SCALING A RECONFIGURABLE DATAFLOW ACCELERATOR

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Yaqi Zhang
May 2020

Abstract

With the slowdown of Moore's Law, specialized hardware accelerators are gaining tractions for delivering 100-1000x performance improvement over general-purpose processors in a variety of applications domains, such as cloud computing, biocomputing, artificial intelligence, etc. [1, 2, 3]. As the performance scaling in multicores is coming to a limit [4], a new class of accelerators–reconfigurable dataflow architectures (RDAs)–is promising in offering high-throughput and energy-efficient acceleration that keeps up with the performance demand. Instead of dynamically fetching instructions like in traditional processors, RDAs have flexible datapath that can be statically configured to spatially parallelize and pipeline the program across distributed on-chip resources. The pipelined execution model and explicitly-managed scratchpad in RDAs eliminate the performance, area, and energy overhead in dynamic scheduling and conventional memory hierarchy.

To adapt to the compute intensity in modern data-analytic workloads, particularly in the deep learning domain, RDAs are increasing to a scale that was unprecedented before. With an area footprint of 133mm^2 at 28nm, Plasticine is a hierarchical RDA supplying 12.3 TFLOPs of computing power [5]. Prior work has shown an up to 76x performance/watt benefit from Plasticine over a Stradix V FPGA due to an advantage in clock frequency and resource density. The increase in scale introduces new challenges in network-on-chip design to maintain the throughput and energy efficiency of RDAs. Furthermore, targeting and managing RDAs at this scale also require new strategies in mapping, memory management, and flexible control to fully utilize their compute power.

In this work, we focus on two aspects of the software-hardware co-design that impact the usability and scalability of the Plasticine accelerator. Although RDAs are flexible to support a wide range of applications, the biggest challenge that hinders the adoption of these accelerators is the required low-level knowledge in microarchitecture design and hardware constraints in order to efficiently implement a new application. To address this challenge, we introduce a compiler stack–SARA–that raises the programming abstraction of Plasticine to an imperative-style domain-specific language

(DSL) with nested control flow for general spatial architectures. Besides architecture-agnostic, this abstraction contains explicit loop constructs, enabling cross-kernel optimizations that are often not explored when programming RDAs. SARA efficiently translates imperative control constructs to a streaming dataflow graph that scale performance with distributed on-chip resources. By virutalizing resources, SARA systematically handles the physical constraints, hiding the low-level physical limitations from the programmers. To address the scalability challenge with increasing chip sizes in RDAs, we present a comprehensive study on the network-on-chip design space for RDAs [6]. We found that network performance highly correlates to network bandwidth, as supposed to latency, for RDAs with streaming dataflow execution model. Lastly, we show that a static-dynamic hybrid network design can sustain performance in a scalable fashion with high energy efficiency.

Acknowledgements

I would like to thank my mother and the little green men from Mars.

Contents

List of Tables

List of Figures

Chapter 1

Introduction (WIP)

1.1 The Rising of Hardware Acceleration

With the end of Dennard Scaling [7], the amount of performance one can extract from a CPU is reaching a limit. To provide general-purpose flexibility, CPUs spend the majority of resources and energy on overheads, including dynamic-instruction fetching and decoding, branch prediction, and a cache hierarchy, etc., with less than 20% of the energy on the actual computation [8]. Even worse, power wall is limiting the entire multicore family to reach the doubled performance per generation scaling enabled by technology scaling in the past [4].

For this reason, hardware acceleration is emerging in various compute-intensive application domains to provide orders of magnitude acceleration, enabling algorithms that were otherwise infeasible [3, 2, 9, 10]. Examples include widely adopted General-Purpose Graphics Processing Units (GPGPUs) in computational genomics, signal processing, graph processing, and deep learning, etc. [3, 2, 1]. Moreover, many recent efforts are spent on leveraging application domain knowledge in hardware design to enable continued performance scaling while meeting the power budget [11]. As artificial intelligence receiving great success in industry and business, past years have seen a growing interest in machine learning accelerators; these accelerators contain specialized circuits for ML kernels that dramatically improve the compute efficiency [12, 13, 14, 15, 16, 17].

Nonetheless, it is non-trivial to achieve good utilization of these accelerators. While the peak FLOPS of GPU has increased by over 20x in the past 10 years, the achievable FLOPS is not increasing accordingly [18, 19]. The massive threads in GPU require embarrassedly parallel workloads to

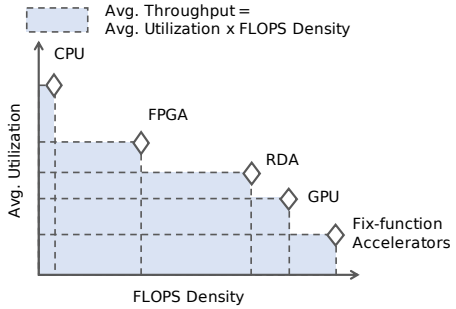


Figure 1.1: Tradeoff between the average utilization of the peak FLOPS over a range of applications vs. the peak compute density among different architectures. The shaded area indicates the effective FLOPS achieved. With all the dynamic scheduling hardware in CPUs, such as prefetching and branch prediction, CPU can achieve a fairly decent instruction per cycle (IPC) for data-analytic workloads that have a regular control flow. However, overheads to support flexibility, security, and programmability in CPU result in a low FLOPS density. While GPU and fix-function accelerators have a high FLOPS density, they are prone to underutilization due to variation in application characteristics. Fine-grained reconfigurable datapath makes it easy to utilize on-chip resources of an FPGA. However, the soft logics and overheads in routing resources lead to a low resource density and clock frequency. RDAs have the right balance between flexibility and efficiency, which gives a good overall average utilization.

fully saturate the compute throughput. GPU's bulk-synchronous nature also causes poor cache efficiency; data are spilled off-chip before getting reused [20]. On the outer hand, while extremely efficient for certain models, machine learning accelerators are highly specialized for specific kernels, especially general matrix multiply (GEMM) and convolution. However, ML algorithms evolve much faster than the development and manufacture cycle of hardware, leading to inefficiency or even unsupported applications. For example, hybrid models, such as Mask R-CNN [21] and DeepLab [22], contain combinations of GEMM-compatible and incompatible operations, both computationally expensive. Fix-functional accelerators focusing on GEMM operations, such as TPU [13], have to rely on CPUs for unsupported operations or convert non-GEMM operations to GEMM operations, resulting in a severe performance gap between the peak and effective FLOPS [23]. Figure 1.1 illustrate a tradeoff between average utilization of the peak FLOPS over a range of data and the peak FLOPS available on the hardware.

Reconfigurable spatial architectures overcome this limitation by changing its datapath based on applications' needs. Applications are configured at the circuit-level without dynamic instruction fetching and decoding, hence improving energy-efficiency [24, 25]. In addition to instruction, data, and task-level parallelism explored by processor architectures, spatial architectures also explore instruction and task-level pipelining that further increase the compute throughput [26]. Exploring

Architectures	DySER [36]	TI [41]	revel [49]	Plasticine [5]	Gorgon [46]
OPS	128GOPS	64GOPS	300GOPS	12.3TFLOPS	38.4TFLOPS

Table 1.1: OPS comparison of different CGRAs. Gorgon is a Plasticine variant proposed for joint machine learning and database acceleration.

pipeline parallelism enables spatial architectures to achieve a high-throughput without massively parallelize every stage of the program. Flexible datapath also permits resource distribution proportional to the compute intensity of program stages. A key performance optimization on reconfigurable accelerators is an application-level design space exploration searching for the best resource distribution scheme that balances the compute pipeline [27].

One of the mainstream reconfigurable spatial architecture is Field Programmable Gate Arrays (FPGAs) that support fine-grain, bit-level reconfigurability with a soft logic fabric [28]. The flexible interconnect and lookup table-based logic gate can be configured to implement arbitrary datapath. FPGAs have been used to deploy services commercially [29, 30, 31] and can be rented on the AWS F1 cloud [32]. Although around for a long time, FPGAs are not broadly accepted among high-level application programmers due to their low-level programming interface and long compilation time. Fine-tuning applications on FPGAs require expertise in digital design and takes a long development cycle, which hinder their accessibility to the general software community. As an application-level accelerator, FPGAs also suffer from overhead in fine-grained reconfigurability; studies have shown that over 60% of the chip area of an FPGA is spent on routing resources [33, 24, 25].

In contrast to fine-grained reconfigurable architectures, Coarse-Grained Reconfigurable Arrays (CGRAs) are spatial architectures with coarse-grained building blocks, such as ALUs, register files, and memory controllers, distributed in a programmable, word or vector-level static interconnect [34, 35, 36, 37, 38, 39, 40]. We refer a subclass of CGRAs with dataflow-driven execution model as Reconfigurable Dataflow Architectures (RDAs)[1] [5, 41, 42, 43, 44, 45]. Lately, RDAs are emerging as a new class of spatial accelerators that retain the desired level of flexibility and energy efficiency without the area overhead and low clock frequency of bit-level reconfigurability. Our previously proposed RDA–Plasticine–has demonstrated a promising acceleration of dense, sparse, database, and streaming applications [5, 46, 47, 48]. To meet the computing demand of recent data-analytic workload, Plasticine is a large-scale RDA compared to traditional CGRAs. Table 1.1 shows the OPS comparison across a few CGRAs proposed in the prior works.

[1] Dataflow overlay architectures on FPGA are technically also RDAs, which are not the primary concern in the discussion of this work.

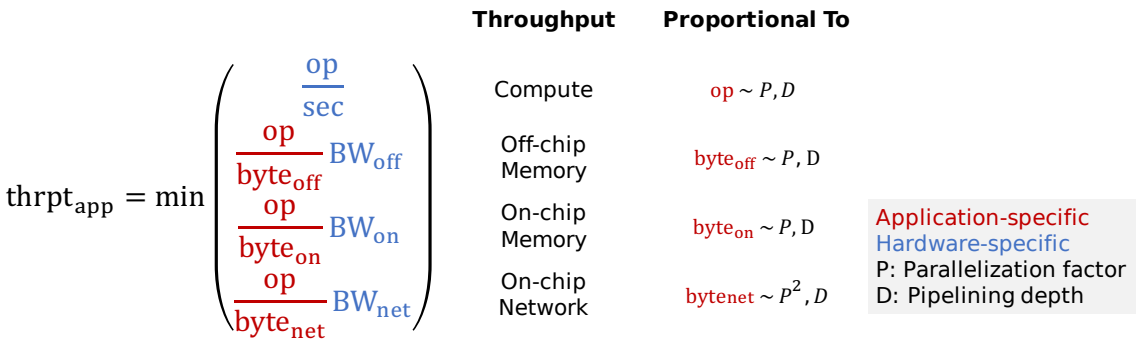


Figure 1.2: High-level performance model of a spatial architecture.

The scale of Plasticine introduces challenges in network-on-chip design to sustain bandwidth requirements while staying energy-efficient. The compilation strategy also needs to explore multiple-levels of concurrency in the program to saturate the compute throughput of the large-scale RDA; this strategy must introduce minimum synchronization overhead to maximize the scalability of the mapped design.

1.2 The Need for Flexible Interconnects

Applications are mapped to RDAs by distributing computations spatially across multiple processing blocks and executing them in a pipelined, data-driven fashion. In traditional Networks-on-Chip (NoCs) for multicore systems, communication is the result of explicit message passing between parallel workers or cache misses that generate messages for coherence protocol; these traffic are bursty and relatively infrequent. On RDAs, however, applications are distributed by parallelizing and pipelining; pipelining introduces frequent and throughput-sensitive communication. Applications with different characteristics, such as compute vs. memory-bound, also exhibit very different communication patterns.

Figure 1.2 shows a high-level performance model of a spatially pipelined and parallelized reconfigurable architecture. Due to pipelined execution, the performance of a spatial architecture is dominant by throughput as supposed to latency. Compute, on and off-chip memory accesses, and network become a integrated pipeline; overall performance is limited by the pipeline stage with lowest throughput. The red terms in the equation are application-specific and capture the compute, memory, or IO-bound characteristics. The blue terms are the bandwidth and FLOPS available on

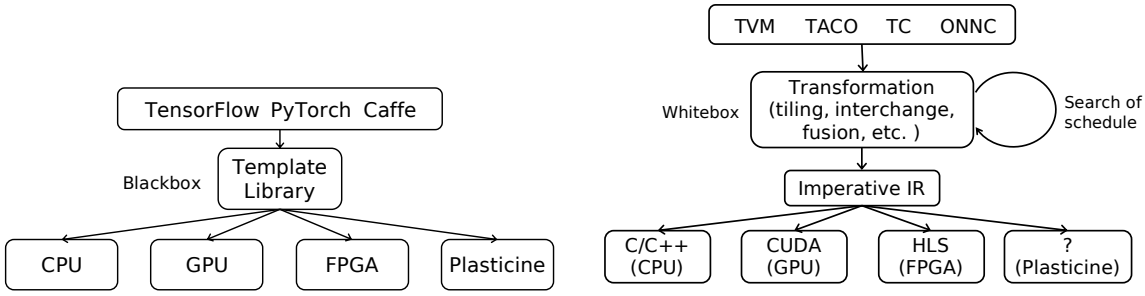
the hardware. While the compute and memory access in application increases linearly with parallelization factor and pipelining depth, the required network bandwidth from application increases quadratically with increasing parallelism. This means as we going to larger chip size, the network bandwidth must grow super linearly to achieve a perfect performance scaling, unless exploring pipeline parallelism.

RDAs need the right amount of interconnect flexibility to achieve good resource utilization; an inflexible interconnect constrains the space of valid application mappings and hinders resource utilization. Furthermore, in the quest to increase compute density, RDA data paths now contain increasingly coarse-grained processing blocks such as pipelined, vectorized functional units [5, 37, 50]. Plasticine, as an example, has a 512-bit vector bus, which necessitates coarser communication and higher on-chip interconnect bandwidth to avoid creating performance bottlenecks. Although many hardware accelerators with large, vectorized data paths have fixed local networks [51], there is a need for more flexible global networks to adapt to future applications. Consequently, interconnect design for these RDAs involves achieving a balance between the often conflicting requirements of high bandwidth and high flexibility.

1.3 The Gap between High-Level DSLs and Dataflow Accelerators

Recent years have seen an explosion of hardware accelerators, primarily motivated by AI applications [16, 17, 52, 12, 53, 15, 14, 54, 55, 56, 57, 58]. Research in software infrastructures for these accelerators, however, is just emerging. Unlike CPUs, accelerators do not support a standard instruction set architecture (ISA), alleviating the overhead from layers of abstractions and the burden of backward compatibility. Nonetheless, lack of a common abstraction makes it very hard to share compiler infrastructure across accelerators.

Most RDAs comes with a co-designed software layer that is very low-level and restrictive, requiring expertise knowledge in hardware architecture to efficiently target the accelerator. On the other side, machine learning frameworks [59, 60, 61, 62, 63, 64] provide high-level and succinct front-end abstraction that targets multiple hardware platforms. Yet most of theses frameworks only target mainstream accelerators, such as GPUs and FPGAs. Very few of the accelerators mentioned above can support more than a few proof-of-concept models with a end-to-end integration with these frameworks.



(a) Blackbox approach. (b) Whitebox approach.

Figure 1.3: Big picture of machine learning frameworks.

Part of the reason behind this is the massive engineering effort to program these accelerators. Frameworks like TensorFlow [59] and PyTorch [60] uses a template library approach, as shown in Figure 1.3 (a). These frameworks treats operations in the ML dataflow graph as blackbox kernels and implement every operation for every backend, requiring expensive engineering hours and hardware expertise to fine tune each accelerator. Furthermore, cross-kernel optimizations, such as fusion, often provide orders of magnitude improvement in efficiency. Recent studies in sparse ML also show promising speedup on large ML models [14]. However, sparse kernels require various data formats with very different implementation than the dense version [65]. In the blackbox approach, each fused operation and data format corresponds to a new blackbox kernel. As a result, the combination of required templates easily becomes untractable.

The alternative whitebox approach showin in Figure 1.3 (b) uses a succinct front-end representation, such as index notations [62], to express the computation of a kernel. This representation, however, cannot be directly mapped to accelerators due to various physical constraints. The framework then uses a series of transformations and optimizations to reformulate the kernel to an imperative implementation that can be supported by the accelerator. The transformation is often target-specific and studies have demonstrated ML-based cost models to facilitate the automatic searching of backend-specific schedules [66]. Next, the compiler can easily translates the imperative IR to the imperative front-ends of different accelerators.

To enable an easy integration of the whitebox approach, we introduce a compiler–SARA–that raise the programming abstraction of Plasticine to an imperative, loop-based language for general reconfigurable hardware—Spatial [67]. Plasticine is a purely dataflow accelerator without a centralized scheduler that executes instructions. This design makes the architecture very scalable to achieve FLOPS comparable to high-end GPUs and FPGAs. Nonetheless, it is not intuitive nor easy

to implement a linear algebra kernel with the low-level declarative streaming configurations. By support imperative constructs on a data-flow architecture efficiently, SARA not only improves the programmability of Plasticine, but also enables cross kernel optimizations on Plasticine. These optimizations have demonstrated an average 30x speedup over a Tesla V100 GPU and 2x speedup over a Stratix 10 FPGA from a Plasticine with much less on-chip resource than the baseline architectures [68] .

1.4 Contribution

In this work, we start by detailing the key considerations involved in building an RDA network, including those arising from network design, RDA architecture, and the characteristics of spatially mapped applications. Network designs must be carefully considered because vectorization magnifies inefficiencies: the increased network area of a vectorized design ensures that any overhead has a significant impact. Next, we evaluate the performance, area, and power requirements of several interconnection designs using cycle-accurate simulation and ASIC synthesis of a switch and router with a $28\,\mathrm{nm}$ industrial technology library. We then explore a variety of design points, including static, dynamic, and hybrid networks, decreased flit widths and VC counts for dynamic networks and different flow-control strategies for static networks.

We show that RDA network designs must consider application characteristics and the execution model of the underlying architecture. Performance scales strongly with network bandwidth, with an 8x average performance gap between the best and worst configurations. The hybrid network gives the best network energy-efficiency: a 1.83x average improvement over the static network. On pipelined architectures, hybrid networks can also match the performance per area of higher bandwidth, purely static networks with less than 8% performance loss.

The key contributions of this paper are:

1. An analysis of key communication patterns exhibited by spatial architectures.

2. A network-aware compiler flow that efficiently targets static, dynamic, and hybrid networks with varying granularities.

3. A quantitative analysis of the performance, area, and energy trade-offs involved in choosing an RDA network, using benchmarks drawn from various application domains.

1.5 Outline

The rest of this thesis is organized as follow: **??** gives a background on the execution schedule of spatial architectures, the front-end imperative language of SARA compiler, and the targeting architecture–Plasticine. **??** goes over the SARA compiler. **??** details the augmentations to the Plasticine architectures and the Network-on-chip study. **??** summarizes the related work. **??** concludes our work.

Bibliography

[1] C. Kachris and D. Soudris, "A survey on reconfigurable accelerators for cloud computing," in *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, pp. 1–10, 2016.

[2] S. Sarkar, T. Majumder, A. Kalyanaraman, and P. P. Pande, "Hardware accelerators for biocomputing: A survey," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pp. 3789–3792, 2010.

[3] S. Aluru and N. Jammula, "A review of hardware acceleration for computational genomics," *IEEE Design Test*, vol. 31, no. 1, pp. 19–30, 2014.

[4] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *2011 38th Annual International Symposium on Computer Architecture (ISCA)*, pp. 365–376, June 2011.

[5] R. Prabhakar, Y. Zhang, D. Koeplinger, M. Feldman, T. Zhao, S. Hadjis, A. Pedram, C. Kozyrakis, and K. Olukotun, "Plasticine: A reconfigurable architecture for parallel paterns," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ISCA '17, (New York, NY, USA), pp. 389–402, ACM, 2017.

[6] Y. Zhang, A. Rucker, M. Vilim, R. Prabhakar, W. Hwang, and K. Olukotun, "Scalable interconnects for reconfigurable spatial architectures," in *Proceedings of the 46th International Symposium on Computer Architecture*, ISCA '19, (New York, NY, USA), pp. 615–628, ACM, 2019.

[7] R. H. Dennard, F. H. Gaensslen, V. L. Rideout, E. Bassous, and A. R. LeBlanc, "Design of ion-implanted mosfet's with very small physical dimensions," *IEEE Journal of Solid-State Circuits*, vol. 9, pp. 256–268, Oct 1974.

[8] R. Hameed, W. Qadeer, M. Wachs, O. Azizi, A. Solomatnikov, B. C. Lee, S. Richardson, C. Kozyrakis, and M. Horowitz, "Understanding sources of inefficiency in general-purpose chips," in *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ISCA '10, (New York, NY, USA), pp. 37–47, ACM, 2010.

[9] M. A. Dias and D. A. P. Ferreira, "Deep learning in reconfigurable hardware: A survey," in *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 95–98, 2019.

[10] A. Gielata, P. Russek, and K. Wiatr, "Aes hardware implementation in fpga for algorithm acceleration purpose," in *2008 International Conference on Signals and Electronic Systems*, pp. 137–140, 2008.

[11] J. L. Hennessy and D. A. Patterson, "A new golden age for computer architecture," *Commun. ACM*, vol. 62, pp. 48–60, Jan. 2019.

[12] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun, and O. Temam, "Dadiannao: A machine-learning supercomputer," in *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-47, (Washington, DC, USA), pp. 609–622, IEEE Computer Seiociety, 2014.

[13] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P.-l. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon, "In-datacenter performance analysis of a tensor processing unit," *SIGARCH Comput. Archit. News*, vol. 45, pp. 1–12, June 2017.

[14] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: Efficient inference engine on compressed deep neural network," in *Proceedings of the 43rd International*

Symposium on Computer Architecture, ISCA '16, (Piscataway, NJ, USA), pp. 243–254, IEEE Press, 2016.

[15] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2017.

[16] M. Gao, X. Yang, J. Pu, M. Horowitz, and C. Kozyrakis, "Tangram: Optimized coarse-grained dataflow for scalable nn accelerators," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '19, (New York, NY, USA), p. 807–820, Association for Computing Machinery, 2019.

[17] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G. Nam, B. Taba, M. Beakes, B. Brezzo, J. B. Kuang, R. Manohar, W. P. Risk, B. Jackson, and D. S. Modha, "Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1537–1557, 2015.

[18] K. Rupp, "www.karlrupp.net."

[19] J. W. Richardson, A. D. George, and H. Lam, "Performance analysis of gpu accelerators with realizable utilization of computational density," in *2012 Symposium on Application Accelerators in High Performance Computing*, pp. 137–140, 2012.

[20] J. Hestness, S. W. Keckler, and D. A. Wood, "Gpu computing pipeline inefficiencies and optimization opportunities in heterogeneous cpu-gpu processors," in *2015 IEEE International Symposium on Workload Characterization*, pp. 87–97, 2015.

[21] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988, 2017.

[22] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2018.

[23] C. Guo, Y. Zhou, J. Leng, Y. Zhu, Z. Du, Q. Chen, C. Li, M. Guo, and B. Yao, "Balancing efficiency and flexibility for dnn acceleration via temporal gpu-systolic array integration," 2020.

[24] B. H. Calhoun, J. F. Ryan, S. Khanna, M. Putic, and J. Lach, "Flexible circuits and architectures for ultralow power," *Proceedings of the IEEE*, vol. 98, pp. 267–282, Feb 2010.

[25] K. K. W. Poon, S. J. E. Wilton, and A. Yan, "A detailed power model for field-programmable gate arrays," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 10, pp. 279–302, Apr. 2005.

[26] M. Budiu, G. Venkataramani, T. Chelcea, and S. C. Goldstein, "Spatial computation," in *Proceedings of the 11th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XI, (New York, NY, USA), pp. 14–26, ACM, 2004.

[27] D. Koeplinger, R. Prabhakar, Y. Zhang, C. Delimitrou, C. Kozyrakis, and K. Olukotun, "Automatic generation of efficient accelerators for reconfigurable hardware," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pp. 115–127, June 2016.

[28] I. Kuon, R. Tessier, and J. Rose, "Fpga architecture: Survey and challenges," *Foundations and Trends in Electronic Design Automation*, vol. 2, pp. 135–253, 01 2007.

[29] A. Putnam, A. M. Caulfield, E. S. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G. P. Gopal, J. Gray, M. Haselman, S. Hauck, S. Heil, A. Hormati, J.-Y. Kim, S. Lanka, J. Larus, E. Peterson, S. Pope, A. Smith, J. Thong, P. Y. Xiao, and D. Burger, "A reconfigurable fabric for accelerating large-scale datacenter services," in *Proceeding of the 41st Annual International Symposium on Computer Architecuture*, ISCA '14, (Piscataway, NJ, USA), pp. 13–24, IEEE Press, 2014.

[30] J. Ouyang, S. Lin, W. Qi, Y. Wang, B. Yu, and S. Jiang, "Sda: Software-defined accelerator for largescale dnn systems," Hot Chips 26, 2014.

[31] K. Guo, L. Sui, J. Qiu, S. Yao, S. Han, Y. Wang, and H. Yang, "From model to fpga: Software-hardware co-design for efficient neural network acceleration," in *2016 IEEE Hot Chips 28 Symposium (HCS)*, pp. 1–27, Aug 2016.

[32] A. AWS, "Amazon ec2 f1 instances." `https://aws.amazon.com/ec2/instance-types/f1`.

[33] I. Kuon, R. Tessier, and J. Rose, "Fpga architecture: Survey and challenges," *Found. Trends Electron. Des. Autom.*, vol. 2, pp. 135–253, Feb. 2008.

[34] B. Mei, S. Vernalde, D. Verkest, H. De Man, and R. Lauwereins, "Adres: An architecture with tightly coupled vliw processor and coarse-grained reconfigurable matrix," in *Field Programmable Logic and Application* (P. Y. K. Cheung and G. A. Constantinides, eds.), (Berlin, Heidelberg), pp. 61–70, Springer Berlin Heidelberg, 2003.

[35] R. Kress, "A fast reconfigurable alu for xputers," 1996.

[36] V. Govindaraju, C.-H. Ho, and K. Sankaralingam, "Dynamically specialized datapaths for energy efficient computing," in *Proceedings of the 2011 IEEE 17th International Symposium on High Performance Computer Architecture*, HPCA '11, (Washington, DC, USA), pp. 503–514, IEEE Computer Society, 2011.

[37] S. C. Goldstein, H. Schmit, M. Moe, M. Budiu, S. Cadambi, R. R. Taylor, and R. Laufer, "Piperench: a coprocessor for streaming multimedia acceleration," in *Proceedings of the 26th International Symposium on Computer Architecture* (*Cat. No.99CB36367*), pp. 28–39, May 1999.

[38] M. Mishra, T. J. Callahan, T. Chelcea, G. Venkataramani, S. C. Goldstein, and M. Budiu, "Tartan: Evaluating spatial computation for whole program execution," in *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XII, (New York, NY, USA), pp. 163–174, ACM, 2006.

[39] M. Gao and C. Kozyrakis, "Hrl: Efficient and flexible reconfigurable logic for near-data processing," in *2016 IEEE International Symposium on High Performance Computer Architecture* (*HPCA*), pp. 126–137, March 2016.

[40] M. Karunaratne, A. K. Mohite, T. Mitra, and L.-S. Peh, "Hycube: A cgra with reconfigurable single-cycle multi-hop interconnect," in *Proceedings of the 54th Annual Design Automation Conference 2017*, DAC '17, (New York, NY, USA), pp. 45:1–45:6, ACM, 2017.

[41] A. Parashar, M. Pellauer, M. Adler, B. Ahsan, N. Crago, D. Lustig, V. Pavlov, A. Zhai, M. Gambhir, A. Jaleel, R. Allmon, R. Rayess, S. Maresh, and J. Emer, "Triggered instructions: A control paradigm for spatially-programmed architectures," in *Proceedings of the 40th Annual International Symposium on Computer Architecture*, ISCA '13, (New York, NY, USA), pp. 142–153, ACM, 2013.

[42] T. Nowatzki, V. Gangadhar, N. Ardalani, and K. Sankaralingam, "Stream-dataflow acceleration," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture* (*ISCA*), pp. 416–429, June 2017.

[43] C. Farabet, B. Martini, B. Corda, P. Akselrod, E. Culurciello, and Y. LeCun, "Neuflow: A runtime reconfigurable dataflow processor for vision," in *CVPR 2011 WORKSHOPS*, pp. 109–116, 2011.

[44] M. Oh, C. Lee, S. Lee, Y. Seo, S. Kim, J. Wang, and C. Sungchung Park, "Convolutional neural network accelerator with reconfigurable dataflow," in *2018 International SoC Design Conference (ISOCC)*, pp. 42–43, 2018.

[45] A. Niedermeier, J. Kuper, and G. Smit, "Dataflow-based reconfigurable architecture for streaming applications," in *2012 International Symposium on System on Chip (SoC)*, pp. 1–4, 2012.

[46] M. Vilim, A. Rucker, Y. Zhang, S. Liu, and K. Olukotun, "Gorgon: Accelerating machine learning from relational data," in *Proceedings of the 47th International Symposium on Computer Architecture*, ISCA '20, 2020.

[47] R. Singhal, Y. Zhang, J. Ullman, R. Prabhakar, and K. Olukotun, "Efficient multiway hash join on reconfigurable hardware," in *Technology Conference on Performance Evaluation and Benchmarking*, 05 2019.

[48] R. Prabhakar, O. A. Olukotun, C. Kozyrakis, and C. Re. PhD thesis, 2018.

[49] J. Weng, S. Liu, Z. Wang, V. Dadu, and T. Nowatzki, "A hybrid systolic-dataflow architecture for inductive matrix algorithms," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 703–716, 2020.

[50] J. Noguera, C. Dick, V. Kathail, G. Singh, K. Vissers, and R. Wittig, "Xilinx project everest: 'hw/sw programmable engine'," Hot Chips 30, 2018.

[51] J. Fowers, K. Ovtcharov, M. Papamichael, T. Massengill, M. Liu, D. Lo, S. Alkalay, M. Haselman, L. Adams, M. Ghandi, S. Heil, P. Patel, A. Sapek, G. Weisz, L. Woods, S. Lanka, S. K. Reinhardt, A. M. Caulfield, E. S. Chung, and D. Burger, "A configurable cloud-scale DNN processor for real-time AI," in *45th ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2018, Los Angeles, CA, USA, June 1-6, 2018*, pp. 1–14, 2018.

[52] M. N. Bojnordi and E. Ipek, "Memristive boltzmann machine: A hardware accelerator for combinatorial optimization and deep learning," in *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 1–13, 2016.

[53] D. Liu, T. Chen, S. Liu, J. Zhou, S. Zhou, O. Teman, X. Feng, X. Zhou, and Y. Chen, "Pudiannao: A polyvalent machine learning accelerator," in *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '15, (New York, NY, USA), p. 369–381, Association for Computing Machinery, 2015.

[54] X. Liu, M. Mao, B. Liu, H. Li, Y. Chen, B. Li, Yu Wang, Hao Jiang, M. Barnell, Qing Wu, and Jianhua Yang, "Reno: A high-efficient reconfigurable neuromorphic computing accelerator design," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2015.

[55] W. Qadeer, R. Hameed, O. Shacham, P. Venkatesan, C. Kozyrakis, and M. A. Horowitz, "Convolution engine: Balancing efficiency & flexibility in specialized computing," in *Proceedings of the 40th Annual International Symposium on Computer Architecture*, ISCA '13, (New York, NY, USA), p. 24–35, Association for Computing Machinery, 2013.

[56] B. Reagen, P. Whatmough, R. Adolf, S. Rama, H. Lee, S. K. Lee, J. M. Hernández-Lobato, G. Wei, and D. Brooks, "Minerva: Enabling low-power, highly-accurate deep neural network accelerators," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pp. 267–278, 2016.

[57] D. Shin, J. Lee, J. Lee, and H. Yoo, "14.2 dnpu: An 8.1tops/w reconfigurable cnn-rnn processor for general-purpose deep neural networks," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 240–241, 2017.

[58] L. Song, Y. Wang, Y. Han, X. Zhao, B. Liu, and X. Li, "C-brain: A deep learning accelerator that tames the diversity of cnns through adaptive data-level parallelization," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2016.

[59] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, (Berkeley, CA, USA), pp. 265–283, USENIX Association, 2016.

[60] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[61] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 8024–8035, Curran Associates, Inc., 2019.

[62] F. Kjolstad, S. Kamil, S. Chou, D. Lugato, and S. Amarasinghe, "The tensor algebra compiler," *Proc. ACM Program. Lang.*, vol. 1, pp. 77:1–77:29, Oct. 2017.

[63] W. Lin, D. Tsai, L. Tang, C. Hsieh, C. Chou, P. Chang, and L. Hsu, "Onnc: A compilation framework connecting onnx to proprietary deep learning accelerators," in *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pp. 214–218, 2019.

[64] N. Vasilache, O. Zinenko, T. Theodoridis, P. Goyal, Z. DeVito, W. S. Moses, S. Verdoolaege, A. Adams, and A. Cohen, "Tensor comprehensions: Framework-agnostic high-performance machine learning abstractions," *CoRR*, vol. abs/1802.04730, 2018.

[65] S. Chou, F. Kjolstad, and S. Amarasinghe, "Format abstraction for sparse tensor algebra compilers," *Proc. ACM Program. Lang.*, vol. 2, pp. 123:1–123:30, Oct. 2018.

[66] T. Chen, T. Moreau, Z. Jiang, L. Zheng, E. Yan, M. Cowan, H. Shen, L. Wang, Y. Hu, L. Ceze, C. Guestrin, and A. Krishnamurthy, "Tvm: An automated end-to-end optimizing compiler for deep learning," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'18, (USA), p. 579–594, USENIX Association, 2018.

[67] D. Koeplinger, M. Feldman, R. Prabhakar, Y. Zhang, S. Hadjis, R. Fiszel, T. Zhao, L. Nardi, A. Pedram, C. Kozyrakis, and K. Olukotun, "Spatial: A language and compiler for application accelerators," in *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2018, (New York, NY, USA), pp. 296–311, ACM, 2018.

[68] T. Zhao, Y. Zhang, and K. Olukotun, "Serving recurrent neural networks efficiently with a spatial accelerator," in *Proceedings of the 2 nd SysML Conference*, 2019.