Methods and Infrastructure in the Era of Accelerator-Centric Architectures

Brandon Reagen Yakun Sophia Shao[§] Sam (Likun) Xi Gu-Yeon Wei David Brooks
Harvard University NVIDIA Research[§]

***Abstract*—Computer architecture today is anything but business as usual, and what is bad for business is often great for science. As Moore's Law continues to unwaveringly march forward, despite the ceasing of Dennard scaling, continued performance gains with each processor generation has become a significant challenge, and requires creative solutions. Namely, the way to continue to scale performance in light of power issues is through hardware specialization. Hardware accelerators promise not only orders of magnitude in performance improvements over general purpose processors, but sport similar energy efficiency gains. However, accelerators are equal parts problem solver as they are creator. The major problem is designing and integrating accelerators into a complex environment within the stringent SoC design cycles. Given that each accelerator has a rich design space and convoluted implications and interactions with the memory system, better mechanisms for studying this new-breed of SoC are needed. To usher in the new era of computer architecture, we have built Aladdin: a high-level accelerator simulator enabling rapid accelerator design. Aladdin was recently extended to operate in conjunction with gem5 to study memory system interactions. In this paper we will recount the operation and utilities of Aladdin and gem5-Aladdin, concluding with a case study of how Aladdin can be used to optimize DNN accelerators.**

I. Introduction

In the era of diminishing returns from technology scaling, hardware acceleration is widely used to improve performance, power, and energy. Accelerators are now an integral component of modern SoCs, targeting a variety of applications including video decoding, image processing, cryptography, and machine learning. The natural evolution of this trend will lead to a growing volume and diversity of customized accelerators in future SoCs, where a comprehensive assessment of potential benefits and trade-offs across the entire system will be critical for system designers. However, current customized architectures only contain a handful of accelerators, as large design space exploration is currently infeasible due to the lack of a fast simulation infrastructure for accelerator-centric systems.

To overcome this problem we built Aladdin: a pre-RTL, power, performance, and area simulator designed to enable rapid design space exploration of accelerator-centric systems [1]. Aladdin takes a high-level descriptions of an algorithm as input and uses a dynamic data dependence graph (DDDG) to represent the accelerator, rather than relying on RTL. Starting with an unconstrained program's DDDG, a natural representation of an accelerator, Aladdin applies optimizations and constraints to manipulate the graph and create a realistic model for a particular accelerator microarchitecture. Our results show that Aladdin can model performance, power, and area within 0.9%, 4.9%, and 6.6%, respectively, compared to accelerators generated using traditional RTL flows. In addition, Aladdin provides these estimates over $100\times$ faster.

Accelerators, like the ones Aladdin models, are often designed in isolation and rely on Direct Memory Access (DMA) interfaces to interact with the system. This modularity significantly simplifies the design and system integration process, which rely on device drivers to handle tasks including data movement and coherency. Fundamentally, all systems should be designed in a way that balances the bandwidth of the memory interface with the amount of compute allocated in the datapath. To study this, we have extended Aladdin to interface with gem5 and introduce gem5-Aladdin [2] to enable research considering SoCs with complex accelerator-system interactions.

With gem5-Aladdin, we find that co-designing accelerators with system-level considerations has two major ramifications in accelerator microarchitecture. First, datapaths should be less aggressively parallel, which results in more balanced designs and improved energy efficiency compared to accelerators designed in isolation. Second, the choice of local memory interfaces (DMA or cache) is highly dependent on the dynamic memory characteristics of the accelerated workload, the system architecture, and the desired power/performance targets. We show that accelerator-system co-design can improve energy-delay-product by up to $7.4\times$ and on average $2.2\times$.

To conclude, we present Minerva [3] to show how Aladdin can be used to conduct accelerator optimization research. Minerva is a 5-stage optimization framework for aggressively reducing the power of DNN accelerators. Central to its operation is Aladdin, which enables fast evaluations of the design spaces and is easily extensible to consider the proposed Minerva optimizations. Of the 5 Minerva stages, three propose aggressive power-reduction techniques including datatype quantization, eliding insignificant computations, and SRAM supply voltage reduction. Minerva reduces the accelerator power by more than $8\times$.

978-1-5090-6389-5/17/$31.00 ©2017 IEEE

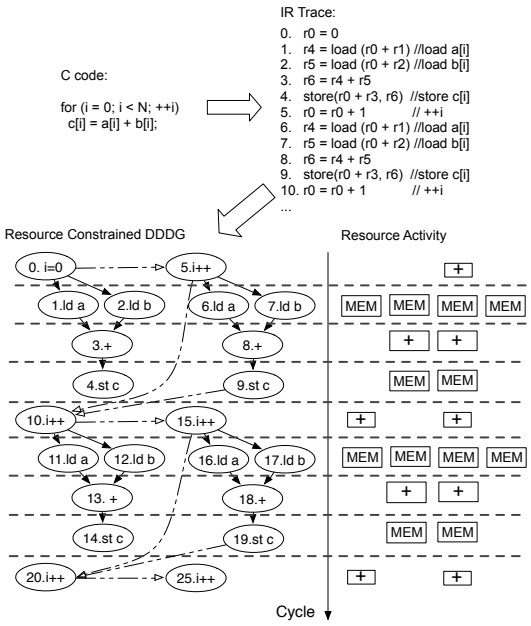

Fig. 1: C, IR, Resource Constrained DDDG, and Activity.

II. The Aladdin Framework

The foundation of the Aladdin infrastructure is the use of a dynamic data dependence graph (DDDG) to represent accelerators. A DDDG is a directed, acyclic graph, where nodes represent computation and edges represent dynamic data dependences between nodes. The dataflow nature of hardware accelerators makes the DDDG a good candidate to model their behavior. Aladdin begins by taking as input a C description of an algorithm, executing the code and generating the Intermediate Representation (IR) trace. The trace is then passed through the *optimization* phase. Here, the DDDG is constructed from the trace and optimized to derive an idealized representation of the algorithm. The idealized DDDG is then passed to the *realization* phase, which restricts the DDDG by applying realistic program dependences and resource constraints. The outcome of these two phases is a pre-RTL, power, performance, and area model for accelerators and is easily integrated with memory hierarchy and interconnect models. For more details see [1].

Figure 1 illustrates different phases of Aladdin transformations using a microbenchmark as an example. After the IR trace of the C code has been produced, the optimization and realization phases generate the resource-constrained DDDG which models accelerator behavior. In this example, we assume the user wants an accelerator with factor-of-2 loop-iteration parallelism, 4 memory ports, and no pipelining. The solid arrows in the DDDG are true data dependences, and the dashed arrows represent resource constraints, such as loop rolling and turning off loop pipelining. The horizontal dashed lines represent clock cycle boundaries. The corresponding resource activities are shown to the right of the DDDG example. We see that the DDDG reflects the dataflow nature

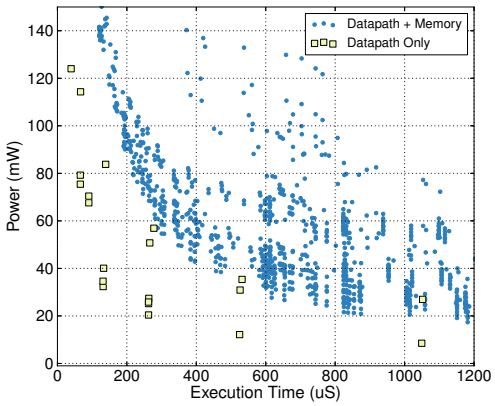

Fig. 2: Resulting design space from sweeping constraints in Aladdin. Point colors denote the whether or not the memory hierarchy is considered.

of the accelerator. Aladdin can accurately capture dynamic behavior of accelerators without having to generate RTL by carefully modeling the opportunities and constraints of the customized datapath in the DDDG.

Figure 2 demonstrates Aladdin's ability to explore the design space. Here, a blocked matrix-matrix multiply code form MachSuite [4] is given as input and various directives including datapath parallelism, memory bandwidth, and pipelining are swept. The figure shows that even for a relatively simple workload, a vast design space exists. With Aladdin, we can quickly identify the Pareto frontier for optimal accelerator microarchitectures and select the one that best suites the target specifications. Note the difference between the blue and yellow points (circles and squares, respectively), as the figure also indicates that a shift occurs when the memory hierarchy is taken into account. When only considering the datapath (i.e., assuming fixed memory latency) we see that the design points are optimistic and that communication and data movement introduce significant overheads.

III. Enabling SoC Simulation

To better understand the complications demonstrated in Figure 2, consider the accelerator execution time breakdown in Figure 3. The data movement overheads consume a substantial amount of overall accelerator execution time. In order to understand how system-level effects impact the behavior of accelerators, we need simulation infrastructures that can model these heterogeneous systems. To address this problem we built gem5-Aladdin [2] by integrating Aladdin with gem5 [5], a widely-used system simulator with configurable CPUs and memory systems.

gem5-Aladdin models interactions and contention between accelerators, CPUs, DMA, hardware-managed caches, and virtual memory. All of these features have implications on how the accelerator behaves. A key contribution of gem5-Aladdin is it enables researchers to understand whether a hardware managed cache or DMA is better on a per-workload basis, as well as to co-design the datapath with the memory

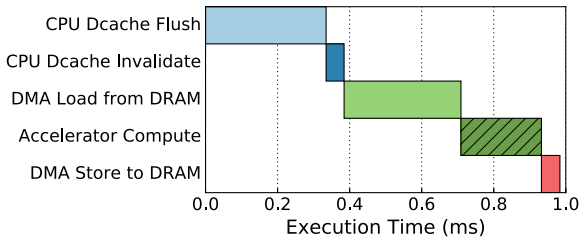

Fig. 3: Breakdown of a MachSuite `md-knn` accelerator's execution time on the Zynq platform.

interface. By integrating Aladdin with gem5, users have two modes to access memory.

DMA Engine: In gem5-Aladdin, accelerators can invoke the DMA engine already present in gem5. To do so, a programmer inserts calls to special `dmaLoad` and `dmaStore` inside the accelerated function with the appropriate source, destination, and size arguments. When the function is traced by Aladdin, it will identify these calls as DMA operations and issue the request to the gem5 DMA engine. As part of the DMA engine, we include an analytical model to account for cache flush and invalidation latency.

Caches and Virtual Memory: For accelerator caches, we use gem5's classic cache model along with a basic MOESI cache coherence protocol. When Aladdin sees that a memory access is mapped to a cache, it sends a request through a cache port to its local cache. Aladdin will receive a callback from the cache hierarchy when the request is completed. To support virtual memory, we implemented a new Aladdin TLB model. To maintain correct memory access behavior, our custom TLB model translates the *trace* address to a simulated *virtual* memory address and then to a simulated *physical* address. TLB misses and page table walks are modeled with a pre-characterized miss penalty.

Figure 4 shows the improvements in EDP when accelerators are co-designed compared to how an accelerator designed in isolation would behave under a more realistic system. On average EDP improves by 1.2×, 2.2×, and 2.0× for accelerators with DMA, caches with 32-bit system bus, and caches with a 64-bit bus, respectively.

The EDP improvements for co-designed cache-based accelerators is higher than that for DMA-based accelerators because an overly aggressive design for a cache-based accelerator results in a large, highly multi-ported cache, which are much more expensive to implement than partitioned scratchpads. Furthermore, we see that on average improvements are greater for cache-based accelerators with a 32-bit system bus than a 64-bit bus.

IV. The Minerva Design Flow

Minerva [3] is a highly automated co-design framework that combines insights and techniques across the algorithm, architecture, and circuit layers, enabling low-power accelerators for executing highly-accurate DNNs. Minerva consists

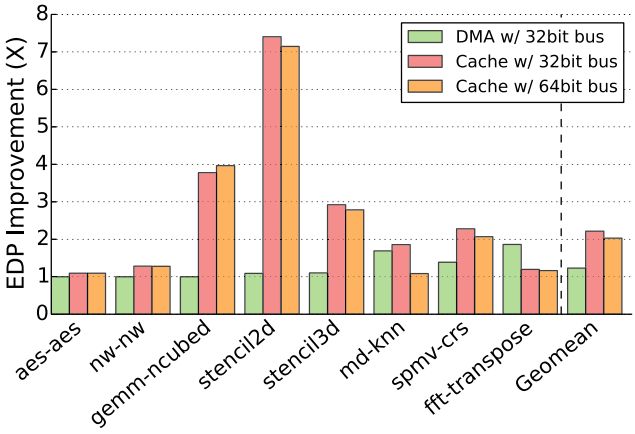

Fig. 4: EDP improvement of co-designed accelerators in different scenarios, normalized to EDP of isolated designs.

of five stages. Stages 1–2 establish a baseline accelerator implementation, and stages 3–5 employ novel co-design optimizations to minimize power consumption over the baseline accelerator design. The organization of the five stages is intended to minimize the possibility of compounding prediction error degradation. Stages include:

Training Space Exploration: Minerva first establishes a DNN baseline that achieves prediction accuracy comparable to state-of-the-art ML results. Of the thousands of uniquely trained DNNs, Minerva selects the network topology that minimizes error with reasonable resource requirements.

Microarchitecture Design Space: This network is fed to the second stage that explores the accelerator design space sweeping microarchitectural parameters in Aladdin. An optimal design point is chosen as the baseline implementation to which all optimizations are compared to.

Data Type Quantization: DNN data types are reduced by independently tuning the range and precision of each signal. Compared to a standard 16 bit fixed-point baseline, and without impacting accuracy, data type quantization reduces power consumption by 1.5×.

Selective Operation Pruning: Analysis of neuron activity values reveals the vast majority of operands are close to zero. Minerva identifies these neuron activities and removes them from the computation such that model accuracy is not affected. Selective pruning further reduces power by 2.0×.

SRAM Fault Mitigation: By combining inherent algorithmic redundancy with low overhead fault mitigation techniques, an additional 2.7× power is saved by aggressively scaling SRAM supply voltages. Minerva employs state-of-the-art circuits to identify potential SRAM read faults and proposes new mitigation techniques based on rounding faulty weights towards zero.

A. A Method for Accelerator Optimization

The Minerva design flow integrates tools at the software, architecture, and circuit levels. Software level analysis lets Minerva quickly evaluate the accuracy impact of optimization

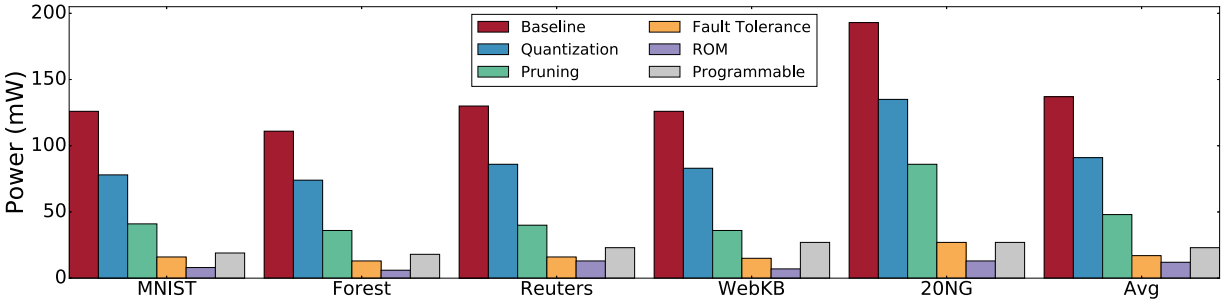

Fig. 5: The Minerva design flow applied to five datasets. The bars labeled ROM show the benefit of full customization (weights are stored in ROMs). Programmable bars show the overhead each incurs when an accelerator is designed to handle all datasets.

trade-offs. Architectural simulation (Aladdin) enables rapid design space exploration and quantification of hardware optimization benefits. Circuit tools provide accurate power-performance-area (PPA) and reliability models, as well as validation of the higher-level simulation results.

Software Level: To understand how each Minerva stage affects prediction accuracy, Minerva uses a software model built on top of Keras.

To evaluate the optimizations, we augment Keras with a software model for each technique. Evaluating fixed-point types was done by building a fixed-point arithmetic emulation library and wrapping native types. For computation pruning, a thresholding operation is added to the activation function of each layer. To study faults in DNN weights, we built a fault injection framework around Keras.

Architecture Level: Stage 2 of Minerva automates a large design space exploration of microarchitectural parameters used for accelerator design in order to settle on a Pareto-optimal starting point for further optimizations. The accelerator design space that we explore is vast, exceeding several thousand points.

To model the optimizations, Aladdin was extended in the following ways: first, fixed-point data types are modeled by adding support for variable length data types. Overheads such as additional comparators associated with operation pruning and SRAM fault detection were modeled by inserting code representative of the overhead into the input C-code. Pruned operations are reported by the software model, which tracks each elided neuronal MAC operation to appropriately estimate dynamic power. To model reduced SRAM voltages, Aladdin's nominal SRAM libraries are replaced with the corresponding low-voltage libraries.

Circuit Level: In order to achieve accurate results, Aladdin requires detailed PPA hardware characterization. PPA characterization libraries are built for all datapath elements with commercial standard cell libraries in 40nm CMOS. For SRAM modeling, we use SPICE simulations with foundry-supplied memory compilers. Fault distributions corresponding to each SRAM voltage reduction step are modeled using Monte Carlo SPICE simulation.

Together, as shown in Figure 5, Minerva reduces power consumption by more than 8× without degrading prediction accuracy. For additional details, see [3].

V. Conclusion

In this paper we argue that to continue scaling performance in light of CMOS power issues future SoCs will become accelerator-centric. To realize high-performance accelerator-centric designs requires new tools which model individual accelerators and reduce design burden, look beyond designing accelerators in isolation and consider their system implications, and better mechanisms to fine-tune and optimize highly utilized applications. In addition to the development of new methodologies, we are also optimistic about the potential of using machine learning to more efficiently search design spaces and identify better designs [6].

Acknowledgments

This work was partially supported by C-FAR, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA. This research was, in part, funded by the U.S. Government under the DARPA CRAFT and PERFECT programs (Contract #: HR0011-13-C-0022). Intel Corporation also provided support. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government or other sponsors.

References

[1] Y. S. Shao, B. Reagen, G.-Y. Wei, and D. Brooks, "Aladdin: A pre-rtl, power-performance accelerator simulator enabling large design space exploration of customized architectures," ISCA, pp. 97–108, 2014.

[2] Y. S. Shao, S. L. Xi, V. Srinivasan, G. Y. Wei, and D. Brooks, "Co-designing accelerators and soc interfaces using gem5-aladdin," MICRO, 2016.

[3] B. Reagen, P. Whatmough, R. Adolf, S. Rama, H. Lee, S. K. Lee, J. M. Hernandez-Lobato, G.-Y. Wei, and D. Brooks, "Minerva: Enabling Low-Power, Highly-Accurate Deep Neural Network Accelerators," in *ISCA*, 2016.

[4] B. Reagen, R. Adolf, Y. S. Shao, G.-Y. Wei, and D. Brooks, "MachSuite: Benchmarks for Accelerator Design and Customized Architectures," in *IISWC*, 2014.

[5] N. L. Binkert, B. M. Beckmann, G. Black, S. K. Reinhardt, A. G. Saidi, A. Basu, J. Hestness, D. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Computer Architecture News*, 2011.

[6] B. Reagen, J. M. Hernández-Lobato, R. Adolf, M. A. Gelbart, P. N. Whatmough, G.-Y. Wei, and D. Brooks, "A case for efficient accelerator design space exploration via bayesian optimization," in *ISLPED*, 2017.