SCALING DEEP-LEARNING INFERENCE WITH MULTI-CHIP-MODULE-BASED ARCHITECTURE

基于多模块架构的扩展深度学习推理

[1]M. Yousaf

ABSTRACT

Package-level integration using multi-chip-modules (MCMs) is a promising approach for building large-scale systems. Compared to a large monolithic die, an MCM combines many smaller chiplets into a larger system, substantially reducing fabrication and design costs. Current MCMs typically only contain a handful of coarse-grained large chiplets due to the high area, performance, and energy overheads associated with inter-chiplet communication. This work investigates and quantifies the costs and benefits of using MCMs with fine-grained chiplets for deep learning inference, an applica-tion area with large compute and on-chip storage requirements.

To evaluate the approach, we architected, implemented, fabricated, and tested Simba, a 36-chiplet prototype MCM system for deep-learning inference

庞大的 粗粒的 芯片 细粒的

[1]University of Okara

KEYWORDS

Multi-chip module, neural networks, accelerator architecture

ACM Reference Format:

Yakun Sophia Shao, Jason Clemons, Rangharajan Venkatesan, Brian Zim-mer, Matthew Fojtik, Nan Jiang, Ben Keller, Alicia Klinefelter, Nathaniel Pinckney, Priyanka Raina, Stephen G. Tell, Yanqing Zhang, William J. Dally, Joel Emer, C. Thomas Gray, Brucek Khailany, and Stephen W. Keckler. 2019.

INTRODUCTION

Deep learning (DL) [44] has become critical for addressing complex real-world problems. In particular, deep neural networks (DNNs) have demonstrated their effectiveness across a wide-range of appli-cations, including image recognition [33, 41, 60, 64, 65], object de-tection [27, 54], language translation [63, 70], audio synthesis [69], and autonomous driving [10]. State-of-the-art DNNs [6, 12, 27, 33, 41, 46, 54, 60, 64, 65] require billions of operations and hundreds of megabytes to store activations and weights. Given the trend towards even larger and deeper networks, the ensuing compute and storage requirements motivate large-scale compute capability in DL hardware, which is currently addressed by a combination of large monolithic chips and homogeneous multi-chip board de-signs [14, 17, 24, 29, 39, 71]. Previously proposed multi-chip DL accelerators have focused on improving total compute throughput and on-chip storage size but have not addressed the scalability chal-lenges associated with building a large-scale system with multiple discrete components.

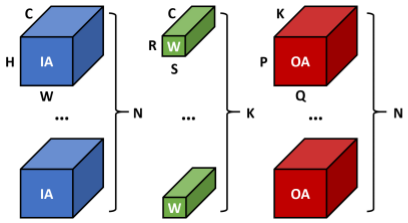


Figure 1: Input activations (IA), weights (W), and output ac-tivations (OA) in convolutional layers.

Recently, the need for high compute throughput in an era of slowing transistor scaling has motivated advances in multi-chip-module (MCM) integration to build large-scale CPUs [7, 37, 40, 62] and GPUs [3, 74]. MCM packaging approaches can also reduce cost by employing smaller chiplets connected together post-fabrication, as yield losses cause fabrication cost to grow super-linearly with die size. Packaging technologies including organic substrates [36] and silicon interposers [28, 55] can be used to assemble a large-scale MCM system. In addition, recent advances in package-level signaling offer the necessary high-speed, high-bandwidth signaling needed for chiplet-based system [72]. As a result, chiplet-based systems using MCMs can provide improved performance more efficiently than board-level integration but with lower cost than monolithic chips. While MCMs have been used for general compute systems, applying MCMs to high-performance DNN inference algo-rithms has not been previously examined. Specific challenges stem from the natural non-uniformity between on-chip and on-package bandwidth and latency. While multi-chip systems also exhibit sim-ilar forms of non-uniformity, this paper focuses on the specific characteristics of MCM-based systems as they provide a natural progression from monolithic single-chip inference accelerators as semiconductor scaling slows.

This paper presents Simba, a scalable deep-learning inference ac-celerator employing multi-chip-module-based integration. Each of the Simba chiplets can be used as a standalone, edge-scale inference accelerator, while multiple Simba chiplets can be packaged together to deliver data-center-scale compute throughput. We specifically ex-amine the implications of the non-uniform latency and bandwidth for on-chip and on-package communication that lead to significant latency variability across chiplets. Such latency variability results in a long “tail latency” during the execution of individual infer-ence layers. As a result, the overall performance for each layer is limited by the slowest chiplet in the system, limiting scalability. To address these challenges, we propose three tail-latency-aware, non-uniform tiling optimizations targeted at improving locality and minimizing inter-chiplet communication: (1) non-uniform work partitioning to balance compute latency with communication la-tency; (2) communication-aware data placement to minimize inter-chiplet traffic; and (3) cross-layer pipelining to improve resource utilization.

To explore the challenges and evaluate the benefits of MCM-based inference accelerator architectures, we designed, implemented, and fabricated a prototype of Simba, consisting of 36 chiplets con-nected in a mesh network in an MCM [76]. The 6 mm^2 chiplets are

```
1  for n = [0 : N) :
2for p = [0 : P) :
3for q = [0 : Q) :
4for k = [0 : K) :
5for r = [0 : R) :
6          for s = [0 : S) :
7            for c = [0 : C) :
8              OA[n,p,q,k] += IA[n,h,w,c] *
               W[k,r,s,c]
```

Listing 1: DNN loop nest.
嵌套循环

fabricated in a 16 nm FinFET process technology and contain both multiply-accumulate (MAC) units and an SRAM-based memory system. Each chip has a peak performance of 4 tera-ops per second (TOPS) using 8-bit weights and activations and 24-bit accumulation. The 36-chiplet MCM achieves up to 128 TOPS with an energy effi-ciency range of 0.2–6.1 TOPS/W depending on operating voltage. Simba supports flexible mapping and resource allocation for effi-cient DNN inference execution across a wide range of workloads. We thoroughly characterize the Simba system and motivate the importance of task and data placement in the Simba MCM system, which can lead to as much as a 2.5× performance difference for indi-vidual ResNet-50 layers. Motivated by our observations, we propose non-uniform work and data placement, together with cross-layer pipelining, to improve system utilization in the presence of small batch sizes and communication latency.

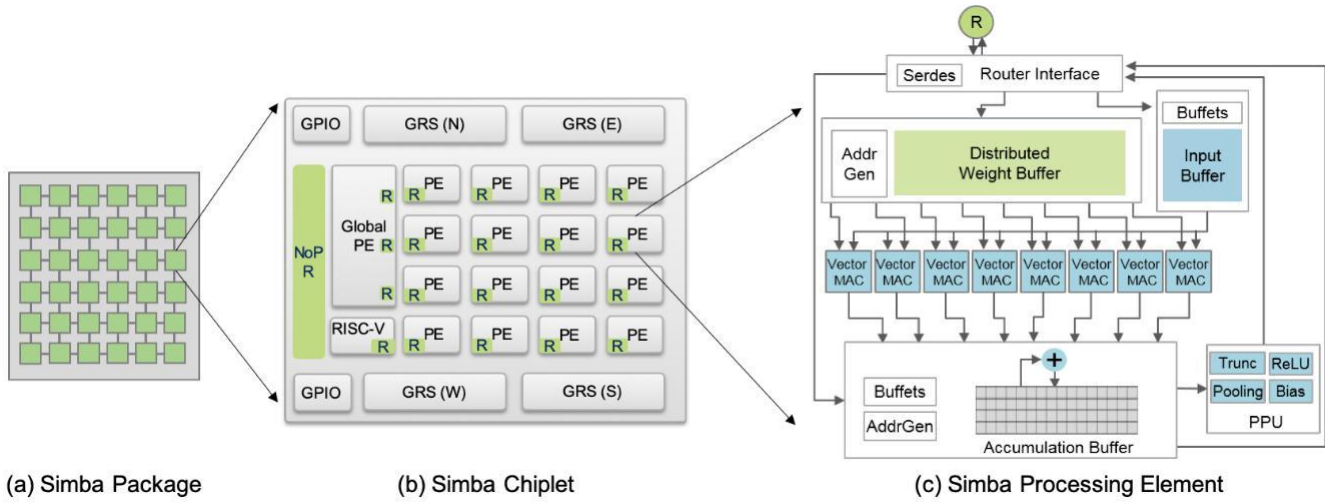
2 BACKGROUND AND MOTIVATION

Many applications from edge devices to data centers demand fast and efficient inference, often with low latency or real-time through-put requirements. Today's DNN inference applications typically run on highly programmable but inefficient CPU-based systems, programmable GPUs with ISA extensions for accelerating tensor operations, or fixed-function DNN inference accelerators. Recent work has shown that fixed-function accelerators can provide or-ders of magnitude better energy efficiency and performance than CPUs and better area and energy efficiency than GPUs [1, 13– 15, 20, 25, 30, 51, 59, 75].

Small-scale inference applications can run on moderately-sized chips, while those demanding higher performance may require large monolithic chips or board-level multi-chip solutions to scale. Partitioning an application across multiple chips at the board level is not easy because of the enormous difference in bandwidth, la-tency, and energy between on-chip communication and inter-chip communication. Other packaging approaches such as multi-chip modules can provide inter-chip interconnect that is closer in nature to on-chip interconnect, offering a more straightforward path to scaling.

2.1 DNN Basics

DNNs are constructed using a series of layers, including convolu-tional layers, pooling layers, activation layers, and fully-connected layers. A convolutional layer is algorithmically formulated as a seven-dimensional nested loop over an input activation (IA) ten-sor, a weight (W) tensor, and an output activation (OA) tensor, as shown in Figure 1. Listing 1 shows the convolution computation embedded in a seven-dimensional loop nest. The same formulation



处理单元

Figure 2: Simba architecture from package to processing element (PE).

also applies to fully-connected layers that are widely used in multi-layer perceptrons (MLPs) and recurrent neural networks (RNNs). An activation layer applies a non-linear function such as ReLU or sigmoid, while pooling layers down-sample the input activations after convolutional layers. Activation and pooling layers are typically merged with convolutional layers during execution to reduce data movement.

多层感知器

循环神经网络

2.2 Multi-Chip-Module Packaging

Package-level MCM integration is a promising alternative for as-sembling large-scale systems out of small building blocks known as chiplets. Such systems consist of multiple chiplets connected together via on-package links using a silicon interposer or an or-ganic substrate and employing efficient intra-package signaling circuits [7, 21, 40, 72]. Compared to a large monolithic die, MCMs can reduce (1) design costs, since logic design, verification, and physical design are all easier on a small chip than a large chip; and (2) fabrication costs, as the much lower manufacturing yield of large chips make them far more expensive than small chips. In addition, different scales of systems can be created merely by adjusting the number of chiplets placed in a package, without re-quiring a different chip tapeout for each market segment. MCMs have been recently applied to a general-purpose CPU design [7] as an alternative to building multi-core CPUs on reticle-limited large die. They have also been an active research area for scaling of multi-CPU [37, 40, 45, 62] and multi-GPU systems [3, 18, 74]. How-ever, package-level wires do not provide the same communication density or energy/bit as on-chip wires. Consequently, MCM archi-tects and software developers must still consider the non-uniform bandwidth, latency, and energy present in these systems to achieve efficient application performance.

内置包

因此

2.3 Non-Uniformity in MCM-based Design

An MCM-based system has a heterogeneous interconnect archi-tecture, as the available intra-chiplet bandwidth is expected to be

各种各样的

内部

significantly higher than available inter-chiplet bandwidth. In addi-tion, sending data to remote chiplets incurs additional latency. This latency may include on-chip wire delays to move data to the edge of the chiplet, synchronizer delays for crossing clock domains, seri-alization and deserialization latency in high-speed communication links, and the on-package wire delays of inter-chiplet links. As a result, communication latency between two elements in a MCM heavily depends on their spatial locality on the package.

Mapping DNN layers to a tile-based architecture is a well-studied research problem [15, 25, 50]. State-of-the-art DNN tiling typically assumes a flat architecture with uniform latency and bandwidth across processing elements and focuses on data reuse for reducing global bandwidth demands. This assumption is acceptable for small-scale systems, as the communication latency variability is small, and the computation is often tolerant of communication latencies. However, as DNN inference performance is scaled up to larger systems, the execution time decreases and latency-related effects become more important. Furthermore, in large-scale systems with heterogeneous interconnect architectures such as MCMs, assump-tions of uniform latency and bandwidth in selecting DNN tiling can degrade performance and energy efficiency. Simba is the first work that quantitatively highlights the challenge of mapping DNN layers to non-uniform, MCM-based DNN accelerators and proposes communication-aware tiling strategies to address the challenge.

3 SIMBA ARCHITECTURE AND SYSTEM

To understand the challenges and opportunities of using MCMs for building large-scale, deep-learning systems, we designed, imple-mented, fabricated, and characterized Simba, the first chiplet-based deep-learning system. This section first presents an overview of the Simba architecture and its default uniform tiling strategy. We then describe Simba's silicon prototype and present a detailed char-acterization of the Simba system in Section 4.

MICRO-52, October 12–16, 2019, Columbus, OH, USA

Table 1: Simba system communication capability.

Packet Source	Unicast 单点传送 Destination	Multicast 多点广播 Destination
PE	Local PEs, Global PE, Controller	-
	Remote PEs, Global PE, Controller	-
Global PE	Local PEs, Controller	Local PEs
	Remote PEs, Controller	Remote PEs
Controller	Local PEs, Global PE	
	Remote PEs, Global PE, Controller	

3.1 Simba Architecture

Tile-based architectures have frequently been proposed for deep-learning accelerator designs [1, 2, 13, 15, 22, 25, 30, 42, 51, 53, 56, 58, 61, 75]. Our design target is an accelerator scalable to data center inference, where state-of-the-art data center accelerators deliver around 100 tera-operations-per-second (TOPS). For example, the first generation of the Tensor Processing Unit (TPU) delivers 92 TOPS [39] and is designed for inference applications. One simple ap-proach to achieve this design goal is to increase the number of tiles in a monolithic single chip. However, building a flat network with hundreds of tiles would lead to high tile-to-tile communication latency, as examined in both multi-core CPU [19] and accelerator [26] research.

Simba adopts a hierarchical interconnect to efficiently connect different processing elements (PEs). This hierarchical 体系的 interconnect consists of a network-on-chip (NoC) that connects PEs on the same chiplet and a network-on-package (NoP) that connects chiplets together on the same package. Figure 2 illustrates the three-level hierarchy of the Simba architecture: package, chiplet, and PE. Fig-ure 2(a) shows a Simba package consisting of a 6 × 6 array of Simba chiplets connected via a mesh interconnect. Each Simba chiplet, as shown in Figure 2(b), contains an array of PEs, a global PE, a NoP router, and a controller, all connected by a chiplet-level interconnect. To enable the design of a large-scale system, all com-munication between the PEs, Global PEs, and controller is designed to be latency-insensitive [11] and is sent across the interconnection network through the NoC/NoP routers.

Simba PE: Figure 2(c) shows the microarchitecture of the Simba PE, which includes a distributed weight buffer, an input buffer, paral-lel vector MAC units, an accumulation buffer, and a post-processing unit. Each Simba PE is similar to a scaled-down version of NVDLA, a state-of-the-art DL accelerator product [59]. The heart of the Simba PE is an array of parallel vector multiply-and-add (MAC) units that are optimized for efficiency and flexibility. The Simba PE uses a weight-stationary dataflow: weights remain in the vector MAC registers and are reused across iterations, while new inputs are read every cycle. Each vector MAC performs an 8:1 dot-product along the input channel dimension C to exploit an efficient spatial

积累 数据流 迭代 矢量 点积 利用

reduction [42]. To provide flexible tiling options, the Simba PE also supports cross-PE reduction with configurable producers and con-sumers. If the current PE is the last PE on the reduction chain, it first sends partial sums to its local post-processing unit that performs ReLU, truncation and scaling, pooling, and bias addition. The final output activation is sent to the target Global PE for computation of the next layer.

Simba Global PE: The Global PE serves as second-level stor-age for input/output activation data to be processed by the PEs. To support flexible partitioning of the computation, the Global PE can either unicast data to one PE or multicast to multiple PEs, even across chiplet boundaries. The Global PE has a multicast manager that oversees these producer-consumer relationships. The Global PE also serves as a platform for near-memory computation. Many DNNs feature some computation that has low data reuse, such as element-wise multiply/add in ResNet [33] or depth-wise convolu-tion in MobileNet [34]. The Global PE can perform such computa-tions locally to reduce communication overhead for these types of operations.

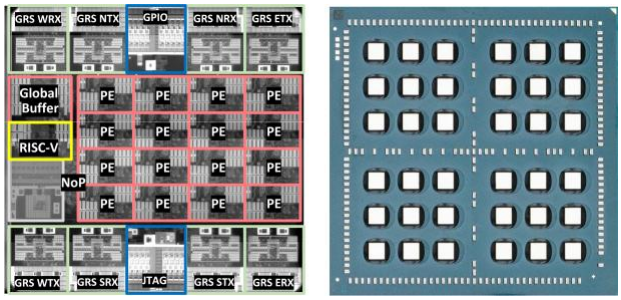
3.2 Simba Silicon Prototype

Simba Controller: Each Simba chiplet contains a RISC-V proces-sor core [4] that is responsible for configuring and managing the chiplet's PEs and Global PE states via memory-mapped registers using an AXI-based communication protocol. 协议
After all states are configured, the RISC-V triggers execution in the active PEs and Global PEs and waits for these blocks to send done notifications via interrupts. Synchronization of chiplet control processors across the package is implemented via memory-mapped interrupts.

Simba Interconnect: To efficiently execute different neural networks with diverse layer dimensions, Simba supports flexible communication patterns across the NoC and NoP. Table 1 lists Simba communication capability across all components. Both NoC and NoP use a mesh topology with a hybrid wormhole/cut-
网状拓扑结构

through flow control. Specifically, unicast packets use wormhole flow control for large packet size, while multicast packets are cut-through to avoid wormhole deadlocks. Each Simba PE can unicast to any local or remote PE for cross-PE partial-sum reduction, to any local or remote Global PE to transmit output activation values, and to any local or remote chiplet controller to signal execution completion. A PE does not need to send multicast packets as its computation requires only point-to-point communication. In addition to unicast communication, a Global PE can also send multicast packets to local and remote PEs for flexible data tiling.

We implemented, fabricated, and tested a silicon prototype of the Simba system, shown in Figure 3, with the microarchitecture



(a) Simba chiplet (b) Simba package

Figure 3: Simba silicon prototype.

Simba: Scaling Deep-Learning Inference with Multi-Chip-Module-Based Architecture

Table 2: Simba microarchitecture parameters.

Package	Number of Chiplets	36
	Size	47.5 mm × 47.5 mm
	Core Voltage	0.52–1.1 V
	PE Clock Frequency	0.48–1.8 GHz
	Chiplet-to-Chiplet Interconnect	Ground-Referenced Signaling
	NoP Interconnect Bandwidth	100 GB/s/Chiplet
	NoP Interconnect Latency	20 ns/Hop
	NoP Interconnect Energy	0.82–1.75 pJ/bit
Chiplet	Number of PEs	16
	Area	2.5 mm × 2.4 mm
	Technology	16 nm FinFET
	Voltage	0.42–1.2 V
	PE Clock Frequency	0.16–2.0 GHz
	Global PE Buffer Size	64 KiB
	Routers Per Global PE	3
	NoC Interconnect Bandwidth	68 GB/s/PE
	NoC Interconnect Latency	10 ns/Hop
	Microcontroller	RISC-V
PE	Weight Buffer Size	32 KiB
	Input Buffer Size	8 KiB
	Accumulation Buffer Size	3 KiB
	Vector MAC Width	8
	Number of Vector MACs	8
	Dataflow	Weight Stationary
	Input/Weight Precision	8 bits
	Partial-Sum Precision	24 bits

parameters in Table 2. We chose parameters so that a Simba chiplet has area and power similar to an efficient edge system, such as Dian-Nao [13] or Eyeriss [15], while a full Simba package is comparable to a data-center-scale system such as TPU [39]. Table 3 shows the synthesis area breakdown of key components in the Simba chiplet architecture.

Shown in Figure 3a, the 2.5 mm × 2.4 mm Simba chiplets were implemented in TSMC 16 nm FinFET process technology [76]. Each Simba package (Figure 3b) contains an array of 6 × 6 chiplets con-nected on an organic package substrate using ground-referenced signaling (GRS) technology for intra-package communication [72]. The top and bottom rows of each chiplet include eight chiplet-to-chiplet GRS transceiver macros. Four macros are configured as receivers and four as transmitters. Each transceiver macro has four data lanes and a clock lane with configurable speed from 11 Gbps/pin to 25 Gbps/pin, consuming 0.82–1.75 pJ/bit, with a to-tal peak chiplet bandwidth of 100 GB/s. We chose GRS as our com-munication mechanism because it delivers 3.5× higher bandwidth per unit area and lower energy per bit compared to other MCM interconnects [7].

The prototype chiplets were implemented using a globally asyn-chronous, locally synchronous (GALS) clocking methodology [23],

Table 3: Area breakdown of the Simba system.

Partition	Component	Area (μm^2)
PE	Vector MACs	12K
	Weight Buffer	41K
	Input Buffer	11K
	Accumulation Buffer	24K
	NoC Router	19K
Global PE	Distributed Buffer	125K
	NoC Routers	27K
RISC-V	Processor	109K
NoP	NoP Router	42K

allowing independent clock rates for individual PEs, Global PEs, RISC-V processors, and NoP routers. Running in a single-chiplet configuration, Simba prototypes have been measured to operate correctly in the lab at a minimum voltage of 0.42 V with a 161 MHz PE frequency, achieving 0.11 pJ/Op (9.1 TOPS/W) core power ef-ficiency on a peak-utilization convolution micro-benchmark. At 1.2 V, each chiplet operates with a 2 GHz PE frequency for a peak throughput of 4 TOPS. The 36-chiplet Simba system is functional over a slightly narrower voltage range, from 0.52–1.1 V, achieving 0.16 pJ/op at 0.52 V and 484 MHz; at 1.1 V, the 36-chiplet system achieves a 1.8 GHz PE frequency and 128 TOPS.

3.3 Simba Baseline Tiling

To map DNN layers onto the hierarchical tile-based architecture, we first use a state-of-the-art DNN tiling strategy that uniformly partitions weights spatially, leveraging model parallelism [13, 14, 39, 51, 59, 73]. Listing 2 shows the default tiling in a loop-nest form. Each dimension of a DNN layer can be tiled temporally, spatially, or both at each level of the system hierarchy: package, chiplet, PE, and vector MAC. The loop bounds and orderings in Listing 2 are configurable in Simba so that users can flexibly map computation to the Simba system. In particular, the default dataflow uniformly par-titions weights along the input channel (C) and the output channel (K) dimensions, as noted in the parallel_for loops. In addition, Simba can also uniformly partition along the height (P) and width (Q) dimensions of an output activation across chiplets and PEs to support flexible tiling. Section 5 highlights the limitations of this approach when mapping networks onto a large-scale, non-uniform network access architecture with an MCM-based integration.

We developed a flow that uses Caffe [38] to map a DNN inference application to the Simba system, which primarily determines an efficient tiling strategy for the dataflow that best exploits data reuse in the memory hierarchy. To facilitate evaluation of different map-ping alternatives, we also developed a fast, analytical energy model for Simba that quantifies the energy cost of a particular mapping, similar to the methodology discussed in prior work [15, 25, 50]. The

compilation process starts with a mapper that is provided with data regarding available system resources (including the number of PEs, the number of Global PEs, and the sizes of buffers in the system) and the parameters of a given layer from the Caffe specification. The mapper determines which PE will run each portion of the loop nest and in which buffers the activations and weights are stored. As this mapping is a logical one, the mapper is followed by a placer which decides in which physical resource in the Simba topology the loop nests and data structures are placed. We use a random search algorithm to sample the mapping space and use the energy and per-formance models to select good mappings and placements. Finally, the flow generates the configuration binaries for each chiplet that implement the execution created by the mapper and placer.

4 SIMBA CHARACTERIZATION

This section details the performance characterization of Simba, focusing on achieved scalability using the uniform-tiling baseline. All evaluation results are measured using the prototype system.

4.1 Methodology

Figure 4 shows the experimental setup for measuring the perfor-mance and power of the Simba prototype system. The silicon pro-totype test board is attached to an x86 host through PCI-E using a Xilinx FPGA. To measure the performance of the Simba prototype system, we use software running on the RISC-V to query cycle counters built into the RISC-V microcontrollers. The runtime soft-ware designates one chiplet's RISC-V microcontroller to be the lead RISC-V, which tracks the time from the start of execution until all chiplets complete their assigned work. Power and performance mea-surements begin after the weights have been loaded into each PE's weight buffer and the inputs have been loaded into the Global PE buffers. Measurements conclude after all other partitions have sig-naled their completion to the lead RISC-V. Unless otherwise noted, the chiplets operate at a core voltage of 0.72 V, a PE frequency of 1.03 GHz, and GRS bandwidth of 11 Gbps. We use sense resistors on the board power supplies and a digital acquisition module to measure energy during experiment execution. Since the chiplets support independent clock frequencies for different units (PEs, Global PEs, RISC-V, and NoP routers), we can vary these frequencies to change the compute-to-bandwidth ratios for our experiments. The NoC and NoP routing tables use dimension-ordered X-Y routing for all inter-chiplet communication. Although all 36 Simba chiplets are functional, our evaluation uses 32 chiplets, as it is easier to partition computation by powers of two since the number of input channels (C) and output channels (K) are typically powers of two.

We focus our application measurements on ResNet-50 [33], a state-of-the-art, representative deep-learning network, and evaluate its layers running on the Simba system with a batch size of one, as low-latency inference is a highly critical deployment scenario for data center inferencing [32, 39, 43]. We compile and run each layer independently, except when we map multiple layers to different physical partitions of Simba and execute them in a pipelined manner. Networks are pre-trained and quantized to 8-bit using TensorRT

without accuracy loss [48]. While we focus on ResNet-50 in this paper, we also present measurement results from DriveNet [10] to demonstrate Simba's weak scaling performance, while AlexNet [41]

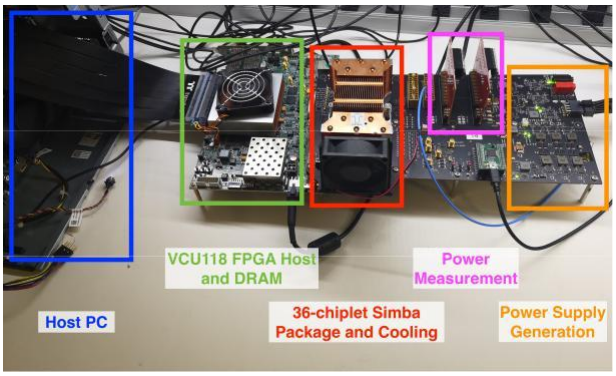


Figure 4: Bench measurement setup for the Simba prototype.

layers exhibit similar behavior. We believe that the diversity of layers in ResNet-50 provides sufficient breadth to cover a wide range of behaviors across different convolutional networks.

4.2 Overview

Figure 5 summarizes performance and energy measurements across all ResNet-50 layers. Each point represents a unique mapping for that layer, while different colors show the number of chiplets active for that mapping. Latency is normalized to a hypothetical best-achievable latency that would be realized if each of the 576 PEs of the system operated with 100% utilization and no communica-tion or synchronization overheads. Simba provides a large number of mapping options with drastically different performance and energy profiles, highlighting the importance of strategies for effi-ciently mapping DNNs to hardware. The figure also demonstrates the highly variable behavior of different layers. For example, the most energy-efficient configurations of layerres3[b-d]_branch2b achieve almost an order of magnitude better efficiency than those of res3a_branch2a. The degree of data reuse highly influences the efficiency; layers with high reuse

factors, e.g., the3 × 3 convolution in 3[b-d]_branch2b, tend to perform computation more efficiently than layers that require more data movement. Finally, although increasing the number of chiplets used in the system improves performance, it also leads to increased energy cost for chiplet-to-chiplet communication and synchronization. Efficiency can drop by nearly an order of magnitude for some layers, which further em-phasizes the effect of data movement on overall efficiency. To better understand system-level trade-offs, the remainder of this section characterizes the sensitivity of Simba to mapping alternatives, layer parameters, bandwidth, latency, and weak scaling, and includes a comparison to modern GPUs.

4.3 Mapping Sensitivity

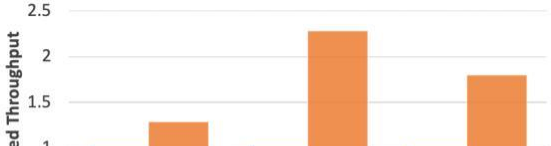
Figure 6 shows a performance comparison of mapping ResNet-50's res4[b-f]_branch2a layer onto multiple PEs, either on-chiplet or spanning multiple chiplets. When mapped to a single chiplet, execution latency decreases linearly from one to eight PEs because of the improved compute throughput with more PEs. At the same time, its performance flattens out beyond eight PEs due to the memory bandwidth contention at the Global PE's SRAM. However,

6 RELATED WORK

DNN inference applications typically run on highly programmable but inefficient CPUs, programmable GPUs with ISA extensions for accelerating tensor operations, or fixed-function DNN inference accelerators. In this work, we present an MCM-based fixed-function DNN inference accelerator prototype system that is capable of run-ning highly complex DNN models at high throughput and low latency. Our aim for Simba is to explore scaling challenges and opportunities without incurring higher inference latency.

Many previous papers have developed hardware accelerator architectures that focus on fast and efficient execution of DL in-ference [2, 13, 15, 30, 51, 52, 57, 59, 61]. In some cases, these accel-erators target small-scale networks and do not consider the chal-lenges associated with scaling to larger networks. In other cases, a proposed accelerator requires structural changes to scale to large high-performance chips. For example, DianNao was originally pro-posed in the context of small DNN layers [13]. Follow-on work with DaDianNao aimed at larger networks by proposing to use a multi-chip network and eDRAM for weights and activations [14]. In contrast, Simba is designed from the ground up for large-scale inference, specifically employing package-level integration.

The TPU [39] is a data-center inference accelerator which lacks the mapping flexibility targeted in Simba . The TPU restricts com-munication to column-wise or row-wise multicast in its systolic computation core, while the communication flexibility in Simba provides the opportunity to perform data-locality-based mapping optimizations. MAERI [42] is a DNN accelerator in which the PEs are controlled by switches at run time; this work does not explore large scale inference with MCMs or chiplets as we do with Simba . MCMs have been explored in CPU design [37, 40, 45, 62, 74] and GPU design [3, 18] to circumvent the fabrication and design costs associated with producing large monolithic chips. Our contribution in Simba is the design,



implementation, and evaluation of the first MCM-based DNN accelerator.

Previous work explored efficient data movement for multi-chip systems on general-purpose workloads [66]. Our work focuses on efficient data movement on MCM-based systems for DNN inference. Tangram [26] provides efficient mappings for tile-based accelerators using shared buffers, loop ordering, and application pipelining. Future work could apply Tangram's mapping techniques to Simba architectures.

7 CONCLUSIONS

This work presents Simba, a scalable MCM-based deep-learning inference accelerator architecture. Simba is a heterogeneous tile-based architecture with a hierarchical interconnect. We developed a silicon prototype system consisting of 36 chiplets that achieves up to 128 TOPS at high energy efficiency. We used the prototype to characterize the overheads of the non-uniform network of an MCM-based architecture, observing that load imbalance and com-munication latencies contribute to noticeable tail-latency effects. We then showed how considering the non-uniform nature of system can help improve performance through techniques such as non-uniform work partitioning, communication-aware data placement, and cross-layer pipelining. Applying these optimizations results in performance increases of up to 16% compared to naive mappings.

Figure 17: Throughput improvement from pipelined execu-tion on three residual blocks of ResNet-50.

ACKNOWLEDGMENTS

The authors would like to thank Frans Sijstermans, Dan Smith, Don Templeton, Guy Peled, Jim Dobbins, Ben Boudaoud, Randall Laperriere, Borhan Moghadam, Sunil Sudhakaran, Zuhair Bokharey, Sankara Rajapandian, James Chen, John Hu, Vighnesh Iyer for package, PCB, signal integrity, fabrication, and prototyping support. This research was, in part, funded by the U.S. Government under the DARPA CRAFT program. 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. Distribution Statement "A" (Approved for Public Release, Distribution Unlimited).

REFERENCES

[1] Jorge Albericio, Patrick Judd, Tayler Hetherington, Tor Aamodt, Natalie Enright Jerger, and Andreas Moshovos. 2016. Cnvlutin: Ineffectual-neuron-free Deep Neural Network Computing. In Proceedings of the International Symposium on Computer Architecture (ISCA).

[2] Manoj Alwani, Han Chen, Michael Ferdman, and Peter Milder. 2016. Fused-layer CNN Accelerators. In Proceedings of the International Symposium on Microarchi-tecture (MICRO).

[3] Akhil Arunkumar, Evgeny Bolotin, Benjamin Cho, Ugljesa Milic, Eiman Ebrahimi, Oreste Villa, Aamer Jaleel, Carole-Jean Wu, and David Nellans. 2017. MCM-GPU: Multi-Chip-Module GPUs for Continued Performance Scalability. In Proceedings of the International Symposium on Computer Architecture (ISCA).

[4] Krste Asanovic, Rimas Avizienis, Jonathan Bachrach, Scott Beamer, David Bian-colin, Christopher Celio, Henry Cook, Daniel Dabbelt, John Hauser, Adam Izraele-vitz, Sagar Karandikar, Ben Keller, Donggyu Kim, John Koenig, Yunsup Lee, Eric Love, Martin Maas, Albert Magyar, Howard Mao, Miquel Moreto, Albert Ou, David A. Patterson, Brian Richards, Colin Schmidt, Stephen Twigg, Huy Vo, and Andrew Waterman. 2016. The Rocket Chip Generator. Technical Re-port UCB/EECS-2016-17. EECS Department, University of California, Berkeley.
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.html

[5] Manu Awasthi, Kshitij Sudan, Rajeev Balasubramonian, and John Carter. 2009. Dynamic Hardware-assisted Software-controlled Page Placement to Manage Capacity Allocation and Sharing within Large Caches. In Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA).

[6] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. 2015. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. CoRR abs/1511.00561 (2015). arXiv:1511.00561

[7] Noah Beck, Sean White, Milam Paraschou, and Samuel Naffziger. 2018. Zeppelin: An SoC for Multichip Architectures. In Proceedings of the International Solid State Circuits Conference (ISSCC).

[8] Bradford M. Beckmann and David A. Wood. 2004. Managing Wire Delay in Large Chip-Multiprocessor Caches. In Proceedings of the International Symposium on Microarchitecture (MICRO).

[9] Nathan Beckmann and Daniel Sanchez. 2013. Jigsaw: Scalable Software-Defined Caches. In Proceedings of the International Conference on Parallel Architectures and Compilation Techniques (PACT).

[10] Mariusz Bojarski, Philip Yeres, Anna Choromanska, Krzysztof Choromanski, Bernhard Firner, Lawrence D. Jackel, and Urs Muller. 2017. Explaining How a Deep Neural Network Trained with End-to-End Learning Steers a Car. CoRR abs/1704.07911 (2017).

[11] Luca P. Carloni, Kenneth L. McMillan, Alexander Saldanha, and Alberto L. Sangiovanni-Vincentelli. 1999. A Methodology for Correct-by-construction La-tency Insensitive Design. In Design Automation Conference.

[12] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. 2016. DeepLab: Semantic Image Segmentation with Deep Convolu-tional Nets, Atrous Convolution, and Fully Connected CRFs. CoRR abs/1606.00915 (2016). arXiv:1606.00915

[13] Tianshi Chen, Zidong Du, Ninghui Sun, Jia Wang, Chengyong Wu, Yunji Chen, and Olivier Temam. 2014. DianNao: A Small-

footprint High-throughput Ac-celerator for Ubiquitous Machine-learning. In Proceedings of the International Conference on Architectural Support for Programming Languages and Operation Systems (ASPLOS).

[14] Yunji Chen, Tao Luo, Shaoli Liu, Shijin Zhang, Liqiang He, Jia Wang, Ling Li, Tian-shi Chen, Zhiwei Xu, Ninghui Sun, et al. 2014. DaDianNao: A Machine-learning Supercomputer. In Proceedings of the International Symposium on Microarchitec-ture (MICRO).

[15] Yu-Hsin Chen, Joel Emer, and Vivienne Sze. 2016. Eyeriss: A Spatial Architecture for Energy-efficient Dataflow for Convolutional Neural Networks. InProceedings of the International Symposium on Computer Architecture (ISCA).

[16] Zeshan Chishti, Michale D. Powell, and T.N. Vijaykumar. 2005. Optimizing Replication, Communication, and Capacity Allocation in CMPs. In Proceedings of the International Symposium on Computer Architecture (ISCA).

[17] Eric Chung, Jeremy Fowers, Kalin Ovtcharov, Michael Papamichael, Adrian Caulfield, Todd Massengill, Ming Liu, Daniel Lo, Shlomi Alkalay, Michael Hasel-man, Christian Boehn, Oren Firestein, Alessandro Forin, Kang Su Gatlin, Mahdi Ghandi, Stephen Heil, Kyle Holohan, Tamas Juhasz, Ratna Kumar Kovvuri, Sitaram Lanka, Friedel van Megan, Dima Mukhortov, Prerak Patel, Steve Rein-hardt, Adam Sapek, Raja Seera, Balaji Sridharan, Lisa Woods, Philip Yi-Xiao, Ritchie Zhao, and Doug Burger. 2017. Accelerating Persistent Neural Networks at Datacenter Scale. In HotChips.

[18] William J. Dally, C. Thomas Gray, John Poulton, Brucek Khailany, John Wil-son, and Larry Dennison. 2018. Hardware-Enabled Artifical Intelligence. In Proceedings of the International Symposia on VLSI Technology and Circuits (VLSI).

[19] Reetuparna Das, Soumya Eachempati, Asit K. Mishra, Vijaykrishnan Narayanan, and Chita R. Das. 2009. Design and Evaluation of A Hierarchical On-chip Inter-connect for Next-Generation CMPs. In Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA).

MICRO-52, October 12–16, 2019, Columbus, OH, USA

[20] Zidong Du, Robert Fasthuber, Tianshi Chen, Paolo Ienne, Ling Li, Tao Luo, Xiaobing Feng, Yunji Chen, and Olivier Temam. 2015. ShiDianNao: Shifting Vision Processing Closer to the Sensor. In Proceedings of the International Symposium on Computer Architecture (ISCA).

[21] Marc Erett, Declan Carey, James Hudner, Ronan Casey, Kevin Geary, Pedro Neto, Mayank Raj, Scott McLeod, Hongtao Zhang, Arianne Roldan, Hongyuan Zhao, Ping-Chuan Chiang, Haibing Zhao, Keehian Tan, Yohan Frans, and Ken Chang. 2018. A 126mW 56Gb/s NRZ Wireline Transceiver for Synchronous Short-reach Applications in 16nm FinFET. In Proceedings of the International Solid State Circuits Conference (ISSCC).

[22] Clément Farabet, Berin Martini, Benoit Corda, Polina Akselrod, Eugenio Culur-ciello, and Yann LeCun. 2011. Neuflow: A Runtime Reconfigurable Dataflow Processor for Vision. In Computer Vision and Pattern Recognition Workshops (CVPRW).

[23] Matthew Fojtik, Ben Keller, Alicia Klinefelter, Nathaniel Pinckney, Stephen G. Tell, Brian Zimmer, Tezaswi Raja, Kevin Zhou, William J. Dally, and Brucek Khailany. 2019. A Fine-Grained GALS SoC with Pausible Adaptive Clocking in 16nm FinFET. In International Symposium on Asynchronous Circuits and Systems (ASYNC).

[24] Jeremy Fowers, Kalin Ovtcharov, Michael Papamichael, Todd Massengill, Ming Liu, Daniel Lo, Shlomi Alkalay, Michael Haselman, Logan Adams, Mahdi Ghandi, Stephen Heil, Prerak Patel, Adam Sapek, G. Weisz, Lisa Woods, Sitaram Lanka, Steve Reinhardt, Adrian Caulfield, Eric Chung, and Doug Burger. 2018. A Con-figurable Cloud-Scale DNN Processor for Real-Time AI. In Proceedings of the International Symposium on Computer Architecture (ISCA).

[25] Mingyu Gao, Jing Pu, Xuan Yang, Mark Horowitz, and Christos Kozyrakis. 2017. Tetris: Scalable and Efficient Neural Network Acceleration with 3D Memory. In Proceedings of the International Conference on Architectural Support for Program-ming Languages and Operation Systems (ASPLOS).

[26] Mingyu Gao, Xuan Yang, Jing Pu, Mark Horowitz, and Christos Kozyrakis. 2019. Tangram: Optimized Coarse-Grained Dataflow for Scalable NN Accelerators. In Proceedings of the International Conference on Architectural Support for Program-ming Languages and Operation Systems (ASPLOS).

[27] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR).

[28] David Greenhill, Ron Ho, David Lewis, Herman Schmit, Kok Hong Chan, Andy Tong, Sean Atsatt, Dana How, Peter McElheny, Keith Duwel, Jeffrey Schulz, Darren Faulkner, Gopal Iyer, George Chen, Hee King Phoon, Han Wooi Lim, Wei-Yee Koay, and Ty Garibay. 2017. A 14nm 1GHz FPGA with 2.5D Transceiver Integration. In Proceedings of the International Solid State Circuits Conference (ISSCC).

[29] Linley Gwennap. 2018. Graphcore Makes Big AI Splash. Microprocessor Report 618 (September 2018).

[30] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A. Horowitz, and William J. Dally. 2016. EIE: Efficient Inference Engine on Compressed Deep Neural Network. In Proceedings of the International Symposium on Computer Architecture (ISCA).

[31] Nikos Hardavellas, Michael Ferdman, Babak Falsafi, and Anastasia Ailamaki. 2009. Reactive NUCA: Near-optimal Block Placement and Replication in Distributed

Caches. In Proceedings of the International Symposium on Computer Architecture (ISCA).

[32] Kim Hazelwood, Sarah Bird, David Brooks, Soumith Chintala, Utku Diril, Dmytro Dzhulgakov, Mohamed Fawzy, Bill Jia, Yangqing Jia, Aditya Kalro, James Law, Kevin Lee, Jason Lu, Peter Noordhuis, Misha Smelyanskiy, Liang Xiong, and Xiangdong Wang. 2018. Applied Machine Learning at Facebook: A Datacenter Infrastructure Perspective. In Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA).

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR).

[34] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.CoRR abs/1704.04861 (2017). arXiv:1704.04861

[35] Jaehyuk Huh, Changky Kim, Hazim Shafi, Lixin Zhang, Doug Burger, and Stephen W. Keckler. 2005. A NUCA Substrate for Flexible CMP Cache Shar-ing. In Proceedings of the International Conference on Supercomputing (ICS).

[36] Subramania S. Iyer. 2016. Heterogeneous Integration for Performance and Scaling. IEEE Transactions on Components, Packaging and Manufacturing Technology 6, 7 (July 2016), 973–982.

[37] Natalie Enright Jerger, Ajaykumar Kannan, Zimo Li, and Gabriel H. Loh. 2014. NoC Architectures for Silicon Interposer Systems. In Proceedings of the Interna-tional Symposium on Microarchitecture (MICRO).

[38] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross B. Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convo-lutional Architecture for Fast Feature Embedding. CoRR abs/1408.5093 (2014).

MICRO-52, October 12–16, 2019, Columbus, OH, USA

arXiv:1408.5093

[39] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. 2017. In-Datacenter Performance Analysis of a Tensor Processing Unit. In Proceedings of the International Symposium on Computer Architecture (ISCA).

[40] Ajaykumar Kannan, Natalie Enright Jerger, and Gabriel H. Loh. 2015. Enabling Interposer-based Disintegration of Multi-core Processors. In Proceedings of the International Symposium on Microarchitecture (MICRO).

[41] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet Classification with Deep Convolutional Neural Networks. In Proceedings of the Conference on Neural Information Processing Systems (NeurIPS).

[42] Hyoukjun Kwon, Ananda Samajdar, and Tushar Krishna. 2018. MAERI: Enabling Flexible Dataflow Mapping over DNN Accelerators via Programmable Interconnects. In Proceedings of the International Conference on Architectural Support for Programming Languages and Operation Systems (ASPLOS).

[43] Yann LeCun. 2019. The Next Challenge in AI: Self-Supervised Learning. In Proceedings of the International Solid State Circuits Conference (ISSCC).

[44] Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. 2015. Deep Learning. Nature 521 (2015), 436–444.

[45] Gabriel H. Loh, Natalie Enright Jerger, Ajaykumar Kannan, and Yasuko Eckert. 2015. Interconnect-Memory Challenges for Multi-chip, Silicon Interposer Systems. In Proceedings of the International Symposium on Memory System (MEMSYS).

[46] Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully Convolutional Networks for Semantic Segmentation. In Proceedings of the Conference on Com-puter Vision and Pattern Recognition (CVPR).

[47] Azalia Mirhoseini, Hieu Pham, Quoc V. Le, Benoit Steiner, Rasmus Larsen, Yue-feng Zhou, Naveen Kumar, Mohammad Norouzi, Samy Bengio, and Jeff Dean. 2017. Device Placement Optimization with Reinforcement Learning. In Proceed-ings of the International Conference on Machine Learning (ICML).

[48] NVIDIA 2018. NVIDIA TensorRT: Programmable Inference Accelerator. https: //developer.nvidia.com/tensorrt.

[49] NVIDIA. 2019. NVIDIA Tesla Deep Learning Product Performance. https://

developer.nvidia.com/deep-learning-performance-training-inference.

[50] Angshuman Parashar, Priyanka Raina, Yakun Sophia Shao, Yu-Hsin Chen, Victor A. Ying, Anurag Mukkara, Rangharajan Venkatesan, Brucek Khailany, Stephen W. Keckler, and Joel Emer. 2019. Timeloop: A Systematic Approach to DNN Accelerator Evaluation. In Proceedings of the International Symposium on Performance Analysis of Systems and Software (ISPASS).

[51] Angshuman Parashar, Minsoo Rhu, Anurag Mukkara, Antonio Puglielli, Rang-harajan Venkatesan, Brucek Khailany, Joel Emer, Stephen W. Keckler, and William J. Dally. 2017. SCNN: An Accelerator for Compressed-sparse Con-volutional Neural Networks. In Proceedings of the International Symposium on Computer Architecture (ISCA).

[52] Wajahat Qadeer, Rehan Hameed, Ofer Shacham, Preethi Venkatesan, Christos Kozyrakis, and Mark A. Horowitz. 2013. Convolution Engine: Balancing Effi-ciency & Flexibility in Specialized Computing. In Proceedings of the International Symposium on Computer Architecture (ISCA).

[53] Brandon Reagen, Paul Whatmough, Robert Adolf, Saketh Rama, Hyunkwang Lee, Sae Kyu Lee, José Miguel Hernández-Lobato, Gu-Yeon Wei, and David Brooks. 2016. Minerva: Enabling Low-power, Highly-accurate Deep Neural Network Ac-celerators. In Proceedings of the International Symposium on Computer Architecture (ISCA).

[54] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You Only Look Once: Unified, Real-time Object Detection. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR). 779–788.

[55] Kirk Saban. 2012. Xilinx Stacked Silicon Interconnect Technology De-livers Breakthrough FPGA Capacity, Bandwidth, and Power Efficiency. http://www.xilinx.com/support/documentation/white_papers/wp380_Stacked_ Silicon_Interconnect_Technology.pdf.

[56] Ali Shafiee, Anirban Nag, Naveen Muralimanohar, Rajeev Balasubramonian, John Paul Strachan, Miao Hu, R. Stanley Williams, and Vivek Srikumar. 2016. ISAAC: A Convolutional Neural Network Accelerator with In-situ Analog Arith-metic in Crossbars. In Proceedings of the International Symposium on Computer Architecture (ISCA).

Shao, et al.

[57] Hardik Sharma, Jongse Park, Naveen Suda, Liangzhen Lai, Benson Chau, Vikas Chandra, and Hadi Esmaeilzadeh. 2018. Bit Fusion: Bit-Level Dynamically Com-posable Architecture for Accelerating Deep Neural Network. In Proceedings of the International Symposium on Computer Architecture (ISCA).

[58] Yongming Shen, Michael Ferdman, and Peter Milder. 2017. Maximizing CNN Accelerator Efficiency Through Resource Partitioning. InProceedings of the Inter-national Symposium on Computer Architecture (ISCA).

[59] Frans Sijstermans. 2018. The NVIDIA Deep Learning Accelerator. In Hot Chips.

[60] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-scale Image Recognition. CoRR abs/1408.1556 (2014). arXiv:1408.1556

[61] Linghao Song, Xuehai Qian, Hai Li, and Yiran Chen. 2017. PipeLayer: A Pipelined ReRAM-Based Accelerator for Deep Learning. In Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA).

[62] Dylan Stow, Yuan Xie, Taniya Siddiqua, and Gabriel H. Loh. 2017. Cost-effective Design of Scalable High-performance Systems using Active and Passive Inter-posers. In International Conference on Computer-Aided Design (ICCAD).

[63] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In Proceedings of the Conference on Neural Information Processing Systems (NeurIPS).

[64] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going Deeper with Convolutions. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR).

[65] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the Inception Architecture for Computer Vision. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR).

[66] Xulong Tang, Orhan Kislal, Mahmut Kandemir, and Mustafa Karakoy. 2017. Data Movement Aware Computation Partitioning. In Proceedings of the International Symposium on Microarchitecture (MICRO).

[67] Po-An Tsai, Nathan Beckmann, and Daniel Sanchez. 2017. Jenga: Software-Defined Cache Hierarchies. In Proceedings of the International Symposium on Computer Architecture (ISCA).

[68] Po-An Tsai, Changping Chen, and Daniel Sanchez. 2018. Adaptive Scheduling for Systems with Asymmetric Memory Hierarchies. In Proceedings of the International Symposium on Microarchitecture (MICRO).

[69] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. 2016. WaveNet: A Generative Model for Raw Audio. CoRR abs/1609.03499 (2016). arXiv:1609.03499

[70] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. CoRR abs/1706.03762 (2017). arXiv:1706.03762

[71] Swagath Venkataramani, Ashish Ranjan, Subarno Banerjee, Dipankar Das, Sasikanth Avancha, Ashok Jagannathan, Ajaya Durg, Dheemanth Nagaraj, Bharat Kaul, Pradeep

Dubey, and Anand Raghunathan. 2017. ScaleDeep: A Scalable Com-pute Architecture for Learning and Evaluating Deep Networks. In Proceedings of the International Symposium on Computer Architecture (ISCA).

[72] John M. Wilson, Walker J. Turner, John W. Poulton, Brian Zimmer, Xi Chen, Sudhir S. Kudva, Sanquan Song, Stephen G. Tell, Nikola Nedovic, Wenxu Zhao, Sunil R. Sudhakaran, C. Thomas Gray, and William J. Dally. 2018. A 1.17pJ/b 25Gb/s/pin Ground-referenced Single-ended Serial Link for Off- and On-package Communication in 16nm CMOS Using a Process- and Temperature-adaptive Voltage Regulator. In Proceedings of the International Solid State Circuits Conference (ISSCC).

[73] Xuan Yang, Mingyu Gao, Jing Pu, Ankita Nayak, Qiaoyi Liu, Steven Bell, Jeff Setter, Kaidi Cao, Heonjae Ha, Christos Kozyrakis, and Mark Horowitz. 2018. DNN Dataflow Choice Is Overrated. CoRR abs/1809.04070 (2018). arXiv:1809.04070

[74] Jieming Yin, Zhifeng Lin, Onur Kayiran, Matthew Poremba, Muhammad Shoaib Bin Altaf, Natalie Enright Jerger, and Gabriel H. Loh. 2018. Modular Routing Design for Chiplet-based Systems. In Proceedings of the International Symposium on Computer Architecture (ISCA).

[75] Shijin Zhang, Zidong Du, Lei Zhang, Huiying Lan, Shaoli Liu, Ling Li, Qi Guo, Tianshi Chen, and Yunji Chen. 2016. Cambricon-X: An Accelerator for Sparse Neural Networks. In Proceedings of the International Symposium on Microarchi-tecture (MICRO).

[76] Brian Zimmer, Rangharajan Venkatesan, Yakun Sophia Shao, Jason Clemons, Matthew Fojtik, Nan Jiang, Ben Keller, Alicia Klinefelter, Nathaniel Pinckney, Priyanka Raina, Stephen G. Tell, Yanqing Zhang, William J. Dally, Joel S. Emer, C. Thomas Gray, Stephen W. Keckler, and Brucek Khailany. 2019. A 0.11 pJ/Op, 0.32-128 TOPS, Scalable Multi-Chip-Module-based Deep Neural Network Acceler-ator with Ground-Reference Signaling in 16nm. In Proceedings of the International Symposia on VLSI Technology and Circuits (VLSI).