A 0.32-128 TOPS, Scalable Multi-Chip-Module-based Deep Neural Network Inference Accelerator with Ground-Referenced Signaling in 16nm

Brian Zimmer, *Member, IEEE*, Rangharajan Venkatesan, *Member, IEEE*, Yakun Sophia Shao, *Member, IEEE*, Jason Clemons, *Member, IEEE*, Matthew Fojtik, *Member, IEEE*, Nan Jiang, *Member, IEEE*, Ben Keller, *Member, IEEE*, Alicia Klinefelter, *Member, IEEE*, Nathaniel Pinckney, *Member, IEEE*, Priyanka Raina, *Member, IEEE*, Stephen G. Tell, *Member, IEEE*, Yanqing Zhang, *Member, IEEE*, William J. Dally, *Fellow, IEEE*, Joel S. Emer, *Fellow, IEEE*, C. Thomas Gray, *Senior Member, IEEE,* Stephen W. Keckler, *Fellow, IEEE*, and Brucek Khailany, *Senior Member, IEEE*

***Abstract*— Custom accelerators improve the energy efficiency, area efficiency, and performance of deep neural network (DNN) inference. This work presents a scalable DNN accelerator consisting of 36 chips connected in a mesh network on a multi-chip-module (MCM) using ground-referenced signaling (GRS). While previous accelerators fabricated on a single monolithic chip are optimal for specific network sizes, the proposed architecture enables flexible scaling for efficient inference on a wide range of DNNs, from mobile to data center domains. Communication energy is minimized with large on-chip distributed weight storage and a hierarchical network-on-chip and network-on-package, and inference energy is minimized through extensive data reuse. The 16nm prototype achieves 1.29 TOPS/mm^2 area efficiency, 0.11 pJ/op (9.5 TOPS/W) energy efficiency, 4.01 TOPS peak performance for a 1-chip system, and 127.8 peak TOPS and 1903 images/s ResNet-50 batch-1 inference for a 36-chip system.**

***Index Terms*— Ground-referenced signaling, single-ended signaling, multi-chip modules, deep neural networks, inference accelerator.**

I. Introduction

Deep neural networks (DNNs) are extremely popular and have been widely adopted to solve problems in a huge variety of fields, including image recognition [1]–[3], semantic segmentation [4], language translation [4], and autonomous driving [5]. DNN inference is currently performed on a variety of traditional computing systems including CPUs, FPGAs, and GPUs, which provide different trade-offs between efficiency, cost, performance and programmability.

Due to the deterministic structure of DNNs, fixed-function accelerators have the potential to further improve area efficiency, energy efficiency, and performance relative to CPUs and GPUs [6]–[12]. However, the absolute performance requirements of DNNs vary from tiny networks on energy-constrained edge devices to large networks in data centers. It is extremely expensive to build a separate chip for each of these applications, as each domain has widely different compute and memory bandwidth requirements. Additionally, in such a rapidly changing field, it is difficult to predict DNN requirements years in advance, the lead time required to develop a custom accelerator for a target market.

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).

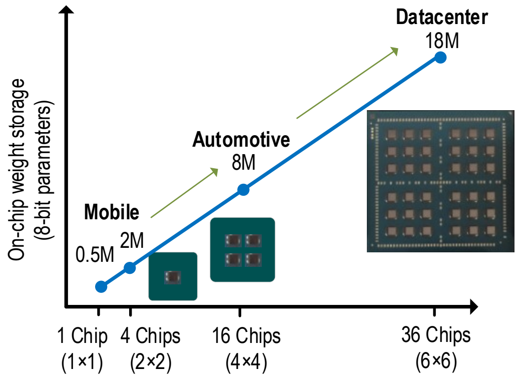

Figure 1: Composing an MCM from various number of chiplets addresses different performance requirements with a single chip.

The goal of this work is to design a system that can easily scale across all application requirements with a single silicon chip. The main idea is to build one small chip and assemble and connect a variable number of these chips together on a package to form a multi-chip module, or MCM, as shown in Figure 1. MCM-based systems offer a variety of benefits [13], [14]. Large designs are sometimes reticle limited, and MCMs provide a method to increase system capacity without requiring board-level or system-level integration. Smaller chips have higher yield, as a fixed number of defects per wafer breaks fewer total chips as the number of chips on a wafer grows. Smaller chips are simpler and easier to design. A system can mix silicon dice from different process nodes to improve design reuse and reduce cost. The acceleration of new networks can be achieved by quickly and easily repackaging existing chips into an optimally sized system without waiting for the development and fabrication of new chips. Despite all these advantages, MCMs have area, performance, and power penalties relative to a large monolithic chip because chip-to-chip communication is more expensive than on-chip communication. The proposed system limits this penalty through energy-efficient chip-to-chip links, a hierarchical interconnect network, algorithmic

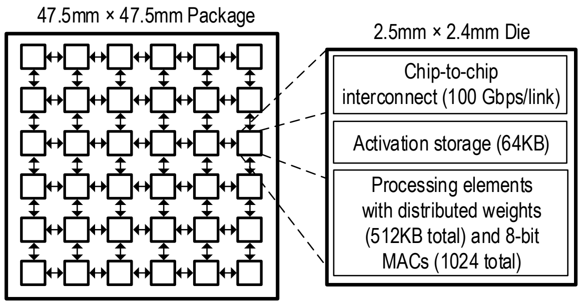
47.5mm × 47.5mm Package
2.5mm × 2.4mm Die
Chip-to-chip interconnect (100 Gbps/link)
Activation storage (64KB)
Processing elements with distributed weights (512KB total) and 8-bit MACs (1024 total)

Figure 2: Prototype system with 36 chips connected on a package.

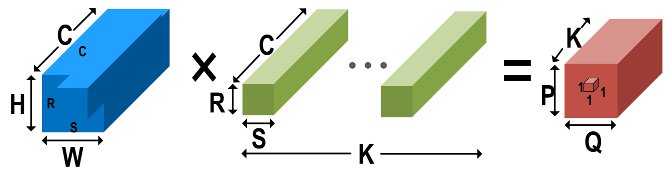

Figure 3: Generation of one output element during convolution.

optimizations to exploit the latency insensitive nature of architecture, and flexible dataflow mapping.

Implementing an MCM-based neural network inference accelerator posed some unique challenges. In order to achieve good performance for modern networks such as ResNet-50 [3], the scale of the system must be very large—compared to recently published 8-bit neural network accelerators [6]–[9], the prototype system has 15-160× the on-chip SRAM storage, and over 67-1300× the performance. The system was designed to achieve strong scaling, in which increased compute capacity directly reduces latency, instead of weak scaling, where larger batches are used to perform more work in parallel but with the same latency. Because there is only one unique chip in the system, the architecture must be efficient for both small single-chip configurations and huge 36-chip configurations.

The prototype system, shown in Figure 2, combines 36 identical chips on an organic package to form a large-scale neural network accelerator. Each chip on the package is 2.5mm by 2.4mm, and 36 are connected on a 47.5mm by 47.5mm organic substrate. Each chip, described in Section II, has 752KB of total SRAM storage and can perform 1,024 multiply-and-accumulates (MACs) per cycle to execute smaller DNNs individually. Each package, described in Section III, combines 36 dice together in a mesh network with 100GB/s interconnect in each direction to execute larger networks with 22.5MB of on-chip SRAM and 36,864 MACs per cycle. Performance ranges from 4 TOPS for a 1-chip system to 128 TOPS for a 36-chip system. The chip was fabricated in the TSMC 16nm FinFET process and implemented with an agile design methodology further discussed in Section IV. Experimental results in Section V discuss energy-efficiency and performance measurements for a peak performance benchmark and ResNet-50.

II. Single-Chip Architecture

Each chip in the system can operate as a standalone neural network accelerator for smaller networks.

A. System Operation

DNNs are composed of a series of many layers, including convolutional layers, pooling layers, activation layers, and fully-connected layers. Each layer processes an input activation tensor from the previous layer and creates an output activation tensor for the next layer. The key workload kernel executed by this neural network accelerator is the convolution of a single layer shown in Figure 3. An input activation tensor with size H×W and C input channels is convolved with a weight tensor with size R×S and C channels. After striding R×S across H×W, an output of size P×Q is formed. K different weight kernels contribute to each of the K output channels in the output activation tensor. Each element of the output activation tensor is formed from the MAC of R×S×C elements from the input and weight tensors, and this MAC is repeated P×Q×K times. Layer dimensions vary for each layer in the DNN, so the total number of MAC operations required can vary from 0.6M-14M in DriveNet [5] to 50M-100M in ResNet-50 [3]. These MAC operations need to be distributed spatially across multiple MAC datapaths and temporally within each datapath.

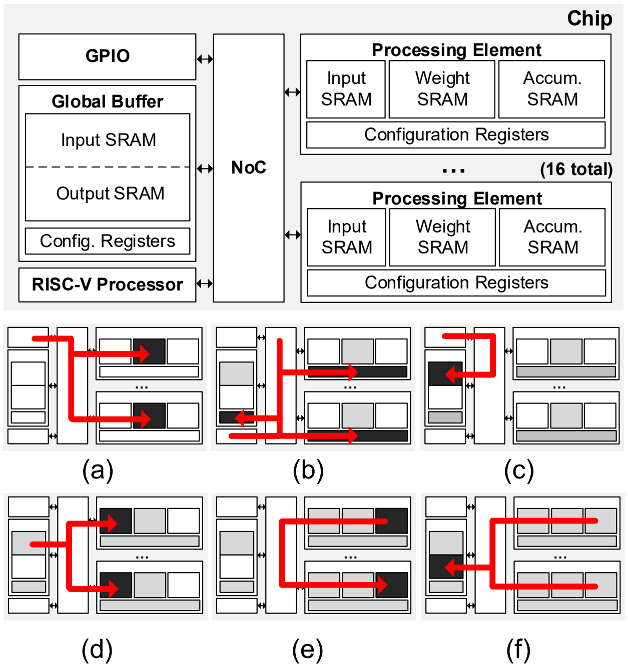

Figure 4: Flow of data across the NoC router to execute a layer.

The system diagram shown in Figure 4 describes the data movement required to execute a convolution. Weight tensors are loaded once at boot through general-purpose input-output (GPIO) from a host and distributed across the network-on-chip (NoC) to SRAMs inside each of the 16 processing elements (PEs) (Figure 4a). Each weight tensor can be split among the PEs if each PE computes separate input/output channels or replicated on different PEs so that each PE can work on a different portion of the output activation in parallel. Each PE and the global buffer (GB) operate autonomously after a go command from the RVP, and are controlled by local configurable state machines (Figure 4b). Input activation tensors are loaded through GPIO and are stored in large SRAMs in the GB (Figure 4c). The GB sends the input activation tensor to the PEs, and the tensor can be split among the PEs along a layer dimension (such as input channel) or replicated so that each PE can compute a different output channel in parallel (Figure 4d). Each PE performs 64 MACs per cycle for a total of 1024 MACs per cycle in the chip. Accumulation of intermediate sums occurs either within the PE or across

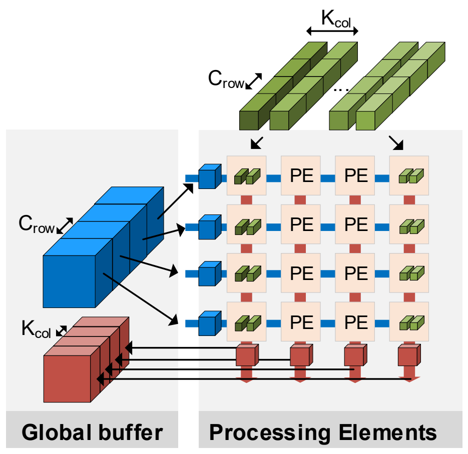

Figure 5: Example mapping splits input channels along rows and output channels along columns.

multiple PEs (Figure 4e). Each PE sends its respective portion of the output activation tensor to the GB to finish layer computation (Figure 4f).

B. Mapping Convolutions

Figure 5 describes one possible mapping of a convolution onto the chip. The input channels are split into four parts, and each part is multicast to the four PEs in each row of the PE array. The weight tensors are allocated between the four columns of the PE array and replicated into each PE along the column. All 16 PEs perform their MACs on the partitioned volumes in parallel. Then, the partial sums of the top row are accumulated row by row until the final row of PEs has accumulated contributions from all C channels back together to form the final output activation.

Valid mappings and communication overheads are determined by the layer dimensions as well as the PE and GB SRAM sizes. The architecture was designed to be very flexible and allows tiling across many different dimensions to achieve high efficiency across layers with very different dimensions. Layers with large K benefit from partitioning weight tensors between PEs and multicasting the input activation to each PE at the cost of more input activation traffic. Layers with large H×W benefit from replicating the weight kernels to minimize input activation traffic at the cost of greater weight storage requirements. Layers with large C benefit from splitting the input channels across PEs to minimize input traffic and weight storage at the cost of output accumulation traffic. The key to this architecture is that the weights remain stationary and are reused across multiple inputs. Only input activations, output partial sums, and output activations need to be transported across the NoC.

C. Network-on-Chip (NoC)

The on-chip network, shown in Figure 6, serves as the transport layer for network transactions between the various blocks in the system. Each transaction can be one of three types: streaming data, interrupts, or AXI transactions [15]. Streaming data transactions are used to send input activations, partial sums, and output activations between the PEs and GB. Interrupts are single-flit packets generated by the PEs and GB and sent to the RVP to signal completion of a layer. AXI transactions are used for all other reads and writes of registers

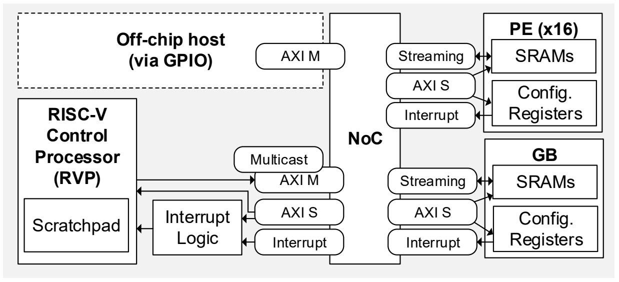

Figure 6: Communication interface capabilities of the NoC.

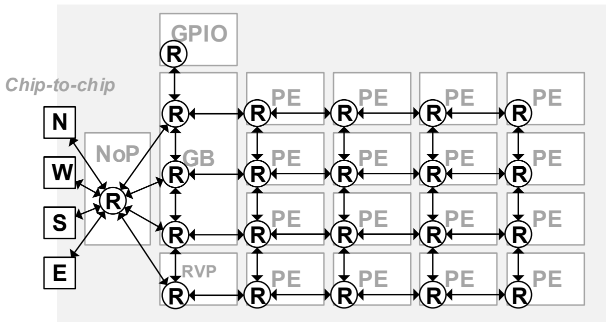

Figure 7: Units are connected in a mesh NoC.

and memory in the system. A protocol similar to 64-bit AXI4-Lite supports bursts up to 8 words in length to allow the RVP to fetch entire cache lines with a single transaction. All architecturally visible state across the multi-chip system is globally addressable, including control registers, PE and GB buffers, and the RVP scratchpad. Only the RVP implements AXI masters that can initiate requests, while the PEs and GBs implement AXI slaves to service requests to local state. To simplify the heavily reused AXI slave in the PE, no write responses are generated anywhere in the system.

Custom hardware extensions to the AXI master block enable the RVP to exploit features of the system that reduce communication latency during runtime. A portion of the global address space is reserved for multicast requests: by writing to a particular global address, the RVP's AXI write is converted into a multicast packet that writes the same data to the same local address of a configurable subset of the PEs on the chip or the RVPs in the MCM system. The RVP's hardware interrupt lines are also memory-mapped, so that a write to a particular address can trigger an interrupt.

The NoC routers are connected in a mesh network as shown in Figure 7. Each NoC transaction is encoded in a packet that is composed of one or more 66-bit flits. A flit is composed of 64 bits of data and two bits of flit identification, indicating a header, body, or tail flit. Singleton flits are indicated by setting both the header and tail bits. The header flit's 64 bits of data are for packet routing and other metadata; subsequent flits contain the packet payload. The NoC supports both unicast and multicast traffic, and routes are specified in the header flit with a bit indicating whether the packet is unicast or multicast. Multicast is one-hot encoded and can address all 36 chips in the system and 20 NoC destinations; this information consumes 56 bits header bits. Unicast packet destinations are binary encoded, consuming 6 bits for network-on-package (NoP) destinations and 5 for NoC destinations. Unicast headers thus have available header bits to encode additional packet-specific information; only certain types of packets can be multicast due to the large bitwidth of the one-hot multicast address.

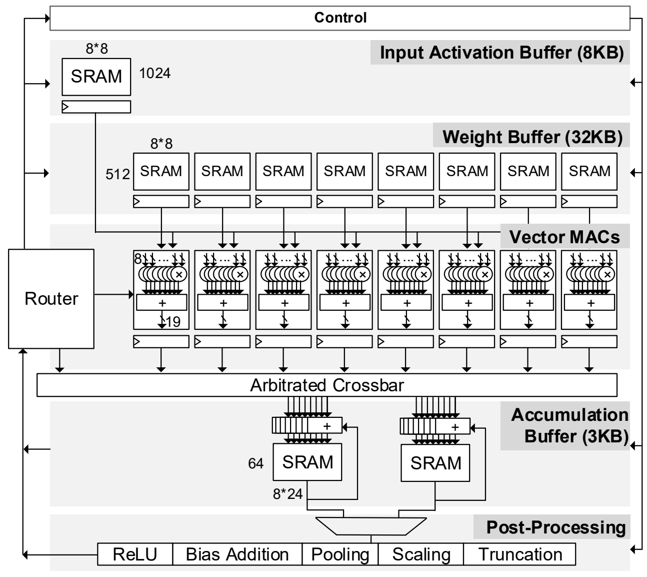

Figure 8: Architecture of the processing element (PE).

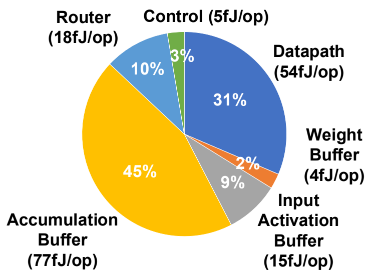

Figure 9: Simulated power breakdown of the processing element at 0.72V.

The NoC router implementation uses cut-through routing with credit-based flow control and is pipelined to operate at full throughput and 2 cycles of latency. At 0.72V, each link of the NoC achieves 70 Gbps bandwidth.

D. *Processing Element (PE)*

Most computation is done by the processing element (PE) in Figure 8, which executes convolutional layers, fully-connected layers, and post-processing functions like bias addition, ReLU, and pooling. There are 8 lanes in the PE, and each lane uses a different weight tensor to generate elements for a separate output channel (K). Within each lane, an 8-bit precision vector MAC multiplies 8 input elements from separate input channels (C) with 8 weight elements and sums them to calculate a single output value. As the numbers of input and output channels (C and K) generally range from 64 to 1024, performing vector operations in sets of 8 elements is very efficient. Local input activation, output activation, and accumulation SRAMs buffer data for the datapath. Minimizing accesses to these SRAMs is critical to maximizing energy efficiency. The input activation SRAM is read every cycle, but the energy cost is amortized by distributing each element to 8 lanes. The weight SRAM is much wider than the input activation SRAM as it needs to supply a separate vector of values to each lane in the datapath, but the weights remain constant for multiple inputs, so the values are reused P×Q times. The accumulation SRAM is written every cycle to hold partial sums, but energy is amortized by writing the accumulation of the 8-wide vector of C channels. The output size P×Q is generally larger than the number of entries in the accumulation buffer, so computation is temporally tiled to generate a subset of output activation dimensions at a time. The accumulation buffer can also be written through the router from other PEs to perform cross-PE reduction when the weight kernel is split between multiple PEs. Once the full accumulation is complete, each PE performs the final post-processing functions such as ReLU, bias addition, pooling, scaling, or truncation to compute the final output activation.

PE power was simulated on the post-synthesis gate-level netlist using activity traces from a representative workload, and a breakdown of PE power is shown in Figure 9 at 0.72V. The largest consumer of energy is the accumulation buffer SRAM, due to the writing of 192 bits every cycle. The second largest consumer is the datapath, which performs the MAC operations. Since the input buffer output is shared between multiple lanes, and the weight buffer output is used for many cycles, the contribution of these two SRAMs is small. PE energy efficiency could be optimized by using a generator to explore the design space of many different possible PE dataflows and precisions for different DNNs [16].

E. *Global Buffer (GB)*

As input activations are generally multicast to multiple PEs, and output activations collected from multiple PEs, the global buffer acts as a second level in the memory hierarchy to store these activations on chip. The GB SRAM is partitioned into four 16KB banks which can be flexibly partitioned between input and output activations. The GB includes three routers that provide higher bandwidth into the NoC. In addition to managing activations, the global buffer can perform some forms of computation (such as element-wise computation) locally, without needing to send data to PEs.

F. *RISC-V Processor (RVP)*

The RISC-V core is an RV64IMAC implementation of the open-source Rocket Chip Generator [17] based on the `SmallCore` instance. The RVP includes a 16KiB instruction cache, an 8KiB data cache, and a 16KiB scratchpad. 128 external interrupt lines are accessible via the NoC, which can trigger them either via interrupt transactions or AXI writes.

G. *General Purpose Input-Output (GPIO) and JTAG*

The chip uses a narrow, low-speed GPIO interface to communicate with a host for the purpose of loading weights, input activations, and RISC-V runtime software. The GPIO interface uses a divided on-chip clock and a ready-valid protocol to communicate with an FPGA, which recovers the clock and performs skew alignment. JTAG is used to configure GPIO, clocking, and routing tables, toggle reset, and provide observability of key signals for debug.

III. Multi-Chip Architecture

A full MCM-based neural network accelerator is formed by connecting 36 dice together on the package in a mesh network as shown in Figure 2. There are two general strategies to utilize the increased computation capacity to improve throughput: increase parallelism or pipeline multiple layers in the system. When increasing parallelism, the system executes a single layer at a time as it does in the one-chip architecture, except computation is split between dice in the same manner as computation was split between PEs in the one-chip case. The

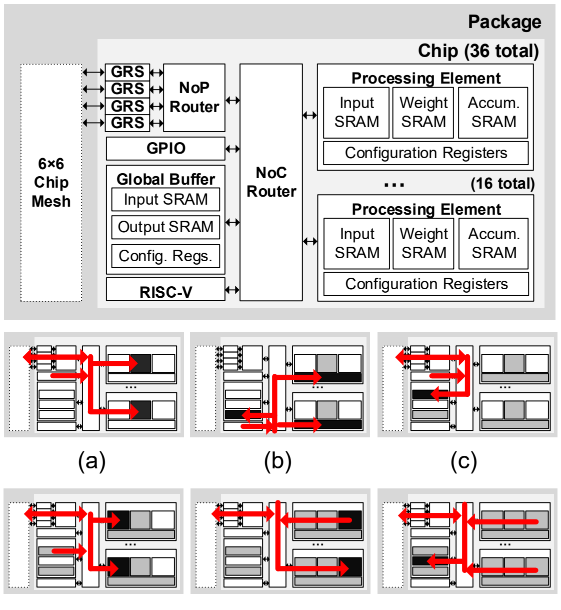

Figure 10: Flow of data across the NoP router to execute a layer.

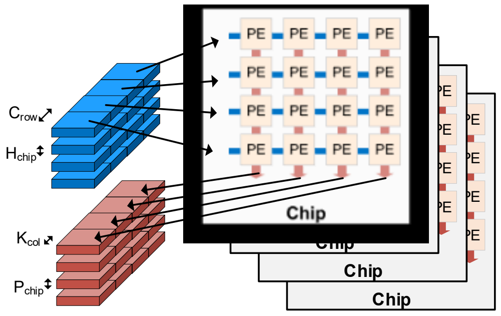

Figure 11: Mapping a convolution to an example 4-chip system.

latency of the layer computation is reduced (strong scaling), so more layers can be executed per unit time, improving throughput. An alternative strategy pipelines multiple layers, with groups of chips executing different layers simultaneously and forwarding their results to the next group. Pipelining improves throughput but does not decrease the latency of layer computation (weak scaling). While pipelining can improve the overall utilization when there is limited parallelism in a layer [18], this work focuses on the increased layer parallelism strategy to understand the limits of strong scaling.

A. System Operation

The system diagram shown in Figure 10 describes the data movement required to execute a convolution on multiple chips. Weight tensors are loaded once at boot through GPIO from the host subsystem and distributed across the NoP and NoC to SRAMs inside each of the 576 processing elements (PEs) (Figure 10a). Each weight tensor can be split among the chips when each chip computes a separate output channel or replicated on different chips so that each chip can work on a different portion of the input activation in parallel. The RVPs on each chip configure state machines within each PE and the GB to match the layer dimensions. Input activation tensors are loaded through GPIO from a host subsystem and are stored in large SRAMs in the GB (Figure 10b).

To initiate layer execution, a lead RVP multicasts a go command to the other worker RVPs, which then multicast local start commands to every unit on the chip (Figure 10c). The GB sends the input activation tensor to the local PEs across the NoC as well as remote PEs across the NoP (Figure 10d). Each PE performs 64 MACs per cycle for a total of 36,864 MACs per cycle in the package. Accumulation of intermediate sums to compute the final output element occurs either within the PE, across multiple PEs in one chip, or across multiple PEs in one package (Figure 10e). Then each PE sends their respective portion of the output activation tensor to the local or remote GB and the layer's computation is finished (Figure 10f). The RVPs across the system synchronize at the completion of each layer using an interrupt-based barrier system controlled by the lead RVP. The worker RVPs wait for interrupts from all local units that indication completion of work. Once received, the worker RVPs send an interrupt to the lead RVP. Meanwhile, the lead RVP first waits for all local interrupts, then waits for interrupts from the other RVP participating in the computation. The lead RVP uses built-in counters to time the execution of each layer.

B. Mapping Convolutions

The mapping strategy across multiple chips in a package is almost identical to mapping across multiple PEs in a chip, except that traffic flows across the NoP in addition to the NoC, and there are now multiple GBs in the system. Mapping remains flexible, and computation can be split along any dimension between chips (K, P, Q, R, S, H, W, and C). Figure 11 shows an example of mapping a layer onto a 4-chip system. Each chip computes a subset of the output rows (P), so each chip stores the corresponding input activation rows (H) in their local GB and the weights are replicated across all 4 chips. Within each chip, input channels (C) are split between rows and output channels (K) between columns as described in Figure 5. Each PE executes 8 K and 8 C per cycle, each chip operates on 4 segments of K and 4 segments of C in parallel, and each package operates on 4 segments of P in parallel for 4,096 MACs per cycle. The other layer dimensions are looped over temporally to complete the convolution. Detailed descriptions and studies of various mapping strategies and measurements of resulting latencies can be found in [18].

C. Network-on-Package (NoP)

Simply connecting the edges of the single-chip NoC mesh to adjacent chips in one large multi-chip mesh would inhibit scalability. Instead, a hierarchical network connects 36 chips in a 6-by-6 chip mesh network-on-package (NoP), and each chip's NoP router connects to 4 NoC local ports as shown in Figure 7. The NoP routers use the same design and packet format as the NoC routers. NoP routing tables are configurable via JTAG-configurable lookup tables. Careful consideration has been given to avoid various deadlock scenarios that can arise from a hierarchical network shared by both unicast and multicast traffic. Multicast-multicast deadlocks are avoided by enforcing cut-through flow control, in which a packet can only advance to the next router when there is enough buffering for the entire packet. Multicast-unicast routing deadlocks are avoided by using the Base-Routing-Conformed-Path model [19], in which multicast and unicast routing tables are programmed such that they share the same network paths. This ensures that as long as

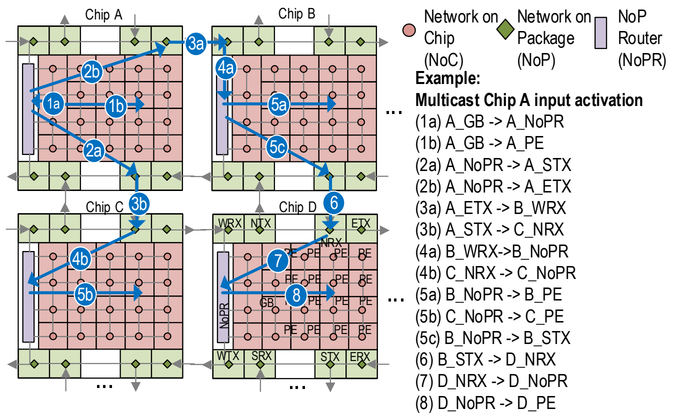

Figure 12: Example multicast operation between 4 chips.

the unicast routing algorithm is deadlock-free, all possible interaction between unicast and multicast is also deadlock-free.

Figure 12 shows an example input activation multicast operation in a 4-chip system. The GB on chip A sends data path into local PEs through the NoC and other chips through the NoP. The data moves to chips B and C through the chip-to-chip interconnect, and arrives at the local GB, where it is forwarded to chip D and deposited into the local NoCs to send to local PEs.

D. Chip-to-Chip Ground-Referenced Signaling (GRS)

The scalability of the MCM-based accelerator relies on efficient chip-to-chip communication within the NoP. To achieve high bandwidth and energy efficiency, each chip in the package is connected with single-ended ground-referenced signaling (GRS) [20]. To implement a package mesh, every chip has 8 chip-to-chip GRS transceivers, where 4 are configured as transmitters (TX) and 4 as receivers (RX) and communicate to adjacent chips in a mesh. Each TX and RX pair has 4 data wires and one forwarded clock wire as shown in Figure 13. Each signal is ground-referenced instead of differential, only requiring one bump, and the transmitters drive a low-swing signal of about 200mV around ground to improve energy efficiency. Each link has configurable equalization and termination at both the receiver and transmitter. An alternative MCM technology to GRS across organic substrates is the silicon interposer [13], which allows much finer pitch bumps, but the wires support much lower data rates and the expense is impractical for many markets. Unlike silicon interposer transceivers, GRS can communicate with other packages through the PCB at the same speed with the same circuits, so the prototype system size is not limited to a single package and could scale further through either denser packing of chips on the package or multiple packages on a PCB.

Since each GRS link is unidirectional, credit-based flow control is used where credits are returned using the GRS link running in the opposite direction. Data sent from the NoP router to the GRS TX is written to a 15-word, 128-bit FIFO memory using the NoP router clock. This same memory is read as a 32-word, 60-bit FIFO by a 1.56 GHz GRS word clock, such that the same FIFO memory is used for both clock domain crossing and splitting the data into 60-bit words. Four bits of header are added per word to signify when data is valid or partially valid and to pass credits. The 64-bit words are then sent to high speed 16:1 serializers to drive the 4 data wires. This process is

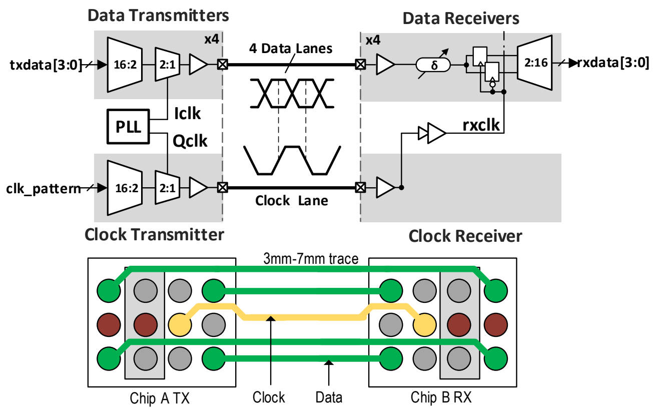

Figure 13: Chips communicate on the package using ground-referenced signaling (GRS).

reversed in the GRS RX to reconstitute data for the receiving NoP router.

Buffering of full packets is used along the GRS NoP and GPIO interfaces, in the former case to ensure that no stalling occurs mid-packet when sending across the GRS interface, and in the latter case to ensure that the much slower GPIO interface does not tie up routing resources while flits are being transmitted or received off-chip. Total packet length is limited by these interfaces to 17 flits (1 header and 16 payload).

IV. MCM SoC Implementation

The 6mm^2 inference accelerator [21] was fabricated in a TSMC 16nm FinFET process, and 36 chips were assembled on a 12-layer organic substrate.

A. HLS-based Agile Design Methodology

The testchip was designed with a high-productivity VLSI design methodology [22], which enabled 24-hour turnaround from design changes to a tape-out-ready GDS. Most of the design was described in C++ using an open-source library of commonly used micro-architectural components called MatchLib [23] and synthesized into Verilog using an industry-standard high-level synthesis (HLS) tool. The design was intentionally modularized into partitions of around 200,000 gates that avoid tight communication or timing constraints to other units by using latency-insensitive (LI) channels. The main partitions in the design were the PE, GB, RVP, NoP, and GRS, shown as rectangles in the floorplan in Figure 14, and were implemented independently from each other in parallel to improve turnaround time. Partitioning the design into smaller units increases the number of cross-unit boundaries, while larger units increase place-and-route runtime. An agile hardware implementation flow using fully automated synthesis and place-and-route tools provided daily feedback about timing, area, and power consumption as the design was optimized. ECOs were avoided entirely by making changes directly in the source code and reimplementing the entire unit. In parallel with VLSI trials, the entire design was prototyped on an FPGA so that software development occurred in parallel with hardware development, which revealed many critical bugs well before tape-out. Overlapping architectural design, VLSI implementation, and software design ensured that effort was focused on improving the final quality-of-result.

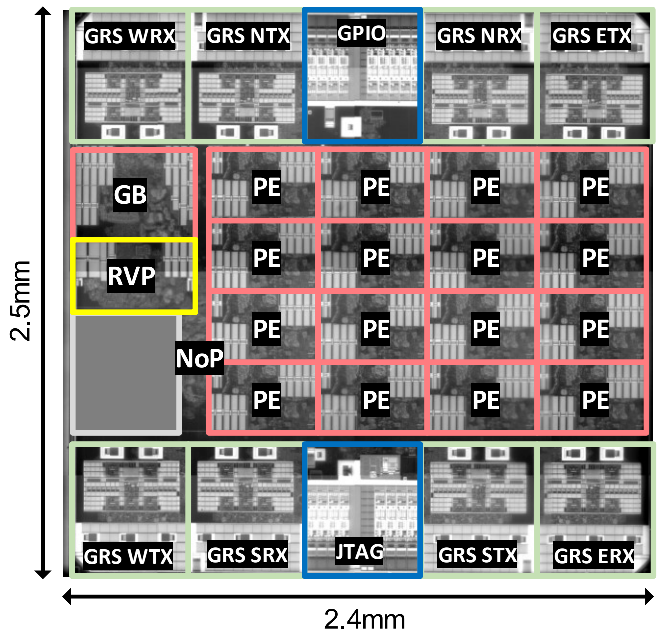

Figure 14: Chip micrograph annotated with the floorplan of the chip.

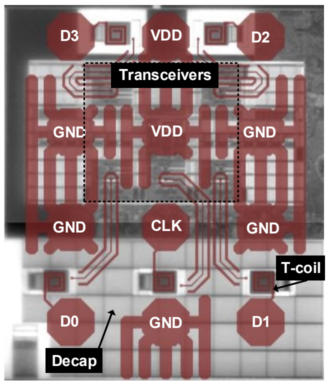

Figure 15: Floorplan of the chip-to-chip GRS interconnect partition.

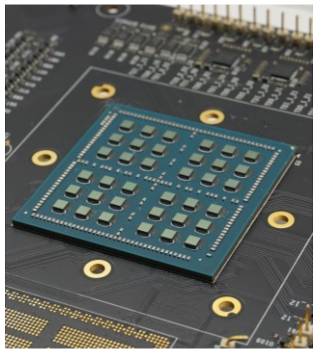

Figure 16: Prototype system connects 36 chips on the package.

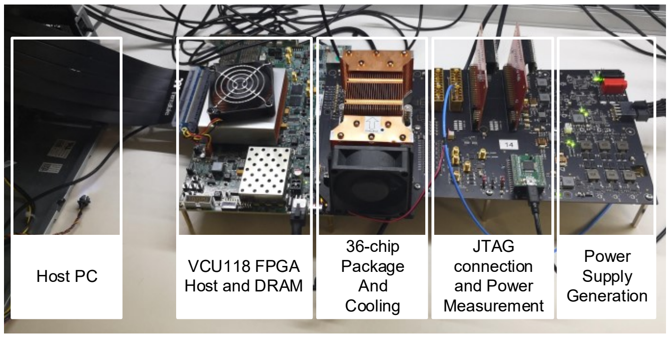
Host PC
VCU118 FPGA Host and DRAM
36-chip Package And Cooling
JTAG connection and Power Measurement
Power Supply Generation

Figure 17: Bench measurement setup.

B. Floorplan

The physical floorplan in Figure 14 largely reflects the logical NoC mesh network shown in Figure 7. Physical partition reuse was critical to reducing design effort. Each of the 16 PE partitions are identical and are designed so that their IO pins connect by abutment. The off-chip communication partitions (GRS, JTAG, and GPIO) are placed on the edge of the chip to avoid disturbing power delivery to the PEs in the center of the chip. The 8 GRS partitions are identical and designed so that they can be mirrored across the X and Y dimension while still abutting correctly to the power grid. The GRS partition floorplan, shown in Figure 15, contains the custom layout transceivers in the center. The connections to the 140µm pitch bumps are made with length-matched and shielded RDL. The t-coils, ESD devices, decoupling capacitance, and link calibration circuits are implemented with a digital place-and-route flow. The JTAG and GPIO partitions use standard 1.8V IO devices to communicate off-chip. The NoP contains the most difficult timing paths because it synchronously communicates with every GRS macro, so it requires careful pipelining and clock distribution.

C. Clocking

Each partition in the design is clocked by an adaptive clock generator in the center of the partition and is asynchronous to other partitions. The latency cost of synchronization between clock domains is mitigated with the use of pausible bisynchronous FIFOs [24][25]. Each partition can run at independent frequencies, so physically large partitions such as the NoP can run at a slower frequency than the PEs.

A JTAG interface is used to configure the chip during the boot process. Every partition has a separate JTAG tap controller to avoid synchronous paths between partitions, and the entire chip consists of a chain of 31 controllers with the JTAG signals serially snaked through the chip. Reset is toggled through the JTAG interface and is synchronized into each local clock domain. A high-frequency reference clock for GRS (1.56GHz), a low frequency testing clock for measurement circuits in each partition (100MHz), and a global on-chip clock are distributed as a tree from the JTAG partition.

V. Experimental Results

A. Test Setup

Figure 16 shows the prototype package with 36 chips and Figure 17 shows the bench measurement setup. The test package is mounted on a custom PCB with voltage regulators, clock generators, and power measurement circuitry. The test board connects via FMC to a Xilinx VCU118 FPGA board, which is connected to a host PC via PCIE. The FPGA communicates with the prototype via the GPIO interface of one of the chips. The FPGA fabric implements an AXI interconnect that shares the global memory map of the prototype system, allowing the RVPs to access FPGA state that includes 4GB of DRAM. To execute an inference operation, a RISC-V program, which includes all weights, input activations, and configuration settings, is loaded into FPGA DRAM. The RVPs then execute the program, fetching from DRAM and loading state into the PEs and GBs before initiating layer execution.

B. NoP Performance

Each data lane operates at a configurable speed between 11-25Gbps and consumes 0.82-1.75pJ/bit. Compared to previous interconnect on organic substrates for MCM systems [14], GRS has about 3.5× higher bandwidth per chip area and lower energy per bit. Measurements show an eye opening of 0.7UI at 25Gbps.

C. Peak Performance

Table 2 compares our system to prior inference accelerators with 8-bit precision running a peak performance benchmark

Table 1: Comparison to other inference accelerators for peak performance benchmark.

	B. Moons, ENVISION [7]	Z. Yuan, STICKER [8]	J.Lee, UNPU [9]	J. Song, Exynos [6]	Proposed*		
					1 Chip	4 Chip (2 × 2)	36 Chip (6 × 6)
Technology	28nm	65nm	65nm	8nm	16nm		
Cumulative Core Area	1.87 mm^2	7.8 mm^2	13 mm^2	5.5 mm^2	3.1 mm^2	12.4 mm^2	111.6 mm^2
Cumulative Chip Area	unknown	12 mm^2	16 mm^2	unknown	6 mm^2	24 mm^2	216 mm^2
Precision	4b,8b,16b	8b	1-16b	8b,16b	8b		
On-Chip SRAM (MB)	0.14	0.17	0.25	1.53	0.625	2.5	22.5
Supply Voltage (V)	1	0.67-1.1	0.63-1.1	0.5-0.8	0.41-1.2	0.52-1.2	0.52-1.1
Frequency (MHz)	200	200	5-200	67-933	161-2001	515-1998	484-1797
Core Power (mW)	165	21-248	3.2-297	39-1,553	30-4160	630-16,420	5,310-106,090
GRS Power† (mW)	n/a	n/a	n/a	n/a	n/a	215-220	3,840-4,090
MACs per cycle	512 @8b	256	1,728@8b	1,024	1,024	4,096	36,864
Performance (TOPS)	~0.15@8b	0.1	0.69@8b	1.91	0.32-4.01	3.93-15.7	32.5-127.8
Core Energy Efficiency (pJ/op)	~1.1@8b	0.96	~0.18@8b	0.087@8b	0.105-1.04	0.160-1.05	0.164-8.30
Core Area Efficiency (TOPS/mm^2)	0.08	0.013	0.053	0.35	0.10-1.29	0.32-1.27	0.29-1.15

* Measured results reported for 40% density weights and input activations †11Gbps mode

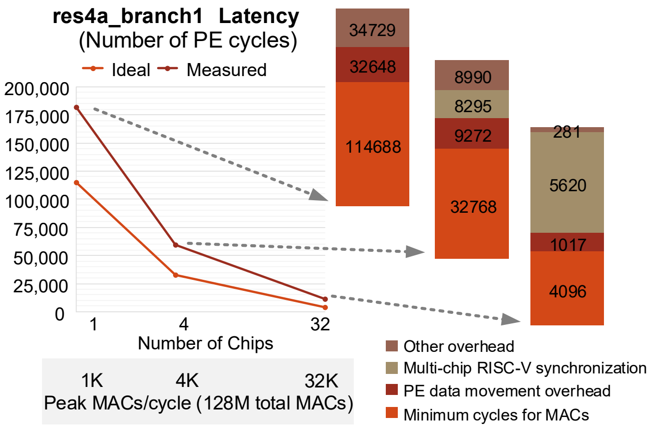

Figure 18: Increasing the number of chips decreases the latency of layer execution.

that saturates MACs on each chip. The digital core efficiency numbers exclude chip-to-chip interconnect power for comparison purposes. GRS consumes constant power, even with no traffic, as there is no sleep mode in this prototype. At peak performance mode, this fixed power overhead is less than 5% of overall power, but it becomes more significant at minimum voltage. Overall the prototype achieves around between 67× and 1280× higher TOPS, 0.8-10× higher energy efficiency, and 4-100× higher area efficiency.

D. Application Measurements

Table 2 demonstrates the architecture's scalability with measured performance of a 32-chip datacenter-scale system running each layer of ResNet-50 [3]. GRS chip-to-chip interconnect enables multi-chip scalability while consuming 12% of the total energy, and a batch size of one minimizes inference latency and energy.

E. Strong Scaling

Figure 18 shows that measured latency in terms of PE cycles is reduced by 16× when executing the res4a_branch1 layer of ResNet-50 [3] on 32 chips instead of 1 chip. One chip maintains 63% utilization of the MAC units. With 32 chips, the computation is spread across so many PEs that the number of cycles spent doing computation is only 4096 cycles, and 6000 cycles of synchronization between chips across the NoC starts to dominate runtime. Overall, a 32-chip system can execute 128

Table 2: Measurement of a 36-chip system running ResNet-50 at 0.80V.

Layer	Latency (µS)	Core Energy (µJ)	GRS Energy (µJ)
conv1-pool1	41.00	902.90	147.70
res2a_branch1	8.87	209.00	32.02
res2a_branch2a	6.44	141.21	23.26
res2[a-c]_branch2b	9.26	250.84	33.40
res2[a-c]_branch2c	8.87	209.00	32.02
res2[b-c]_branch2a	14.04	417.68	50.56
res3a_branch1	8.92	281.39	32.15
res3a_branch2a	7.59	199.90	27.41
res3[a-d]_branch2b	9.11	237.57	32.91
res3[a-d]_branch2c	8.18	220.74	29.52
res3[b-d]_branch2a	8.40	232.08	30.29
res4a_branch1	8.11	264.19	29.21
res4a_branch2a	6.06	154.99	21.87
res4[a-f]_branch2b	11.98	302.36	43.35
res4[a-f]_branch2c	6.64	187.68	23.94
res4[b-f]_branch2a	6.86	194.77	24.77
res5a_branch1	12.49	326.72	45.18
res5a_branch2a	21.09	464.69	76.28
res5[a-c]_branch2b	13.33	349.58	48.20
res5[a-c]_branch2c	7.38	181.21	26.74
res5[b-c]_branch2a	8.23	203.74	29.78
fc1000	3.32	27.37	3.29
Total (batch=1)	0.525 ms	16.3 mJ/image	2.33 mJ/image

million MACs in 11µs, and design improvements to the synchronization scheme could further improve strong scaling.

VI. Conclusion

This work presents a scalable DNN inference accelerator that uses MCM assembly of multiple chips on an organic substrate to improve yield, reduce design cost, and address different market segments with a single chip. Scalability is enabled by a flexible multi-chip architecture and hierarchical NoC and NoP. The 36-chip system achieves high energy efficiency (9.5 TOPS/W), high area efficiency (1.29 TOPS/mm^2), and high performance (128 TOPS).

Acknowledgement

The authors would like to thank 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.

References

[1] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. B. Wojna, “Rethinking the Inception Architecture for Computer Vision,” 2016.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, 2012, pp. 1097–1105.

[3] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[4] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation: 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part VII,” 2018, pp. 833–851.

[5] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, “End to End Learning for Self-Driving Cars,” *arXiv:1604.07316*, 2016.

[6] J. Song, Y. Cho, J.-S. Park, J.-W. Jang, S. Lee, J.-H. Song, J.-G. Lee, and I. Kang, “7.1 An 11.5TOPS/W 1024-MAC Butterfly Structure Dual-Core Sparsity-Aware Neural Processing Unit in 8nm Flagship Mobile SoC,” in *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*, 2019, pp. 130–132.

[7] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, “Envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable Convolutional Neural Network processor in 28nm FDSOI,” in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, 2017, pp. 246–247.

[8] Z. Yuan, J. Yue, H. Yang, Z. Wang, J. Li, Y. Yang, Q. Guo, X. Li, M.-F. Chang, H. Yang, and Y. Liu, “Sticker: A 0.41-62.1 TOPS/W 8Bit Neural Network Processor with Multi-Sparsity Compatible Convolution Arrays and Online Tuning Acceleration for Fully Connected Layers,” in *2018 IEEE Symposium on VLSI Circuits*, 2018, pp. 33–34.

[9] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, “UNPU: A 50.6TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision,” in *2018 IEEE International Solid-State Circuits Conference (ISSCC)*, 2018, pp. 218–220.

[10] 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*, 2016, pp. 243–254.

[11] Y. Chen, S. Member, T. Krishna, J. S. Emer, V. Sze, and S. Member, “Eyeriss : An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks,” pp. 1–12, 2016.

[12] N. P. Jouppi *et al.*, “In-Datacenter Performance Analysis of a Tensor Processing Unit,” in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, 2017, pp. 1–12.

[13] K. Saban, “Xilinx Stacked Silicon Interconnect Technology Delivers Breakthrough FPGA Capacity, Bandwidth, and Power Efficiency,” 2012. [Online]. Available: https://www.xilinx.com/support/documentation/white_papers/wp380_Stacked_Silicon_Interconnect_Technology.pdf.

[14] N. Beck, S. White, M. Paraschou, and S. Naffziger, “‘Zeppelin’: An SoC for multichip architectures,” in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, 2018, pp. 40–42.

[15] “AMBA ® AXI and ACE Protocol Specification AXI3, AXI4, AXI5, ACE and ACE5,” 2017. [Online]. Available: arm.com.

[16] R. Venkatesan, Y. S. Shao, M. Wang, J. Clemons, S. Dai, M. Fojtik, B. Keller, A. Klinefelter, N. Pinckney, P. Raina, Y. Zhang, B. Zimmer, W. J. Dally, J. Emer, S. W. Keckler, and B. Khailany, “MAGNet: A Modular Accelerator Generator for Neural Networks,” in *2019 International Conference On Computer Aided Design (ICCAD), to appear*, 2019.

[17] K. Asanovic *et al.*, “The Rocket Chip Generator,” in *EECS Department, University of California, Berkeley, Technical Report*, 2016, no. UCB/EECS-2016-17.

[18] Y. S. Shao, J. Clemons, R. Venkatesan, B. Zimmer, M. Fojtik, N. Jiang, B. Keller, A. Klinefelter, N. Pinckney, P. Raina, S. Tell, Y. Zhang, W. Dally, J. Emer, C. Gray, B. Khailany, and S. W. Keckler, “Simba: Scaling Deep-Learning Inference with Multi-Chip-Module-Based Architecture,” in *2019 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), to appear*, 2019.

[19] D. K. Panda, S. Singal, and R. Kesavan, “Multidestination message passing in wormhole k-ary n-cube networks with base routing conformed paths,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 10, no. 1, pp. 76–96, Jan. 1999.

[20] J. W. Poulton, J. M. Wilson, W. J. Turner, B. Zimmer, X. Chen, S. S. Kudva, S. Song, S. G. Tell, N. Nedovic, W. Zhao, S. R. Sudhakaran, C. T. Gray, and W. J. Dally, “A 1.17-pJ/b, 25-Gb/s/pin ground-referenced single-ended serial link for off- and on-package communication using a process- and temperature-adaptive voltage regulator,” *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 43–54, 2019.

[21] B. Zimmer, R. Venkatesan, Y. S. Shao, J. Clemons, M. Fojtik, N. Jiang, B. Keller, A. Klinefelter, N. Pinckney, P. Raina, S. G. Tell, Y. Zhang, W. J. Dally, J. S. Emer, C. T. Gray, S. W. Keckler, and B. Khailany, “A 0.11 pJ/Op, 0.32-128 TOPS, Scalable Multi-Chip-Module-based Deep Neural Network Accelerator with Ground-Reference Signaling in 16nm,” in *2019 Symposium on VLSI Circuits*, 2019, pp. C300–C301.

[22] B. Khailany, E. Krimer, R. Venkatesan, J. Clemons, J. S. Emer, M. Fojtik, A. Klinefelter, M. Pellauer, N.

Pinckney, Y. S. Shao, S. Srinath, C. Torng, S. (Likun) Xi, Y. Zhang, and B. Zimmer, "A modular digital VLSI flow for high-productivity SoC design," in *Proceedings of the 55th Annual Design Automation Conference on - DAC '18*, 2018, pp. 1–6.

[23] "MatchLib." [Online]. Available: https://github.com/NVlabs/matchlib.

[24] B. Keller, M. Fojtik, and B. Khailany, "A Pausible Bisynchronous FIFO for GALS Systems," in *2015 21st IEEE International Symposium on Asynchronous Circuits and Systems*, 2015, pp. 1–8.

[25] M. Fojtik, B. Keller, A. Klinefelter, N. Pinckney, S. G. Tell, B. Zimmer, T. Raja, K. Zhou, W. J. Dally, and B. Khailany, "A Fine-Grained GALS SoC with Pausible Adaptive Clocking in 16 nm FinFET," in *2019 25th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, 2019.