX2 ISA (Instruction Set Architecture), as well as additional memory and interprocessor network interfaces. In this work, we modify the standard 80-instruction ISA included on each LX2 to include custom designed instructions and extensions specifically geared to accelerate stencil-based RTM modeling(Section 5.3). In addition, we customize the size and configuration of the memory hierarchy — defining data cache sizes and even local stores (software controlled memories) to fit the RTM problem requirements as well as give an overview over the full system design. This study focuses on single-node performance.

The XPG tool outputs both synthesizable RTL (register transfer level) that can be used to create masks for a full chip design, or target an FPGA platform for cycle-accurate modeling of the target core design. In addition, XPG automatically generates C/C++ compilers, debuggers, and functional models that facilitate rapid software porting and testing of each new architectural variant. This environment for rapid prototyping and cycle-accurate emulation environment is central to our hardware/software co-design process.

The Green Wave on-chip interprocessor fabric is derived from the Green Flash design [45, 11]. The lightweight cores are interconnected using a scalable 4-way concentrated torus topology Network-on-Chip (NoC) shown in Figure 2, which is parameterized allowing networks of different performance and scale. We adopted this topology based on our recent cycle-accurate NoC studies [19, 18], which have shown that this approach provides the most energy-efficient solution for problems — such as stencil-based RTM code — where the communication pattern is predominantly nearest neighbor. For this design study, the interprocessor communication is used primarily for communication between cores and memory controllers (for data load/store) and to facilitate energy-efficient halo (ghost-zone) exchanges between the cores to further reduce memory bandwidth requirements by eliminating redundant loads associated with high-order stencils.

4. GREEN WAVE MODELING

A combination of both software-based tools and hardware emulation techniques are used to predict the performance and power of the Green Wave architecture. Cycle accurate software models from Tensilica provide us with a flexible, pre-built simulation solution for initial architectural exploration. In addition, software-based power models provide dynamic power estimations while executing the RTM application. Network simulation environments, such as PhoenixSim provide the ability to model on-chip communication. Finally, hardware emulation techniques provide us with very fast, real world performance predictions based directly on the RTL simulations of the processor These numbers give confidence that our theoretical chip design, complete with custom ISA extensions, is physically realizable. Together these tools make up a preliminary version of the co-design for exascale (CoDEx) simulation environment [37].

4.1 Modeling Chip Power

Green Wave power estimation is created by combining the output of several specialized models. First, energy for events originating in the cores is calculated using the energy estimates provided by the industrial-strength Tensilica tools [41]. These estimates are created from feedback given to Tensilica from customers who fabricated their processors then measured the actual power consumption. Second, the dynamic energy for the caches and local stores is modeled on a per transaction basis using CACTI5 [42] modeling. Third, on-chip network energy is calculated by starting with the total on-chip network communication requirements and then scaling the energy numbers from recent studies [23] for our target process technology. The NoC traffic patterns for halo exchange and associated power are modeled in detail using PhoenixSim, in collaboration with the Columbia University [19, 18], and showed that a concentrated torus as the most energy-efficient topology for this class of applications. The network simulation use router and wire power costs derived from Dally and Balfour's study on electronic NoC modeling [4]. Leakage power is assumed to be 20% of peak power consumed by the processor, on-chip memory and network for any configuration. Finally, we model DRAM energy using the current profiles from the Micron datasheets [29] for a 1 Gb DDR3-1600 memory module and refine the model for performance and power consumption using the cycle accurate DRAMsim2 memory architectural simulator [44] to analyze memory access traces collected through simulation of the Green Wave core executing the the wave propagation kernel. A breakdown of Green Wave power components is shown in Figure 3(left). Given the low-power nature of the Tensilica cores, observe DRAM power constitutes roughly half of the node's total power. A Green Wave System on Chip (SoC) design would also include an on-board Infiniband (IB) 4x QDR interface, but we do not currently include the power or area for this interface in our model. Future work will also take into account the power consumption from an on-board IB interface, but estimates based on existing examples suggest it will consume area similar to that of the DDR-3 memory interface and between half to one quarter the power depending on drive strength requirements.

4.2 Modeling Chip Area

The area of a given processor configuration is an important metric due to its effect on the end cost of fabricating and

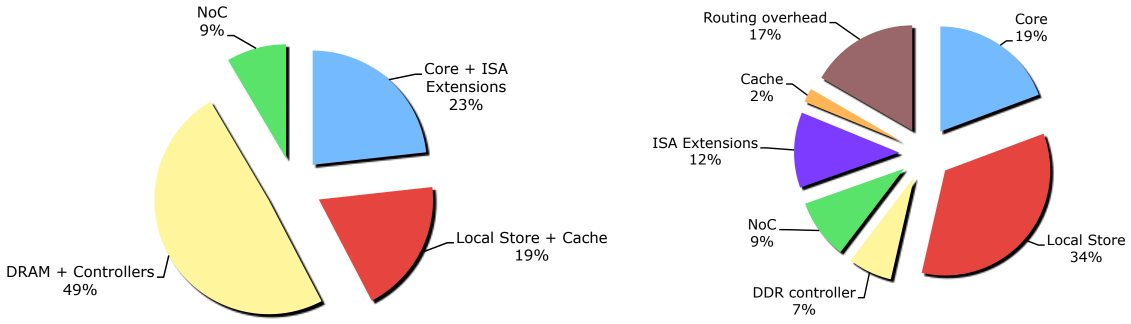

Figure 3: Green Wave Component Breakdown, showing each component contribution of (left) total node power consumption and (right) total die area.

packaging the ASIC design. To this end, we model the hardware configuration area within the design space, assuming 45nm chip lithography technology to ensure a fair comparison against the GPU and Nehalem's that use the a similar feature size. For estimates of the XTensa core area, the Tensilica toolchain provides direct estimates for a 65nm design, which are then reprojected to 45nm based on standard design scaling rules. For our custom processor extensions the Tensilica tools provide area measurement in terms of gate count. As the gate count of the total processor is also provided, it is straightforward to extend the area estimate of the Tensilica tools by the associated instruction extensions overhead. CACTI5 [42] is used to model cache and local store area. There is a fixed cost for routing and clocking that add another 20% to the core chip area. For NoC area estimations, we again used the cost models proposed by Dally and Balfour [4]. Finally, the quad-channel memory interface adds 20 mm^2 to the chip area regardless of the frequency we clock the DIMM. This area estimate is consistent with the specifications for Denali DDR3 memory controller IP blocks from Cadence Inc. [7] together with Silicon Creations [39] Programmable Phase Locked Loop (PLL) for the physical interface. Given each XTensa core is only 1 mm^2 and each 256 KB local store is less than 2 mm^2, the area consumed by memory controllers is quite substantial. As such, there is a clear economy of scale by incorporating many cores to share the memory controller resources. The result is a 294 mm^2 45nm chip, making Green Wave comparable in die size (and thus manufacturing cost) to a Nehalem processor, and substantially smaller than the C2050 GPU that weighs in at a hefty 576 mm^2 as shown in Table 2. Figure 3(right) presents a breakdown of Green Wave area components.

4.3 Modeling Chip Performance

Previous performance modeling of local store (LS) architectures [46] have shown that communication (DRAM–LS) can straightforwardly be decoupled from compute (LS–FPU) on double buffered stencil codes. This allows bound and bottleneck analysis to accurately determine performance. A similar methodology is applied for Green Wave. To model the required time of direct memory access (DMA) data between DRAM and the local store, the thread's ideal block size must first be computed. This is done via a search that determines the block dimension providing the best ratio of stencil area to the requisite DRAM communication that fits in the local store (after double buffering). The time required to transfer such blocks is the ratio of bytes to bandwidth per core. This overhead includes an overfetch of each block due to the stencil halos, as shown in the last row of Table 1.

Next, we calculate the time required to perform the requisite stencils once a block has been transferred into the local store. Here we use the XPG toolchain to generate a cycle-accurate software model of the configured processor, including any custom hardware extensions added to accelerate the RTM stencils. This model uses the XTensa Instruction Set Simulator (ISS) to provide the number of cycles required to execute a given code assuming the data resides in the local store. Finally, to calculate execution time per core using double buffering, we compute the maximum between communication and computation. Given this upper bound allows us to determine overall performance based on the number of cores, stencils per core, and flops per stencil.

4.4 RAMP Simulation

The XPG tools used to design a new processor core can generate a complete RTL gate-list for the target design that can target an ASIC design flow or can be uploaded to an FPGA for fast, cycle-accurate emulation of the target chip design. Our study utilizes the Research Accelerator for Multi-Processors (RAMP) [6] as the FPGA emulation platform. The direct mapping to FPGAs on the RAMP hardware emulation platform and copious performance data provide a fast, accurate emulation environment allowing the benchmarking of real codes ensuring the application developers are intimately involved in the hardware/software codesign process. Our current emulation environment provides direct emulation support of four Green Wave cores complete with appropriately sized local stores, instruction extensions, etc., providing a realistic experimental platform.

Although the software simulation environment is unable to run problems with a volume larger than 128^3 per-core due to the limited amount of memory, 256MB, the simulator is able to allocate. By contrast the FPGA-based platform available in the CoDEx simulation flow allows collection of performance data on problems of more realistic size because each emulated core has access to several gigabytes of data. Further, the performance of the FPGA environment does not degrade as the number of emulated cores increase allowing us to model the performance of multiple cores with little penalty. The emulation is performed directly on the gate-level RTL mirroring the physical design that nominally would be used for place-and-route, mask generation, and chip fabrication. The fact that the emulated logic is precisely the actual circuit design for the target chip provides a superior level of confidence in our software-based methods as we can constantly verify our software simulation results against those of the exact model of the hardware platform. By creating a software simulation environment that mimics our hardware emulation environment we are able to calibrate our software-based methods on four cores giving confidence in the results when we scale to 128 cores using models.

5. PERFORMANCE OPTIMIZATION

In this section we discuss wave equation performance optimization, starting with an existing reference implementation. The optimizations can be divided into software modifications, and in the case of Green Wave, configuration and specialization of the Tensilica processor for the demands of RTM. Broadly speaking, the goals of software optimization (improving locality, maximizing parallelism, minimizing memory traffic) are ubiquitous and independent of the underlying architecture. However, the implementation is dependent on both the architecture and programming model.

5.1 CPU Software Optimization

Our previous stencil optimization work has demonstrated some of the fastest multicore implementations in the literature [10, 9, 46]. In this paper, we modify these techniques for the particular requirements of RTM. Foremost, we optimize the high-order laplacian stencil at the core of the wave equation. First, the reference implementation included an inner loop for generalized order. We manually unroll several variants and select the appropriate at runtime. Second, we merge cache blocking and thread parallelization to minimize the intra-thread cache working set whilst simultaneously maximizing the inter-thread overlap of working sets. On Green Wave, we extend this technique with DMA-filled multibuffering to maximize utilization of its on-chip local stores. Third, we manually unroll and SIMDize the C code (including cache bypass) to express the requisite data- and instruction-level parallelism as well as further optimize the SIMD implementation ("register pipeline") to minimize L1 accesses. Green Wave's selection of scalar or VLIW cores and use of a local store obviates the need for SIMD and write allocate optimizations respectively. Finally, as the Nehalem system is a dual-processor server, we exploit thread pinning a first touch policy to ensure proper NUMA allocation and to avoid under utilizing the memory controllers and over taxing the QuickPath Interconnect. The combination of these optimization techniques resulted in significant performance improvements, achieving almost a 40× speedup comparing the sequential single-core version to the fully optimized kernel running on 8 Nehalem cores. Detailed performance results are presented in Section 6.

5.2 GPU Software Optimization

Performance optimization of high-order stencils can be particularly challenging on GPUs due to the complexities of properly exploiting CUDA shared memory, SIMT execution, and memory coalescing. To mitigate these challenges and provide the highest possible performance, we used the fastest implementation of the high-order stencils available [27, 28] and ran them on our Fermi accelerated Nehalem cluster. Broadly speaking, the GPU implementation uses a multi-buffering scheme similar to that used on the Green Wave design. However, the GPUs hide memory latency via massive multithreading rather than DMA. Due to the limited DRAM capacity, subdomains were limited to 512^3. While the GPU design space was fixed (thus under-utilizing various aspects), our co-design methodology allows us to find the appropriate balance between memory capacity, memory bandwidth, per-core memory and per-core compute.

5.3 Green Wave Optimization

Before starting the iterative co-design process, we selected a number of fixed design choices that were adopted as boundary conditions for our design study. First, we chose to use a commodity quad-channel DDR3-1600 memory subsystem, which presents a low-risk design point from the standpoint of practical ASIC packaging and power dissipation, and reflects the memory performance of existing mainstream products, such as the Nehalem. The choice of conventional memory also simplifies the power model because the DDR components have a well characterized power profile. The second baseline design choice was to target a 45nm process to be consistent with the other chip lithographies used in our study. That choice bounds our target clock frequency to 1 GHz, and clearly defines the parameters for the power and area model of the our design. Finally, Green Wave utilizes a (software controlled) local store architecture as the primary on-chip cache memory, as previous studies on the Cell have demonstrated the substantial efficiency benefits that can be derived from that approach [46] — particularly for stencil computations. A small L1 data cache remains for each processor in order to support convenient code porting, but the local store is the primary approach for latency hiding and capturing temporal recurrences in the RTM algorithm.

5.3.1 Local-Store Size and Core Count

The first hardware optimization was selecting the on-chip memory size so as to capture the maximum number of temporal recurrences for the high-order stencil kernel. In Section 2.2 we calculated an average memory traffic per stencil lower bound of 17.6 bytes for the 8^{th} order wave equation using 64×32 cache blocks. Moreover, as our implementation mandates such a kernel keep a working set of 12 working and 4 buffered planes (some with halos) in the local store, the wave equation would require a 166 KB of local store for this decomposition. As one moves to the 12^{th} order wave equation, the memory requirements increase to 238 KB per core. Once Green Wave is configured with sufficient on-chip memory to capture all recurrences, we then determined the number of processors required to saturate the off-chip memory bandwidth. Note that an iterative optimization process is necessary, as changes in the core count requires a resizing of the local-stores (to incorporate the halos from the blocked implementation), which in-turn impacts the optimal core count (to effective capture temporal recurrences). This analysis resulted in a design choice of approximately 100 processor cores and 238 KB of local-store per core. As a conservative estimate, both of these figures were rounded up to the nearest power-of-two multiple (256 KB local store and 128 processor cores) to simplify the layout of the SRAM mats on chip as well as the NoC topology.

5.3.2 VLIW Extensions

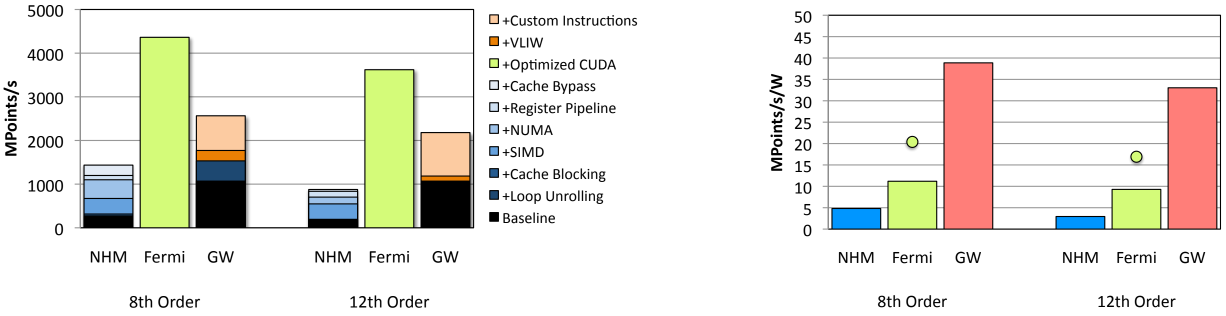

Figure 4: Wave Equation Performance for 8^{th} and 12^{th} order on 512^3 grid. (left) Raw performance. (right) Energy efficiency, with circle representing solely GPU power (host power ignored).

The Tensilica design flow enables us to add hardware optimizations specifically tailored to RTM-based stencil computations to software optimization techniques. Because the Green Wave architecture includes a correctly sized (256 KB) local store for capturing all temporal recurrence of data, our performance model only requires a fixed latency model for memory accesses, and focuses the optimization effort on reducing instruction count. Therefore our co-design approach leverages the Tensilica compiler's ability to bundle instructions into VLIW, allowing for co-issue of instructions. Because RTM computations are floating-point intensive, the base LX2 processor was configured to support maximum instruction dispatch width of 64 bit and data Load/Store width of 128 bits, allowing multiple floating point instructions to be concurrently issued. The Xtensa compiler automatically bundles opcodes depending on the designers specifications of what opcodes are legal in which slots. From a hardware perspective, the processor generator tool creates parallel pipelines capable of executing the various instructions in each slot. This is a simple and effective optimization that requires no code changes while providing a potentially significant performance boost.

5.3.3 *Custom Instruction Design*

Another approach towards reducing instruction count is the creation of custom instructions that allow the "fusing" of commonly used operations. Here we can leverage a key feature in the Tensilica LX2 design flow that allows the creation of custom instructions and data types [41]. These instructions are written in a language similar to Verilog that Tensilica calls via the TIE interface. The custom instructions are fully supported by the Xtensa compiler and become native intrinsics for use in software development and performance modeling. Our first custom allows the computation of Y and Z loop indices for a given stride to be performed concurrently. These indices are stored in special registers for later use as offsets into the data array. The original code version would calculate these offsets individually, then do pointer arithmetic to fetch the correct data point from the array. By pre-computing these values via the custom instructions the user can pass a pointer to the start of the array, then select the direction (Y or Z) as well as the offset (1 through 8) of the desired value. This instruction fetches the pre-computed offset from the register, calculate the address, feeds this new address to the processor's load/store unit and then return the value — essentially collapsing two instructions into a single fast array index operation.

Next, because the LX2 has a limited number (16) of floating point registers, a second 128-bit register file was created to allow space for more temporaries and an efficient path for data exchanges between loops iterations. The register file can be accessed in non-traditional ways, such as rotating a 32-bit float in or out of an individual register, or register loading with four 32-bit values from the most or least significant bit. Note that the TIE language does have limitations: while the TIE compiler auto multi-porting of register files to support multiple reads per instruction, each instruction is limited to one write per register file. Thus allowing for a variety gather operations, but no scatter support.

The novelty in this approach is not the (relatively straightforward) custom instructions taken in isolation, but rather the contribution of these instructions in the context of a co-design methodology where only the specific functionality needed to efficiently solve a problem is added to the hardware. Although many of these features are available in other existing architectures, we are able to provide only the subset that improves performance — thus maximizing energy efficiency while maintaining programmability. In addition, the general-purpose nature of these instructions allows them to be applied to other stencil-based computations, allowing the Green Wave solution to be applicable to a wide-variety of high-order methods. Finally, these custom instructions allow high performance with a simpler programming methodology than Intel intrinsics or CUDA, as detailed in Section 6.

6. PERFORMANCE ANALYSIS

In this section, we discuss the performance and energy efficiencies of Nehalem, Fermi, and Green Wave in the context of our optimized implementation. All experiments are conducted in single-precision as described in Section 2 using a 512^3 grid, with a memory footprint of about 2GB.

6.1 Performance and Energy Efficiency

Figure 4(left) presents the performance of our three machines running the (spatially) 8^{th} and 12^{th} order wave equation on a 512^3 grid. Recall that the wave equation behavior acts as an effective proxy for the more complex full RTM problem. The Fermi accelerated node outperforms both Nehalem and Green Wave, attaining an advantage ranging from 3–4× and 1.6–1.7× respectively. Our Green Wave performance modeling actually shows that bandwidth reduces the potential performance of the 8^{th} order stencil by 26%. For the 12^{th} order stencil Green Wave is perfectly balanced between per-core performance and memory bandwidth. In higher order implementations, the compulsory memory traffic remains roughly constant (4% increase), while the arith-

Architecture	Nehalem	Fermi	Green Wave
Total Nodes	127,740	66,823	75,968
MPoints/Watt	4.27	6.28	32.63
Time in Communication	9.5%	43%	16%
Total MWatts	**38.2**	**26.1**	**5.0**

Table 3: Extrapolated cluster configuration, sustained energy efficiency and power consumption to complete 8^{th} order forward and backward model of 30k × 20k × 10k survey with 12,000 timesteps in one week. Subclusters consist of 256 nodes each with a 512^3 subdomains. Power estimates based on aggregate node-level requirements and do not include interconnect, external storage, or cooling.

metic intensity (as well as flops per point) increases linearly with order. Therefore, when the calculation is not bandwidth bound, there is a slight degradation in per-core performance. Overall, these results suggest that we have selected the right balance between the number of cores, local store capacity, and bandwidth given the spectrum of explicit numerical methods seen in RTM.

Once power considerations are taken into account, Green Wave shows a 8× and 3.5× energy efficiency advantage over Nehalem and Fermi respectively. If host power on the Fermi-accelerated system is ignored, Green Wave's efficiency advantage is reduced to about 2× as shown by the small green circle above the fermi bar in Figure 4. Green Wave would require at least six DDR3 memory controllers to match Fermi's performance. Although this would increase node power by roughly 25% and increase energy efficiency by a further 20%, four controllers is comparable to existing commodity products such as the Xeon and Opteron processors. Using a more conservative design choice for memory technology, we demonstrate that the Green Wave approach is well suited towards the parallel nature of RTM stencil calculation and can utilize hardware customization to maximize energy efficiency whilst maintaining general programmability. Additionally, our approach enables an extremely power-efficient solution with minimal software optimization.

6.2 Projected System Scale Efficiency

We now return to the large-scale survey analysis problem described in Section 2.3 that performs 8^{th} order forward and backward modeling for a 30k × 20k × 10k survey with 12,000 timesteps and a total of 120,000 independent shots within one week. The RTM calculation for each shot requires a 4096×4096×2048 volume that is domain decomposed into 512^3 subdomains per node in a 8 ×8×4 processor grid (256 nodes per shot). These groups of 256 nodes are arranged as tightly-coupled sub-cluster that is connected to an Infiniband 4x QDR leaf switches that uplink to a global system fabric that tapers bandwidth at the upper switch tiers as there is comparatively less communication between sub-clusters. A tapered CLOS network of sub-clusters can easily scale up the total number of sub-clusters to reach the target throughput of 120,000 shots per week.

In order to quantify communication time, collected MPI performance data (including buffer packing, and PCIe transfer time) on the NERSC Dirac Nehalem/GPU cluster running 512^3 problems using IPM. Table 3 presents the extrapolated node configuration, energy efficiency and power consumption for the three evaluated technology solutions. Note that the power estimates are based on a detailed analysis within the node, but do not include power requirements of interconnect, external storage, cooling and building. Future work will include a system scale-model that includes these additional components. However, we assume the total quantity of I/O and other resources required by the compared cluster solutions would be nearly identical to achieve the same system throughput and storage capacity and therefore constitute a fixed-cost for each solution and are smaller than the projected power consumption of the node and memory.

To qualify our claim that node and memory dominate power consumption, we use power measurements from the NERSC Carver cluster [32], to collect the GPU and CPU performance data, and measurements of similarly configured infiniband x86 clusters [34, 21]. We have estimated the power that needs to be consumed by a disk subsystem that is matched to the throughput requirements of the system accounts for less than 5% of the overall system power based on existing RAID-based storage solutions and a similarly configured scalable Infiniband switch solution based IB 4x QDR would account for less than 5-12% of overall system power contributions. Finally, the NERSC center where the GPU and CPU tests were performed, delivers a Power Usage Effectiveness (PUE) of 1.3, so cooling is not a significant factor. Therefore, the memory and node power captures the dominant contributors to overall system power consumption.

Extrapolating per-node performance, communication, and power requirements, we find that a Nehalem-based cluster would require on the order of 128,000 nodes (arranged in subclusters of 256) and consume approximately 38 MWatts of power for the nodes alone. A GPU-based solution would reduce the requirements to 66,800 nodes but would still require a substantial 26 MWatts. The Green Wave approach requires more nodes (75,000) but only 5 MWatt of power — a 7.6× and 5.2× energy efficiency improvement in this aspect of the system as compared to Nehalem and Fermi (respectively). The advantage of a Green Wave full system design would likely be further increased through passive savings e.g. for cooling and software development. This significant power savings would open the potential for high-quality interactive survey analysis at large scale.

6.3 Analysis of Green Wave Methodology

The co-design methodology used to architect Green Wave demonstrates a clear performance-per-watt advantage over more general-purpose architectures by leveraging application specificity. This trend is evident when observing the increase in performance when moving from larger, more complex cores to arrays of simpler cores. Fermi is similar to Green Wave as it is a collection of relatively simple cores, however, while Fermi is able to achieve a greater raw performance number, it's energy efficiency is burdened with features specific for graphics processing that do not assist with the stencil computation, such as rasterizer, texture sampler, ISA extensions, etc. In contrast, Green Wave begins with an extremely simple, low-power core and relies on the ability to selectivity add only the instructions and features necessary for accelerating stencil computation. In addition to extending the instruction set, thorough application profiling allows us to properly size the Green Wave local stores, caches, and on-chip network to create a more balanced architecture. These customizations do not come at the expense of flexibility or ease of programmability and allow a straightforward mapping from ANSI C to the stencil-specific optimizations. Finally, we note that the potential challenges of correctly utilizing the software managed memories that act as a key component in the superior performance of the Green Wave architecture are not unique to our design.

Our approach relies on off-the-shelf IP to reduce design and verifications costs. Based on a survey of current industry pricing, the IP cost of licensing circuit designs for DDR3 memory controllers, PLL for off-chip drivers, NoC implementations, and customizable cores for Green Flash would be approximately $2.5M. By contrast, a full-custom design for any single component would increase design and verification costs by one to two orders of magnitude. Projected non-recurring expenses (NRE) for design services for design integration, verification and mask generation averaged $2M. Manufacturing costs for the first 10,000 chips at a 45nm fab (including testing and packaging) is estimated to be $200/unit and reduces by 35% for volumes greater than 50,000 units. Given this cost model, the important lesson is that using this design methodology we can move beyond thinking of the chip as the commodity. Rather, one should think of the Intellectual Property that you put onto the chip as the commodity. We believe this design philosophy, combined with a co-design methodology, could transform the way we design supercomputing systems in the future.

7. CONCLUSION AND FUTURE WORK

The computational demands for RTM simulations for seismic analysis have risen to enormous levels due to ever rising survey sizes and the need for ever increasing image quality. Unfortunately, the cost and performance of supercomputing systems used to address these problem are increasingly dominated by power bills that are set to exceed the cost of hardware acquisition. The urgent need to improve performance and simulation quality at a fixed power budget has pushed industry to evaluate alternative computing platforms. In this work we have compared competing hardware approaches for energy-efficient seismic modeling, which include modern CPU, GPU-accelerated computing, and solution composed of an array of lightweight embedded cores that we call *Green Wave.* Green Wave embodies a hardware/software co-design design methodology that focuses on using agile design synthesis tools and cycle-accurate simulation from the embedded design space to optimize the tradeoff between performance and power requirements while maintaining general-purpose programmability.

The Green Wave architecture augments baseline low-power embedded processors with layered hardware optimizations, including optimized local store size, core count, VLIW extensions, and register file configuration. Additionally, we reduced CPI (cycles per instruction) by creating custom instructions that allow the fusion of commonly used operations in PDE solver address calculations. These new instructions are fully supported by the XTensa compiler and become native intrinsics for use in software development and performance modeling, and are broadly applicable to a range of PDE solver problems outside of RTM kernel we investigated in this paper. Detailed modeling of Green Wave's power consumption using Tensilica's power estimation tools and performance using the FPGA-accelerated cycle accurate played an essential role in the codesign and performance evaluation process. Unlike many hardware simulation environments, the power and timing models for this emulation environment have been verified by comparison against similar taped-out ASIC designs from a broad range of embedded applications. Green Wave comparisons with fully-optimized CPU and GPU code version showed that, although raw performance was slower than Fermi by up to 1.5×, Green Wave outperformed the Nehalem and Fermi energy efficiency by more than 8× and 3.5× respectively for node to node comparison and 7.6× and 5.2× for system level. Although the peak Flop/s of Intel's latest Xeon-E3 processors (Sandy Bridge) is more than double that of Nehalem, they have no more bandwidth. As such, their sustained energy efficiency on bandwidth-bound, high-order wave equations will be comparable to today's Nehalems. Furthermore, the energy efficiency of future designs (Intel, NVidia, and Green Wave) *all* benefit from improved chip lithography and memory technology, so efficiencies demonstrated for Green Wave can reasonably continue for future die shrinks. Thus, even at this early development stage, these results highlight the long-term potential impact of our design methodology.

It is important to note that, although memory organization and locality is critical for all three evaluated architectures, tuning for Green Wave is arguably no more complex compared with Nehalem and Fermi. We highlight that a relatively short amount of effort was required to modify the simple, general-purpose embedded core to one that is enhanced for stencil computations. This application-targeted core was quickly generated using mostly off-the-shelf IP and represents a significantly easier validation target than a full custom design — a testament to the sophistication and flexibility that is available in modern embedded design tools.

In future work, we will refine our node models by integrating a more flexible cycle accurate network-on-chip simulation that we have studied in previous work with Cornell [20, 18] with the node simulation infrastructure. This combined simulation platform will enable us to explore the energy efficiency and performance of alternative on-chip NoC options and their interaction with novel extensions to the Tensilica instruction set for efficient on-chip communication. Additionally, we will build on our efforts to improve the robustness of the Green Wave architecture to address a broader range of full RTM code requirements. This includes refining our performance modeling of the full system configuration in terms of external storage configuration, as well as addressing upcoming algorithmic challenges such as implementing RTM using the full elastic wave equation. Lastly, we will further improve our model with measurements of Infiniband power consumption for on-chip interfaces for the SoC and the network switch infrastructure. Our current measurements of IB hardware indicates that it is a comparatively small contribution to the overall power consumption of the system, but will present results after we have more complete data. Across this increased design space, we will search for a pareto-optimal design point that ensures robust performance and energy efficiency across the range of RTM requirements.

In summary, we have demonstrated the potential of the co-design methodology to develop and evaluate HPC designs that can offer substantially improved energy efficiency for RTM codes. We believe that our proposed approach holds tremendous promise for the broader landscape of energy-efficient HPC design in the face of skyrocketing computational demands and power-limited environments — for both the seismic industry as well as a broad range of numerically demanding applications classes that are of interest to the worldwide scientific community.

Acknowledgments

All authors from Lawrence Berkeley National Laboratory were supported by the DOE Office of Advanced Scientific Computing Research under contract number DE-AC02-05CH11231. The authors kindly thank Paulius Mickievicius for providing us with the optimized GPU code version and his numerous insightful comments. The authors also thank Chris Rowen and Tensilica Inc. for their advice and support of this project. We also thank Marty Deneroff who has provided a lot of advice and expertise on system design trade-offs and cost models for practical implementation.

References

[1] M. Araya-Polo, F. Rubio, M.Hanzich, R. de la Cruz, J.M. Cela, and D.P. Scarpazza. High-performance seismic acoustic imaging by reverse-time migration on the cell/b.e. architecture. In *ISCA2008*, 2008.

[2] Krste Asanovic, Ras Bodik, Bryan Christopher Catanzaro, Joseph James Gebis, Parry Husbands, Kurt Keutzer, David A. Patterson, William Lester Plishker, John Shalf, Samuel Webb Williams, and Katherine A. Yelick. The landscape of parallel computing research: A view from Berkeley. Technical Report UCB/EECS-2006-183 (http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html), EECS Department, University of California, Berkeley, December 2006.

[3] Satish Balay, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc Web page, 2009. http://www.mcs.anl.gov/petsc.

[4] James Balfour and William J. Dally. Design tradeoffs for tiled cmp on-chip networks. In *ICS '06: Proceedings of the 20th annual international conference on Supercomputing*, pages 187–198, New York, NY, USA, 2006. ACM.

[5] Edip Baysal, Dan D Kosloff, and John W.C. Sherwood. Reverse time migration. In *Geophysics*, volume 48(11), 1983.

[6] Berkeley Wireless Research Center. RAMP Homepage. `http://ramp.eecs.berkeley.edu/`.

[7] Cadence Inc. Denali DDR3 memory controller IPt. Whitepaper, April 2011. `http://www.cadence.com/solutions/dip/memorystorage/ddr_cntrl_ip/Pages/default.aspxl`.

[8] Robert G. Clapp, Haohuan Fu, and Olav Lindtjorn. Selecting the right hardware for reverse time migration. *The Leading Edge*, 29(1), 2010.

[9] Kaushik Datta, Shoaib Kamil, Samuel Williams, Leonid Oliker, John Shalf, and Katherine Yelick. Optimization and performance modeling of stencil computations on modern microprocessors. *SIAM Review*, 51(1):129–159, 2009.

[10] Kaushik Datta, Mark Murphy, Vasily Volkov, Samuel Williams, Jonathan Carter, et al. Stencil Computation Optimization and Auto-Tuning on State-of-the-art Multicore Architectures. In *Proceedings SC '08*, pages 1–12, Piscataway, NJ, USA, 2008. IEEE Press.

[11] D. Donofrio, L. Oliker, J. Shalf, M. Wehner, C. Rowen, J. Krueger, S. Kamil, and M. Mohiyuddin. Energy-efficient computing for extreme-scale science. In *IEEE Computer*, 2009.

[12] James P. Durbano, Fernando E.Ortiz, John R. Humphrey, Mark S. Mirotznik, and Dennis W.Prather. Hardware implementation of a three dimensional finite-difference time-domain algorithm. In *IEEE Antennas and Wireless Propagation Letters*, volume 2, 2003.

[13] Eric Dussaud, William W. Symes, Paul Williamson, Larent Lemaistre, Paul Singer, Bertrand Denel, and Adam Cherrett. Computational strategies for reverse-time migration. In *SEG*, Las Vegas, 2008.

[14] Peter Kogge et al. Exascale computing study: Technology challenges in achieving exascale systems. `http://users.ece.gatech.edu/~mrichard/ExascaleComputingStudyReports/exascale_final_report_100208.pdf`, 2008.

[15] Darren Foltinek, Daniel Eaton, Jeff Mahovsky, Peyman Moghaddam, and Ray McGarry. Industrial-scale reverse time migration on gpu hardware. In *SEG Houston International Exposition*, 2009.

[16] Haohuan Fu, William Osborne, Robert G. Clapp, Oskar-Mencer, and Wayne Luk. Accelerating seismic computations using customized number representations on fpgas. *EURASIP Journal on Embedded Systems*, 2008.

[17] Chuan He, Mi Lu, and Chuanwen Sun. Accelerating seismic migration using fpga-based coprocessor platform. In *IEEE Symposium on Field-Programmable Custom Computing Machines*, 2004.

[18] Gilbert Hendry, Johnnie Chan, Shoaib Kamil, Lenny Oliker, John Shalf, et al. Silicon nanophotonic network-on-chip using tdm arbitration. *High-Performance Interconnects, Symposium on*, 0:88–95, 2010.

[19] Gilbert Hendry, Shoaib Kamil, and Aleksandr Biberman. Analysis of photonic networks for a chip multiprocessor using scientific applications. In *NOCS*, pages 104–113, 2009.

[20] Gilbert Hendry, Shoaib Kamil, Aleksandr Biberman, Johnnie Chan, Benjamin G. Lee, Marghoob Mohiyuddin, Ankit Jain, Keren Bergman, Luca P. Carloni, John Kubiatowicz, Leonid Oliker, and John Shalf. Analysis of photonic networks for a chip multiprocessor using scientific applications. In *NOCS*, pages 104–113, 2009.

[21] Voltaire Inc. Datasheet for voltaire qdr infiniband switch for ibm idataplex, 2011. `http://www.voltaire.com/assets/files/Voltaire_IBM_BC-H_HSSM_datasheet-WEB-070109.pdf`.

[22] Jonathan G Koomey. Worldwide electricity used in data centers. *Environ. Res. Lett.*, 3(034008), 2008.

[23] Jacob Leverich, Hideho Arakida, Alex Solomatnikov, Amin Firoozshahian, Mark Horowitz, and Christos Kozyrakis. Comparing Memory Systems for Chip Multiprocessors. In *International Symposium on Computer Architecture*, 2007.

[24] A. Lim, S. Liao, and M. Lam. Blocking and array contraction across arbitrarily nested loops using affine partitioning. In *Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, June 2001.

[25] Wei Liu and et al. Anisotropic reverse-time migration using co-processors. In *SEG Houston International Exposition*. SEG, 2009.

[26] Paulius Micikevicius. 3D finite difference computation on GPUs using CUDA. In *GPGPU-2: Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units*, 2009.

[27] Paulius Micikevicius. 3d finite difference computation on gpus using cuda. In *GPGPU-2: Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units*, pages 79–84, New York, NY, USA, 2009. ACM.

[28] Paulius Micikevicius. Performance scaling of 3d finite difference computation on gpu clusters. Technical Report NVR-002-2009, NVIDIA, December 2009.

[29] Micron Inc. Calculating Memory System Power for DDR3, June 2010. http://www.micron.com/support/dram/power_calc.html.

[30] M. Mohiyuddin, M. Murphy, S. Williams, L. Oliker, J. Shalf, and J. Wawrzynek. A Case for Hardware/Software Co-Tuning for Power Efficient Scientific Computing. In *Proc. SC'09*, 2009.

[31] E. Motuk, R. Woods, and S Bilbao. Implementation of finite difference schemes for the wave equation on fpga. Technical report, University of Belfast, 2005.

[32] NERSC. Carver/Magellan Web page, 2010. http://www.nersc.gov/systems/carver-ibm-idataplex/.

[33] Francisco Ortigosa, Mauricio Araya-Polo, Felix Rubio, Mauricio Hanzich, Raul de la Cruz, and Jose Maria Cela. Evaluation of 3d rtm on hpc platforms. In Barcelona Supercomputing Center, editor, *SEG*, 2008.

[34] Ramesh Radhakrishnan, Rizwan Ali, and Vishvesh Sahasrabudhe. Evaluating energy efficiency in infiniband-based dell poweredge energy smart clusters. *Dell Power Solutions*, February 2008.

[35] G. Rivera and C. Tseng. Tiling optimizations for 3D scientific computations. In *Proceedings of SC'00*, Dallas, TX, November 2000. Supercomputing 2000.

[36] S. Sellappa and S. Chatterjee. Cache-efficient multigrid algorithms. *International Journal of High Performance Computing Applications*, 18(1):115–133, 2004.

[37] John Shalf, David Donofrio, Curtis Janssen, and Dan Quinlan. CoDEx Web page, 2011. http://www.nersc.gov/projects/CoDEx.

[38] David E. Shaw, Ron O. Dror, John K. Salmon, et al. Millisecond-scale molecular dynamics simulations on anton. In *Proceedings SC'09*, pages 1–11, New York, NY, USA, 2009. ACM.

[39] Silicon Creations Inc. Si Creations Programmable PLL IP product. Whitepaper, April 2008. http://www.siliconcr.com/news.html.

[40] William W. Symes. Reverse time migration with optimal checkpointing. In *SEG*, 2007.

[41] Tensilica Inc. Xtensa Architecture and Performance. Whitepaper, October 2005. http://www.tensilica.com/pdf/xtensa_arch_white_paper.pdf.

[42] Shyamkumar Thoziyoor, Naveen Muralimanohar, Jung Ho Ahn, and Norman P. Jouppi. CACTI 5.1. Technical Report HPL-2008-20, HP Labs, 2008.

[43] Reverse time migration with random boundaries. Reverse time migration with random boundaries. In *SEG*, 2009.

[44] D. Wang, B.Ganesh, N. Tuaycharoen, K. Baynes, A. Jaleel, and B. Jacob. Dramsim: a memory system simulator. *SIGARCH Comput. Archit. News*, 33(4):100–107, 2005.

[45] M. Wehner, L. Oliker, and J. Shalf. Green Flash: Designing an energy efficient climate supercomputer. In *IEEE Spectrum*, 2009.

[46] S. Williams, J. Shalf, L. Oliker, S. Kamil, P. Husbands, and K. Yelick. Scientific Computing Kernels on the Cell Processor. *International Journal of Parallel Programming*, 35(3):263–298, 2007.

[47] Öz Yilmaz. *Seismic Data Analysis*. Society of Exploration Geophysics, 2001.