SMIV: A 16-nm 25-mm^2 SoC for IoT With Arm Cortex-A53, eFPGA, and Coherent Accelerators

Sae Kyu Lee, *Member, IEEE*, Paul N. Whatmough, *Member, IEEE*, Marco Donato, *Member, IEEE*, Glenn G. Ko, *Member, IEEE*, David Brooks, *Fellow, IEEE*, and Gu-Yeon Wei, *Senior Member, IEEE*

***Abstract*— Emerging Internet of Things (IoT) devices necessitate system-on-chips (SoCs) that can scale from ultralow power always-on (AON) operation, all the way up to less frequent high-performance tasks at high energy efficiency. Specialized accelerators are essential to help meet these needs at both ends of the scale, but maintaining workload flexibility remains an important goal. This article presents a 25-mm^2 SoC in 16-nm FinFET technology which demonstrates targeted, flexible acceleration of key compute-intensive kernels spanning machine learning (ML), DSP, and cryptography. The *SMIV* SoC includes a dedicated AON sub-system, a dual-core Arm Cortex-A53 CPU cluster, an SoC-attached embedded field-programmable gate array (eFPGA) array, and a quad-core cache-coherent accelerator (CCA) cluster. Measurement results demonstrate: 1) 1236× power envelope, from 1.1 mW (only AON cluster), up to 1.36 W (whole SoC at maximum throughput); 2) 5.5–28.9× energy efficiency gain from offloading compute kernels from A53 to eFPGA; 3) 2.94× latency improvement using coherent memory access (CCA cluster); and 4) 55× MobileNetV1 energy per inference improvement on CCA compared to the CPU baseline. The overall flexibility-efficiency range on SMIV spans measured energy efficiencies of 1× (dual-core A53), 3.1× (A53 with SIMD), 16.5× (eFPGA), 54.9× (CCA), and 256× (AON) at a peak efficiency of 4.8 TOPS/W.**

***Index Terms*— Deep neural networks (DNNs), embedded field-programmable gate array (eFPGA), hardware accelerators, Internet of Things (IoT), machine learning (ML), system-on-chip (SoC).**

Manuscript received March 23, 2021; revised June 14, 2021, July 25, 2021, and September 20, 2021; accepted September 20, 2021. Date of publication October 11, 2021; date of current version January 28, 2022. This article was approved by Associate Editor Vivek De. This work was supported in part by the U.S. Government, through the Defense Advanced Research Projects Agency (DARPA), the Circuit Realization at Faster Timescales (CRAFT), and the Power Efficiency Revolution for Embedded Computing Technologies (PERFECT) Programs, in part by the NSF under Award 1551044 and Award 1718160, in part by Arm Inc., and in part by Intel Corporation. *(Sae Kyu Lee and Paul N. Whatmough contributed equally to this work.) (Corresponding author: Paul N. Whatmough.)*

Sae Kyu Lee was with the School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138 USA. He is now with IBM Research, NY 10598 USA (e-mail: saekyu.lee@ibm.com).

Paul N. Whatmough is with Arm Research, Boston, MA 02451 USA, and also with the School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138 USA (e-mail: pwhatmough@eecs.harvard.edu).

Marco Donato is with the Department of Electrical and Computer Engineering, Tufts University, Medford, MA 02155 USA (e-mail: marco.donato@tufts.edu).

Glenn G. Ko, David Brooks, and Gu-Yeon Wei are with the School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138 USA (e-mail: gko@seas.harvard.edu; dbrooks@eecs.harvard.edu; gywei@g.harvard.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSSC.2021.3115466.

Digital Object Identifier 10.1109/JSSC.2021.3115466

I. Introduction

INTERNET of Things (IoT) devices have rapidly become ubiquitous and are shaping new use cases in both consumer and industrial markets. Machine learning (ML) inference on data arising from various sensor modalities often constitutes the key functionality, enabling detection of interesting or anomalous events capture *in situ*, in real time. ML also takes an increasingly important role in enabling evolved user interfaces (UIs) driven by simple speech commands or hand gestures, which are tailored to small and cheap IoT device form factors with limited physical controls. IoT devices are typically battery-powered, and therefore, hardware accelerators have become a key enabling technology to provide the energy efficiency required for not only ML inference, but also DSP, cryptography, and a host of other demanding algorithms. They also often demand a very wide performance range, with a large proportion of the time typically spent idle, periodically performing low-complexity tasks such as data logging.

CPUs have by far the greatest flexibility and the most widely supported programming model. However, CPU throughput and energy efficiency on compute-intensive tasks are limited, even with single-instruction multiple-data (SIMD) instruction extensions. In contrast, dedicated memory-mapped accelerators offer orders of magnitude higher throughput and energy efficiency for a more narrow range of workloads. Some of the optimizations implemented by deep neural network (DNN) accelerators include small data types (e.g., 1–8 bits [1]), aggressive reuse of operands in local SRAM [2], [3], and exploiting sparsity [4]. Other circuit techniques previously reported include mixed-signal datapaths [5], in-memory architectures [6], and timing error tolerance [7]. Industry activity is also proliferating [8]–[12].

Nonetheless, the limited flexibility of specialized accelerators introduces an elevated risk of hardware obsolescence. This is a particular concern with DNNs, which have evolved rapidly [13]. In addition to this, accelerators inflate software development cycles, as the programming model is inflexible with a limited set of semantics and often requires explicit memory management. Therefore, we seek to better understand the *flexibility–efficiency* trade-offs, in order to understand how to build efficient system-on-chips (SoCs) comprising heterogeneous accelerators.

In this article, we describe *SMIV* [14], [15], a heterogeneous SoC for IoT devices (Fig. 1). The SoC comprises four main compute clusters: more specifically, a mobile-class CPU and

three distinct accelerators. The three accelerators together support a broad range of target compute kernels, albeit with a little overlap, while representing very distinct trade-offs in terms of programmability and efficiency, which we enumerate. The main contributions of this work are summarized below.

1) *Embedded field-programmable gate array (eFPGA) cluster:* We demonstrate the utility of small eFPGAs of a few mm^2 integrated into an IoT SoC, through offloading common compute kernels (both arithmetic and bit-wise dominant) from the CPU to eFPGA (Section II).
2) *Cache-coherent datapath accelerators (CCA):* Data movement is a key consideration in accelerator design. We propose the CCA approach, which provides optimized DNN compute primitives via a cache-coherent interface to reduce the data-movement cost and provide a coherent programming model (Section III).
3) *Always-on (AON) subsystem:* IoT devices typically spend the vast majority of the time in an inactive or partially active state. We propose an ultralow-power AON subsystem which provides enough performance to carry out small control and DNN inference tasks independently, while the rest of the SoC is powered down, including main memory (Section IV).

In the remainder of this article, we first highlight two of the key compute clusters integrated in *SMIV*: the eFPGA (Section II) and the cache-coherent accelerator (CCA) (Section III). The SoC organization and main compute and memory system components are described in Section IV, while Sections V and VI present the chip implementation and measurement result, respectively. Finally, Section VII concludes the article.

II. Embedded FPGA

FPGAs are a well-established semiconductor product, widely used in prototyping, military, and telecommunications [16]. Traditional standalone FPGAs occupy an interesting middle ground between CPUs and application-specific integrated circuits (ASICs), with more performance and efficiency than the former. FPGAs can be effective for bit-level operations implemented using lookup tables (LUTs), and with the addition of MAC datapaths, algorithms such as DSP filters and transforms also map well.

A. eFPGA in SoCs

In this work, we incorporate an eFPGA macro [17] as an accelerator resource on the SoC. This occupies the middle ground between the fully software programmable CPUs (A53) and the specialized hardware accelerators (AON and CCA). The eFPGA can potentially implement a huge range of functions within the SoC, including IO multiplexing, direct memory access (DMA) engines, compression, encryption/decryption, sorting, and so on. Compared to implementing such tasks on a CPU in software, eFPGA may offer improved performance and efficiency at the cost of a longer development time. Fig. 2(a) illustrates the FlexLogix eFPGA, in a 2×2 array, with two logic tiles and two DSP tiles. The logic tile [Fig. 2(a)] includes 2.5 K six-input LUTs arranged into logic

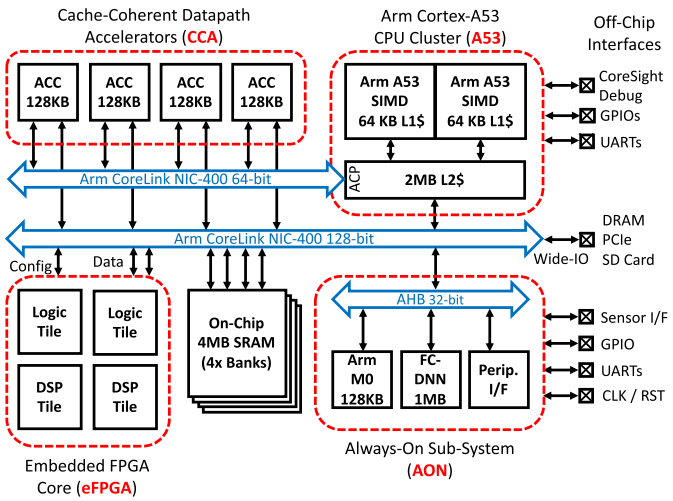


Fig. 1. SMIV SoC block diagram, showing main components.

compute elements (CEs) and interconnected with a boundaryless mixed-radix interconnect [17]. The DSP tiles include 40× 22-bit DSP datapaths with less programmable logic (1.88 K six-input LUTs).

B. Array Size and Tile Mix

The array size determines the peak performance available. For example, Fig. 2(b) shows simulated throughput for three MAC-dominant workloads (Section VI-C) increasing fairly linearly with array size. In addition to this, place and route tools often struggle to achieve high utilization on small arrays of one or two tiles. Nonetheless, large arrays are expensive in terms of silicon area, as each tile is $\sim$1 mm^2 in 16 nm.

The mix of DSP and logic tiles is also important. Designs with predominantly combinational logic favor an array of logic tiles, which have more LUTs than DSP tiles. However, designs with integer arithmetic datapaths greatly benefit from hardware DSP tiles, because implementing the datapaths in LUTs is expensive. Therefore, the ratio of logic and DSP tiles should ideally match the intended workload mix. For example, the finite-impulse response (FIR) and fast Fourier transform (FFT) designs [Fig. 2(b)] are simple pipelines with streaming data, and hence, a single logic tile is sufficient to achieve close to 100% DSP utilization as we add more DSP tiles. However, the generic matrix multiplication (GEMM) design is based on a systolic array with more complex data movement, and hence, the number of logic tiles must be increased as more DSP tiles are added.

In this work, we implemented an array with two DSP tiles and two logic tiles, as a reasonable compromise between throughput on datapath-dominated designs [Fig. 2(b)], while providing an abundance of LUTs to explore other use cases.

C. Programming Latency

Naturally, the compelling feature of an eFPGA is that it can be reconfigured on demand; either throughout the device lifetime or even during different phases of execution. However, in the latter case, there may be a balance between

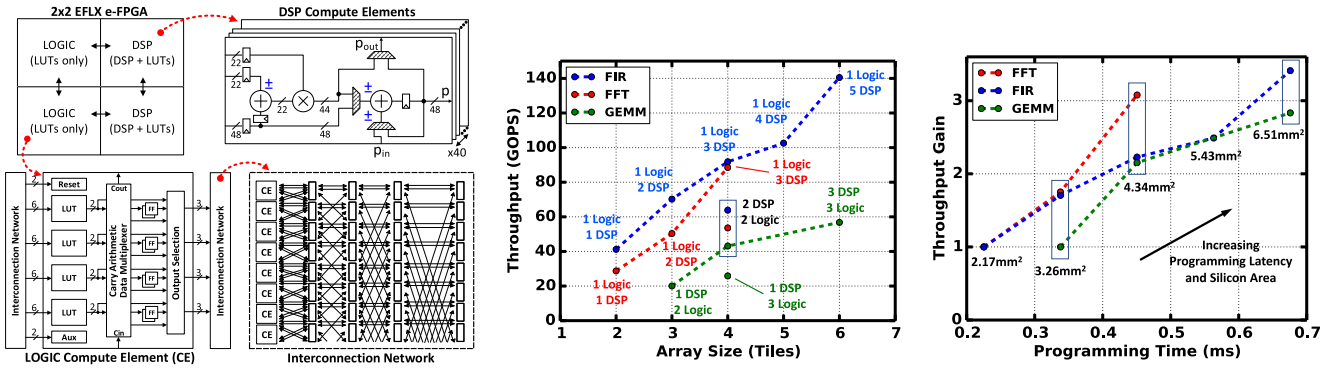


Fig. 2. eFPGA: (a) array with two logic and two DSP tiles, with details of DSP compute, logic compute, and interconnection network, (b) simulated throughput for three typical workloads (Section VI-C), showing pareto-optimal logic/DSP mix (dashed lines) varies by workload, and (c) programming time and silicon area increases with array size.

achievable throughput and programming latency, since the time required to program the array is proportional to the array size. Fig. 2(c) shows how programming latency and silicon area increase with throughput for the FFT, FIR, and GEMM kernels, assuming that the bitstream data is stored in DRAM and accessed via the synthesizable interface to off-chip main memory (Section IV-G).

D. eFPGA Programming Effort

Compute functions can be implemented using either an overlay architecture,[1] or a fixed function design [19]. The designs in Fig. 2 were coded using two different approaches. The FIR design was implemented in hand-optimized RTL, designed to achieve maximum DSP utilization. While the GEMM systolic array was implemented as a modular RTL implementation, with scalable resource usage. In these two cases, the number of lines of Verilog (Table III) is actually close to the C/C++ implementations. For the FFT RTL implementation, we used a Verilog IP generator [20], which generates fairly verbose RTL. There is also significant interest in using high-level synthesis (HLS) from C++/SystemC [21], potentially making eFPGA more accessible to software developers.

III. Cache Coherent Accelerators

Accelerator integration is an oft neglected aspect of accelerator research, but is essential to achieving good system performance. Many accelerator test chips implement standalone hardware, tested in isolation. However, in real systems, accelerator transactions always begin on a host CPU executing an application software thread. The approach taken to synchronize the data and control flow handover between CPU and accelerator impacts both the data movement efficiency and the ease of programming. As we continue to leverage heterogeneous hardware specialization to increase energy efficiency, data movement can quickly become a bottleneck to achieving practical gains in throughput and/or energy efficiency.

[1]Essentially a programmable accelerator implemented on the eFPGA [18].

A. Non-Coherent Accelerator Attach

Very simple baremetal systems with only physical addressing place the burden of manual data movement on the programmer. Since the CPU is often moving and manipulating relevant data prior to initiating an accelerator task, the data required by the accelerator may well be resident in dirty cache lines. The CPU cache is not visible to the rest of the system, and the programmer must be careful to flush the cache to main memory before a non-coherent accelerator can access it. A missing cache flush software bug can be very difficult to diagnose, resulting in subtly unpredictable behavior. Hence, software development for such SoCs containing numerous heterogeneous hardware accelerators is extremely challenging.

The cache flush/invalidate operations incur latency and are wasteful in terms of the data movement round trip from the CPU to main memory and eventually back on chip again. This obviously increases the cost of moving data to and from an accelerator and also makes it more difficult to implement composable accelerators, which require frequent fine-grained data movement in cooperation with the CPU.

B. Coherent Accelerator Attach

In a coherent system, additional on-chip bus signaling is implemented to ensure that any transactions will return fresh data, even if it is resident in a dirty cache line elsewhere in the system. However, this obviously incurs additional complexity, logic, and wires. These overheads can represent a significant cost for an accelerator block and may compound as the number of accelerators grows. Subsequently, coherent accelerator attach is not that common in research test chips.

C. Accelerator Coherency Port

In this work, we implement coherent accelerator attach with minimal additional logic, by using an accelerator coherency port (ACP) on the L2 cache in the A53 cluster. ACP essentially allows coherent, physically addressed access to the CPU L2 cache [22], [23], which is much faster compared to a non-coherent programming model, where the cache must be flushed to main memory before an accelerator can touch it. The low data migration cost via ACP enables a flexible, composable approach to accelerating individual kernels, orchestrated

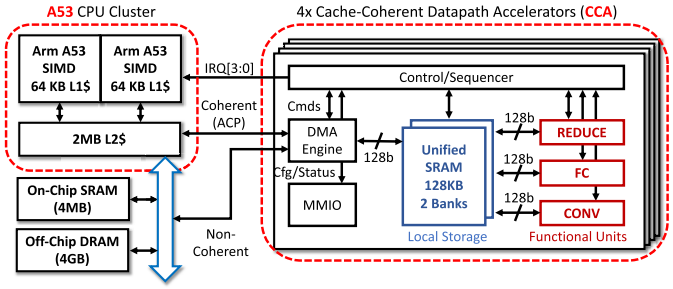


Fig. 3. Cache-coherent datapath accelerators top-level organization and attachment to the CPU cluster and memory system.

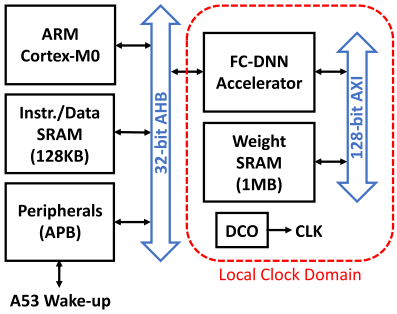


Fig. 4. Autonomous low-power AON subsystem.

by software running on the CPU, while sharing data in the L2 cache. This also increases utilization of the large L2 cache SRAM, which is otherwise typically idle while the accelerator is executing and blocking the software execution path.

We implement CCAs connected to the ACP port using an Arm NIC-400 segment with 64-bit data width, so they can coherently collaborate jointly on data stored in the L2 cache, as shown in Fig. 3. The CCAs are also attached (non-coherently) to the main SoC interconnect, which is important for data with low reuse which is therefore more efficiently handled by a direct path, rather than through the cache subsystem. The two paths (coherent and non-coherent) allow us to compare the approaches. The CCA design itself is discussed in Section IV-E.

IV. SoC Architecture

In this section, we describe the 16-nm SoC architecture (Fig. 1), consisting of: 1) AON subsystem; 2) dual-core Arm Cortex-A53 CPUs; 3) 2 × 2 eFPGA array; and 4) quad-core cache-coherent datapath accelerators (CCA). The memory system includes an interconnect, a 4-MB four-bank software-managed SRAM, and an off-chip interface to an FPGA board for DRAM and other peripherals.

A. Accelerator Offload Overheads

The typical accelerator execution model begins with a CPU thread, which writes setup/control registers and then transfers input data to the accelerator. When the accelerator execution is complete, an interrupt is raised and the CPU can retrieve the output data from the accelerator. The hardware aspects of these overheads are discussed in this article, however, we also note that a full software stack with operating system and drivers can also increase offload overheads.

B. AON Subsystem

The AON subsystem (Fig. 4) has the lowest power profile on the SoC and can operate autonomously from its own dedicated SRAM, while the remainder of the SoC is powered down. AON is used to perform SoC housekeeping tasks and autonomous continuous sensing, such as repeated inference on small DNNs [24], [25]. For more complex tasks requiring more compute or a rich feature set, AON boots the A53 CPUs.

The subsystem is based around an Arm Cortex-M0 microcontroller, an AHB interconnect with 32-bit data width, a 128-KB SRAM for storing program binaries and data, simple peripherals such as GPIOs for sensor interfaces and timers to orchestrate wake-up events. It also includes a fully connected DNN (FC-DNN) inference accelerator optimized for sparsity and includes a self-contained 1 MB SRAM for storing weights without requiring an off-chip memory interface [24]. An integrated digitally controlled oscillator (DCO) allows the performance of the accelerator to be scaled to meet a range of throughput requirements. The AON cluster implementation is optimized for power consumption rather than energy, as it is typically running most of the time.

C. Dual-Core Arm Cortex-A53 Cluster (A53)

A general-purpose CPU cluster is used to run the system and application software. The Arm Cortex-A53 is widely used in commercial mobile and IoT products and is significantly more capable than the microcontroller in the AON subsystem. It implements a rich 64-bit ISA with an in-order, eight-stage, dual-issue pipeline, with wide 128-bit SIMD units, and a floating point unit. The cluster has private 64 KB L1 caches, a shared 2 MB L2 cache, and also includes multi-core debug and a generic interrupt controller. The ACP on the cluster provides direct (physically addressed) access into the large L2 cache, which we use for attaching accelerators with low offloading overhead.

D. eFPGA Cluster

The eFPGA (Section II) is integrated as a first-class citizen on the SoC, via dedicated bus interfaces (Fig. 5). To save resources on the eFPGA, these interfaces are mainly implemented in logic on the SoC, with some simple minimal interfacing and handshaking on the eFPGA side synthesized into LUTs. The first of these interfaces is an AXI slave used for programming the eFPGA bitstream. Programming is driven by the A53, which reads the bitstream from the main memory and writes it to the programming slave word-by-word, where a data integrity checksum is performed, before it is driven onto scan chains inside the eFPGA macro to configure the array. Once programming is complete, A53 releases the resets and enables the user clocks to the eFPGA, at which point the programmed design is operational. Another slave interface attached to a 128-byte register file provides storage for user control and status registers (CSRs). These registers are directly connected to the eFPGA macro pins and save precious LUTs inside the macro. Finally, a master and slave AXI interface pair provides

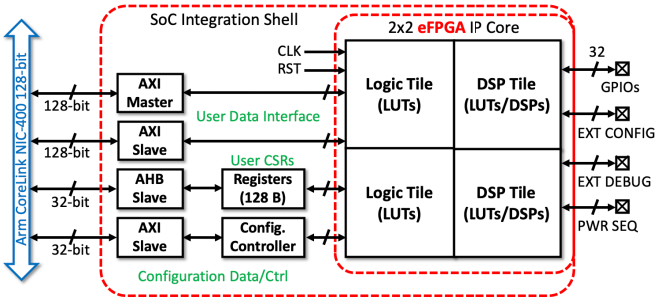


Fig. 5. SoC integration of the eFPGA cluster, with four interfaces: 1) slave for programming the bitstream; 2) slave for CSRs; 3) slave for user data; and 4) master for user data.

128-bit data movement into and out of the SoC interconnect, with support for bursts and other standard features. GPIO pins are also provided for PCB interfacing flexibility via the eFPGA, as well as some pins for external configuration, debug, and power sequencing.

E. Quad-Core Cache Coherent Datapath Accelerators (CCA)

A very wide variety of hardware accelerators for DNN inference have been published to date [8]–[12]. They typically consist of large datapaths and large SRAMs, with relatively simple control and data movement. The improvements in throughput and energy efficiency compared to CPUs can be multiple orders of magnitude. However, specialized hardware has limited flexibility and can be prone to obsolescence.

In this work, we implement a CCA with composable kernel primitives, a paradigm practical only due to the low-latency coherent ACP interface (Section III). The CCA core (Fig. 3) implements a datapath with three key atomic kernels of: 1) 2-D convolution; 2) dot product; and 3) vector reduction. These are arguably the most common fundamental operations in DNN inference and can be composed together to implement specific DNN layers, with the CPU implementing auxiliary processing such as activation functions. The datapath for 2-D convolution is shown in Fig. 6. The activations are streamed in via an internal DMA engine, which provides native support for 1×1 convolution by loading activations channel-wise (red in Fig. 6) or row-wise (denoted in blue) for 2-D convolutions. Parallel MACs produce partial products that are stored in two 8×32-bit register files. The MACs are organized into 2×4 parallel lanes to allow the accelerator more parallelism in 2-D convolution with smaller convolution kernels becoming more prevalent. Finally, a merge stage optionally adds partial sums before writeback to the local SRAM. Input operands are 16-bit, with 32-bit partial sums.

F. On-Chip SoC Interconnect and Memory System

The main on-chip SoC interconnect is an Arm NIC-400 with a 128-bit data width, designed and configured using the Arm Socrates tool [26]. To allow aggressive dynamic voltage and frequency scaling (DVFS), each compute cluster operates on an independent clock domain, with fully asynchronous clock domain crossing (CDC). The interconnect is configured to balance the data transfer bandwidth with the throughput of the compute clusters. A 4-MB, four-bank on-chip SRAM memory

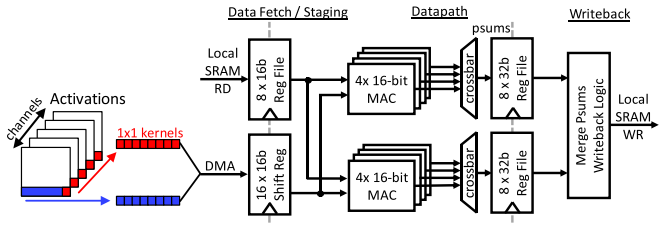


Fig. 6. Design of the three-stage flexible accelerator datapath optimized for 2-D convolution (blue) with native support for 1×1 convolution (red).

is attached to this interconnect and provides high-performance on-chip software-managed scratchpad memory. For simple tests, this SRAM is sufficient. However, for larger tests on more representative workloads, as well as programming the eFPGA, a much larger main memory is required. The test chip does not include a DRAM main memory interface due to the high cost, time, and risk this introduces. Instead, we interface the test chip with a Xilinx KCU105 FPGA, which acts as a dummy slave memory system. This allows the SoC to initiate read and write transactions on the Xilinx FPGA, which includes DRAM and other non-trivial peripherals such as a PCIe interface.

G. Synthesizable Off-Chip Interface

We implemented a simple parallel off-chip interface between the test chip and a Xilinx FPGA. Fig. 7 gives an overview of the interface. The forward link (test chip to FPGA) occupies 38 bits and consists of 32-bit data, control signaling, source-synchronous clock, and a reset. The reverse link (FPGA to test chip) is 71 bits, mainly due to a larger 64-bit data payload. A simple token flow control is used to ensure that buffers do not overflow on the FPGA side, which may be very slow and blocking in the case of some low-performance peripherals. The address, data, and control signaling from the on-chip interconnect is then multiplexed down to the link width.

The physical layer uses simple rail-to-rail source synchronous signaling, with forwarded clocks and resets from the SoC side. The PHY on the test chip side does not use any custom cells or layout and simply consists of a set of launching flip-flops and standard library IO pads. On the Xilinx FPGA side, the per-pin de-skewing functionality is used to align the incoming and outgoing data to correct for delay matching between data bits and the forwarded clock. This can be done automatically for the forward link. For the reverse link from the Xilinx FPGA, we added a debug mode [denoted by debug signal paths (DBG) in Fig. 7] using a loop-back on the FPGA side, with logic to capture consecutive data words on the test chip to allow implement automatic de-skew using the A53 CPUs.

The synthesizable PHY achieves reliable error-free communication up to around 200 MHz clock frequency in our implementation. This gives a forward data rate of around 6.4 Gbps and reverse link data rate of 12.8 Gbps. However, we typically operate the link at 157 MHz, which is an integer ratio of the memory controller clock on the FPGA.

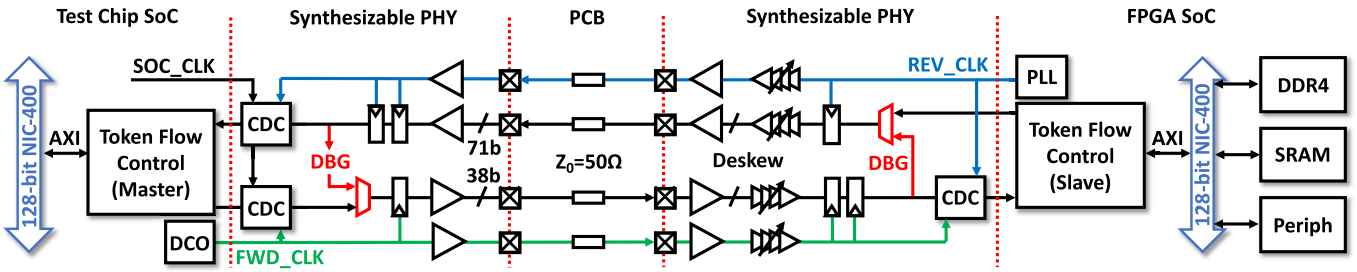


Fig. 7. Synthesizable off-chip interface to off-chip Xilinx FPGA which provides DRAM main memory and other peripherals. Abbreviations: CDC—clock domain crossing; PLL—phase-locked loop; DCO—digitally controlled oscillator; DBG—debug signal paths.

TABLE I
TEST CHIP SUMMARY

Technology	TSMC 16nm FinFET 1P9M
Nominal Supply Voltage	0.8V
Die Size	25mm^2
Transistor Count	>0.5 billion
Total SRAM	72.2 Mbits
Package	Flip-chip 672-pin BGA
Clock / Power Domains	7 / 5
Supply Voltage Range	0.5–1.05V (AON, A53) 0.4–1.05V (CCA, eFPGA[1])
Fmax Range (V_{MIN}–V_{MAX})	142–972 MHz (AON) 131–1004 MHz (A53) 25–1006 MHz (CCA) 51–747 MHz (eFPGA[1])
Power Dissipation (V_{MEP})	1.13 mW (AON) 6.89 mW (A53) 3.88 mW (CCA) 31.80 mW (eFPGA[1])

[1]eFPGA operating points are heavily design dependent.

V. TEST CHIP IMPLEMENTATION

The 25-mm^2 test chip was implemented in a TSMC 16-nm FinFET technology (Table I). Each compute cluster on the test chip operates on an independent voltage and clock domain, as does the SoC fabric that includes the NIC-400 and 4-MB on-chip SRAM. This provides the SoC with flexibility in efficiency and throughput by enabling each compute cluster to tailor its operating point to the workload needs, in exchange for added design complexity. Annotated photographs of the die and PCB are given in Fig. 8.

A. RTL Design and Validation

The IP integrated in SMIV falls into four categories.

1) *In-House Soft IP (Verilog RTL):* CCA and AON blocks.
2) *In-House Hand-Mapped Cell Netlists:* DCOs and PHY.
3) *Commercial Soft IP (Verilog RTL):* Arm Cortex-A53, Cortex-M0, and CoreConnect interconnects.
4) *Commercial Hard IP Macros (GDS):* eFPGA.

A large portion of the SoC integration was performed using Arm Socrates [26], which configures the NIC-400 interconnect, and generates a top-level netlist for this portion. This was then integrated by hand with the remainder of the design, including the AON subsystem, the wide-IO interface, clocking and reset circuits, and the top-level pad-ring.

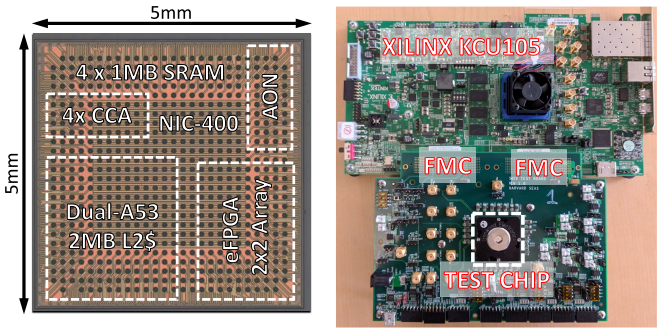


Fig. 8. Photographs of (a) 25-mm^2 SMIV test chip in 16-nm FinFET, and (b) PCBs used for bring up and characterization. The test chip PCB shown at the bottom of (b) interfaces to a Xilinx KCU105 development board (top) via FMC connectors [27], which provides DRAM main memory.

The RTL implementation and validation of a non-trivial SoC can quickly become an overwhelmingly time-consuming task, so was developed simple automation tooling to aid in developing and maintaining vital and common components such as the pad ring, test circuits, memory maps, and memory-mapped registers. The open-source *CHIPKIT* project [28] provides a number of tools and IPs, including the *VGEN* tool which was used extensively during the design process.

We also developed a comprehensive full-chip validation methodology, which includes RTL and netlist validation steps, as well as FPGA emulation. Particular validation effort was spent on CDCs, clock and reset behavior, and chip boot up sequencing. The eFPGA IP, in particular, requires a controlled power up sequence to prevent crowbar current, which involved additional validation effort. All the main functional blocks (Fig. 1) include asynchronous CDCs on the bus interfaces to allow DVFS flexibility.

B. Physical Design

Standard cell libraries and SRAM compilers from Arm and TSMC were used, with a multi-vendor EDA flow based on Synopsys Design Compiler for synthesis, Cadence Innovus for place and route, and Synopsys PrimeTime for static timing analysis (STA). The 25-mm^2 test chip dictated that we use a hierarchical physical implementation approach in order to allow for faster respins of the design from RTL to timing of extracted layout. The hierarchical approach also facilitates implementation of the independent power domains, simplifying the power specification required for SoC integration, while

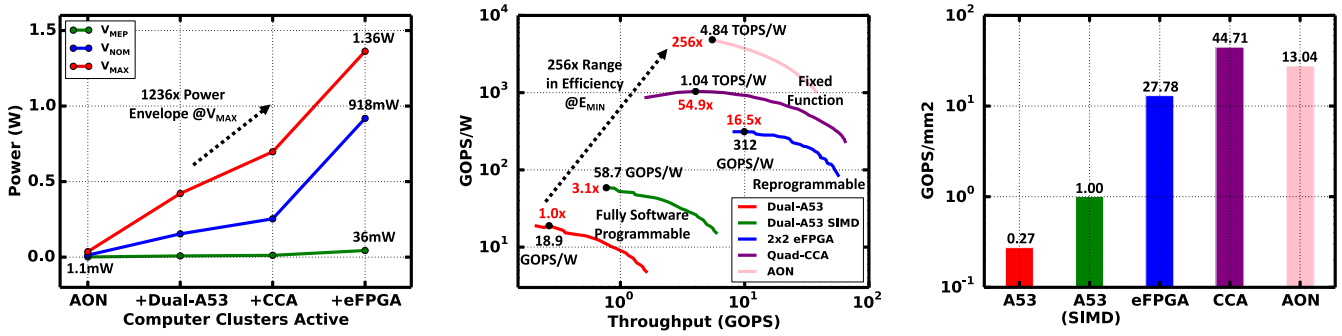


Fig. 9. Measured SoC characterization results. (a) Power dissipation scales as compute clusters are activated, from sleep mode with only AON active up to fully active SoC. (b) Energy and throughput across compute clusters for a matrix multiply workload. The curves are generated by optimal voltage/frequency scaling. The total energy scales by 256× as we move from CPU programmed in software, through to eFPGA programmed with RTL and up to increasingly fixed-function hardware accelerators (CCA and AON). The black markers represent the minimum energy point (V_{MEP}), and the extremes of each line represent V_{MAX} and V_{MIN}. All are processing dense matrix–matrix multiplication, except for AON which is performing multiplication of dense matrix and a sparse vector. (c) Silicon area efficiency scales from the relatively low compute density of the CPUs, to eFPGA, and up to the CCA and AON accelerators. (a) Power. (b) Energy. (c) Area.

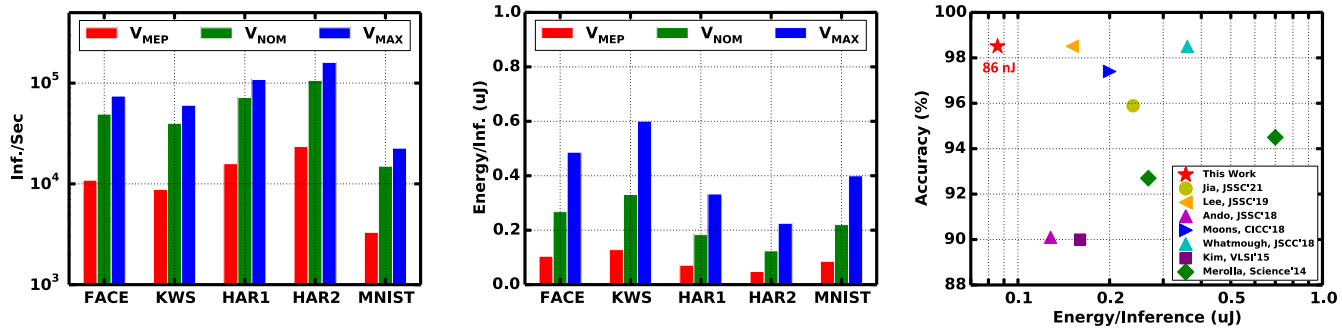


Fig. 10. Measured FC accelerator in AON cluster executing neural network benchmarks, showing (a) throughput and (b) energy across the operating point range. Measured energy at V_{MEP} and accuracy on the MNIST dataset (c) compared with previous measured silicon results, including: Whatmough *et al.* [7], Lee *et al.* [24], Jia *et al.* [29], Ando *et al.* [30], Moons *et al.* [31], Kim *et al.* [32], and Merolla *et al.* [33]. V_{MEP}: 0.5 V/142 MHz, V_{NOM}: 0.8 V/644 MHz, V_{MAX}: 1.05 V/972 MHz.

keeping voltage domain requirements for each cluster manageable. Implementation constraints encompassed a range of power/performance points covering foundry process, voltage and temperature (PVT) corners. The die was flip-chip bonded to a custom 671-pin BGA package substrate, with dedicated power delivery networks for each voltage island on the SoC.

C. Test and Measurement Setup

Fig. 8(b) shows the test setup. The test chip PCB uses a high-performance membrane BGA socket to avoid significantly compromising signal and power integrity. The PCB provides regulated supply voltages for each power domain, all from a single 5-V dc connector. It also provides a power-on reset signal and a 50-MHz system clock to the SoC, with all other clocks and resets generated on-chip. Fast on-chip clocks are generated using a simple open-loop DCO, which is composed of only standard cells with no custom layout, but does require calibration. A UART to USB transceiver chip on the PCB is used to allow a laptop to be directly connected to the PCB for communication with the test chip. This facility is used to load binary programs and even remotely host the SoC, which is very convenient for testing. The test chip PCB is connected to the Xilinx KCU105 using the FPGA mezzanine card (FMC) connectors [27].

TABLE II
AON FC BENCHMARK TASKS

Name	Task	Dataset	Topology	Accuracy
FACE	Face Pair Verification	Labelled Faces in the Wild [34]	512-128-128-128-2	78.4%
KWS	Audio Keyword Detection	Command [35]	403-200-200-12	99.1%
HAR1	Human Activity Recognition	Opportunity [36]	924-56-56-56-56-18	89.7%
HAR2	Human Activity Recognition	Smartphone Raw [37]	384-72-72-72-72-13	79.5%
MNIST	Handwritten Digit Classification	MNIST [38]	784-256-256-256-10	98.5%

VI. Measurement Results

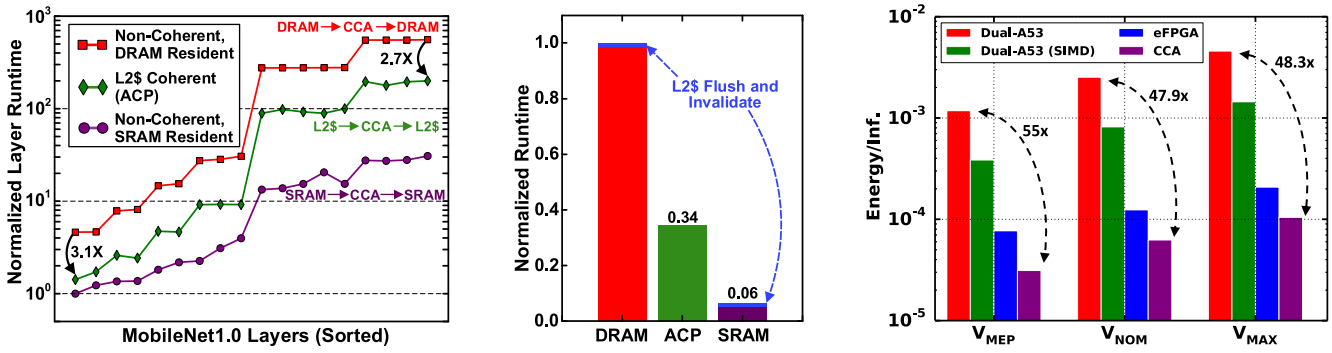
Measurements are for typical silicon at room temperature.

A. SoC Power Envelope

The measured 1236× power envelope [Fig. 9(a)] spans 1.1 mW (AON only, V_{MEP}), through 36 mW (all clusters active, V_{MEP}), and up to 1.36 W (all clusters active, V_{MAX}).

B. AON Cluster

The FC accelerator in the AON cluster is intended to run continuously to detect wake up events, and therefore,



(a) (b) (c)

Fig. 11. Measured results for the MobileNet workload. (a) Runtime benefits of coherent ACP interface on individual MobileNet layers, due to keeping the data local in the L2 cache. (b) Cumulative runtime benefit for a single inference on the whole model. (c) Measured energy per inference on MobileNet at three operating points across the different compute clusters. Note that the AON cluster has limited functionality and does not support a model of this size.

efficiency is critical. We implemented five relevant ML tasks (Table II): face matching [34], keyword detection [35], two different human activity recognition tasks [36], [37], and handwritten digit classification [38]. The FC accelerator was then measured in terms of throughput [Fig. 10(a)] and energy [Fig. 10(b)], over three operating points spanning the full dynamic range: from the minimum energy point (V_{MEP}) at 0.5 V/142 MHz, through nominal (V_{NOM}) at 0.8 V/644 MHz, and up to the highest frequency at the highest voltage, V_{MAX} at 1.05 V/972 MHz. All five tasks are well below 1 uJ per inference at a throughput of many thousands of inferences per second. This ensures low power consumption in AON operation, with plenty of headroom for high sample rates.

A comparison of the AON FC accelerator measurements with previously published full accelerator silicon measurements is given in Fig. 10(c) for the MNIST benchmark. A wide range of approaches have been presented, including spiking neural networks (SNNs) [32], [33], [39] which tend to offer efficiency but poor accuracy, and convolutional neural networks (CNNs) [31], [40] which tend to have a high accuracy, but have a very high operation count which increases energy. FC networks offer a good balance between model size and compute for these AON tasks. Analog [39] and especially analog in-memory compute [30], [41]–[43] approaches have previously shown limited accuracy due to non-idealities, but are improving rapidly. For example, Yin *et al.* [44] report 98.8% accuracy on the MNIST task, although their test chip only implements the matrix–vector computation and does not include activation storage and processing. Binary networks [43] are also a promising direction in digital too.

C. Compute Kernel Acceleration on eFPGA

We implemented eFPGA designs for common workloads that span basic linear algebra subroutines (BLAS), digital signal processing (DSP), and cryptography (crypto) (Table III). The eFPGA designs include a custom 8-bit 10×8 systolic array, 40-/80-tap non-symmetric FIR filters [45], 16-bit fixed-point FFTs supporting 64- and 32-point transforms [20], and 128-bit AES encryption and decryption cores [46]. These are compared against optimized software implementations on the A53 cluster, based on the Ne10 library [47] for DSP, and Eigen [48] for BLAS, both of which are optimized for Arm SIMD. The crypto routines were based on Tiny-AES-C [49].

Table III shows the results of offloading the workloads to the eFPGA, presenting energy efficiency and throughput gains over the A53 cluster at nominal voltage (V_{NOM}) of 0.8 V, as well as the maximum frequency achievable by the eFPGA at V_{MAX} of 1.05 V. The energy efficiency and throughput increase by 5.5× and 27×, respectively, for CONV-2D offloaded to eFPGA. For an 80-tap direct-form FIR filter (FIR-80) design with 100% DSP utilization, the energy efficiency increases by 17.36× compared to the Ne10 software implementation. A 40-tap variant (FIR-40) leaves half of the DSPs idle, and hence, the energy efficiency gain drops to 13.4×. The largest FFT that fits (FFT-64) uses 75% of the DSPs and achieves 47.6× throughput and 7.07× energy gains. The AES128 ECB encryption/decryption kernel (AES-ENC/DEC) emphasizes bit-wise operations, which can be efficiently implemented in LUTs. The eFPGA implementation provides up to 28.9× and 120× improvement for energy efficiency and throughput, mainly due to the efficient use of LUTs by bit-wise operations. Note that all of these designs allow pipelined invocations: they accept new inputs consecutively, except for AES which blocks new inputs until the current computation is complete.

In summary, we were able to achieve very large improvements in both throughput and energy from offloading compute kernels to eFPGA. Although the programming effort is higher than for CPU, eFPGA does still retain programmability.

D. CCA Offload on CCA

To demonstrate the CCA cluster in a typical application scenario, we extensively characterized MobileNetV1 [13] inference performance. This model represents a significant reduction in operations and memory footprint, compared to older image classification models, such as AlexNet [50] and VGG [51]. The model is mainly composed of alternate depth-wise and point-wise convolution layers, using strided convolution rather than pooling, and a single FC layer.

Fig. 11(a) shows the measured runtime on the CCA cluster for each layer in the model. Each line in the plot represents a different offload model. The fastest is to use non-coherent on-chip software-managed SRAM. However, this represents

TABLE III
MEASURED KERNELS ACCELERATED ON EFPGA

Kernel	Task	Data Type	Lines of Code		eFPGA Implementation				Offload Gain vs Dual-A53	
			C/C++[1]	Verilog[2]	Offloading	LUTs (%)	DSPs (%)	F_{MAX}[3]	Throughput[4]	Energy[4]
CONV-2D	BLAS	8-bit	493	485	Pipelined	99.6	100	353.8	27×	5.5×
FIR-40	DSP	8-bit	234	322	Pipelined	13.4	50	595.8	41.9×	13.4×
FIR-80	DSP	8-bit	234	322	Pipelined	27.4	100	521.6	79.9×	17.36×
FFT-32	DSP	16-bit	404	5,019	Pipelined	8.8	35	747.1	34×	10.25×
FFT-64	DSP	16-bit	404	10,061	Pipelined	18.7	75	683.2	47.6×	7.07×
AES-ENC	Crypto	128-bit	564	2,703	Blocking	37.2	0	734.0	64×	19.23×
AES-DEC	Crypto	128-bit	564	2,703	Blocking	37.2	0	732.5	120×	28.9×

Lines of Code: [1]Excludes library code, [2]Synthesizable RTL only. Operating point for both eFPGA and A53: [3]V_{MAX}: 1.05V, and [4]V_{NOM}: 0.8V.

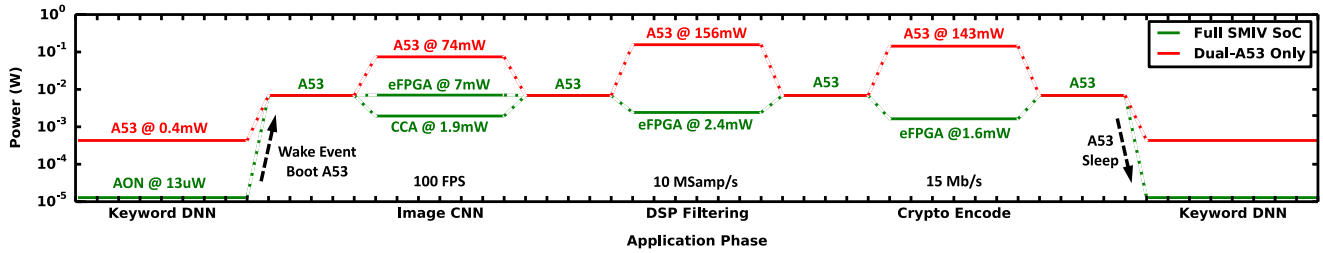


Fig. 12. Measured power consumption for four key application tasks executing at a fixed-throughput on specialized accelerators compared to a baseline of A53 CPU only. Specialized hardware achieves more than an order of magnitude lower power consumption for all tasks. Each kernel is measured on the test chip in isolation, with dotted lines between phases, indicating that the transition is not measured. For operating points, see Section VI-F. The CCA accelerator is specialized for DNN workloads and cannot execute the DSP and Crypto workloads.

the most challenging programming model, as the software is entirely responsible for allocating and moving buffers. For larger models, the weights will not fit in on-chip SRAM, and so they must be loaded from off-chip DRAM. With non-coherent memory access, any data that is resident in the CPU cache must be manually flushed to main memory.

The benefits of the ACP interface were previously explored on FPGA in [22] and in ASIC simulation in [23]. Measured results on silicon are given here. The CCA performance with the ACP interface is shown by the green line in Fig. 11(a). Here, we achieve a runtime that falls between non-coherent SRAM and non-coherent DRAM. However, although not quite as fast as non-coherent SRAM, it is much faster than DRAM and at the same time is much easier to program and does not require explicit L2 cache flushes. Fig. 11(b) summarizes the runtime for the whole model, for the three data movement schemes.

Fig. 11(c) gives energy per inference on the whole MobileNetV1-128 model, for A53, eFPGA, and CCA clusters (AON does not support CNNs). The energy per inference is compared at the three operating points of minimum energy point (V_{MEP}), nominal (V_{NOM}), and max frequency (V_{MAX}). Relative to the CPU baseline, we see an improvement of 3.1× (SIMD), 22.7× (eFPGA), and 47.9× (CCA) at V_{NOM}.

E. Compute Efficiency Across Clusters

GEMM is a key kernel in many compute-intensive workloads, including DNN inference. In this section, we benchmark GEMM performance across all four of the compute clusters on the SMIV test chip. Fig. 9(b) shows a comparison of GEMM throughput and energy efficiency for all the compute clusters. Each curve represents the range of operating points given by the highest clock frequency at each functional supply voltage. The black marker on each curve indicates the minimum energy point (MEP) of the range for each design. The baseline SIMD CPUs achieve 58.7 GOPS/W energy efficiency. Moving to eFPGA, which still retains post-silicon programmability, increases energy efficiency to 312.4 GOPS/W at a throughput of around 100 GOPS, limited by the number of DSP blocks available in the eFPGA. The CCA cluster achieves energy efficiency of 1.04 TOPS/W from dedicated custom hardware, with limited post-silicon flexibility and a more rigid, non-portable programming model. Finally, the AON cluster includes a very specialized accelerator that only supports FC DNN layers, but is heavily optimized for energy efficiency [7] and achieves 4.88 TOPS/W.

The throughput normalized area efficiency is compared in Fig. 9(c). Typically, CPUs have a low peak arithmetic throughput relative to their area, and the A53 is no exception. The eFPGA has fairly high throughput, even at moderate clock frequencies and achieves more than an order of magnitude higher area efficiency as a result. However, the highest area efficiency is achieved by the most specialized hardware accelerators (CCA and AON). AON has lower area efficiency, as it operates without the rest of the SoC and hence, requires a dedicated 1 MB SRAM, increasing the area significantly.

F. Real-Time Application Acceleration

To demonstrate the power savings from a heterogeneous SoC at fixed throughput, we use an application workload

TABLE IV
COMPARISON WITH RELATED WORK

	Celerity [52]	Hwacha [53]	Zimmer et al. [54]	SMIV [14] (This Work)			
Technology	16nm FinFET	16nm FinFET	16nm FinFet	16nm FinFET			
SoC Area	25mm^2	24mm^2	6mm^2 (1 Die)	25mm^2			
Total SRAM	5.56 MB	4.5 MB	625 KB	9 MB			
Host CPU	5-Core RISC-V	RISC-V	RISC-V	Dual-Core Arm Cortex-A53			
Accelerator	**496-CPU Array**	**Vector Acc.**	**Spatial NPU**	**Dual-A53 (Host)**	**2x2 eFPGA**	**Quad-CCA**	**AON**
Area	15.25mm^2	24mm^2	3.1mm^2	5.88mm^2	4.34mm^2	1.44mm^2	1.35mm^2
F_{MAX}	1.4 GHz	1.6 GHz	2 GHz	1 GHz	354 MHz	1 GHz	972 MHz
SRAM	752 KB	3 MB L3$	625 KB	2 MB L2$	—	4x 128 KB	1 MB
Precision	INT	16/32/64b FP	8b INT	INT, FP	Arbitrary	16b INT	8b INT
Throughput[1]	695 GRVI/s	368.4 GFLOPS	4.01 TOPS	5.90 GOPS	56.506 GOPS	64.38 GOPS	37.5 GOPS
Energy[2]	93 GRVI/s/W	209.5 GFLOPS/W	0.96 TOPS/W	58.7 GOPS/W	312.4 GOPS/W	1.04 TOPS/W	4.84 TOPS/W
Area Efficiency[1]	45.6 GRVI/s/mm^2	15.4 GFLOPS/mm^2	1.29 TOPS/mm^2	1 GOPS/mm^2	13 GOPS/mm^2	45 GOPS/mm^2	28 GOPS/mm^2

As reported at design operating points: [1]V_{MAX}, [2]V_{MEP}.

with four key compute phases. Between each major compute task, application code runs on the host CPU. The first task is a DNN audio keyword detection workload, which runs continuously at the real-time audio rate, until a keyword is detected, at which point the A53 is booted up. The second task is an image classification CNN, which runs MobileNet at 100 image patches per second. The third is a DSP filtering task, which runs a 40-tap FIR on a 1-D signal block at 10 MSamples/s. The fourth is a cryptograph task, which applies AES encoding of a data block at 15 Mb/s. DVFS is tuned optimally to meet the throughput requirement on each compute cluster.

Fig. 12 compares the power consumption of these application phases for: 1) an A53 CPU-only system and 2) the full SMIV SoC, mapping each phase to one of the three accelerators. Across the whole workload, the accelerators demonstrate at least an order of magnitude lower power consumption than the A53 alone, at the same fixed throughput. The keyword detection DNN dissipates 13 μW on the AON block at 0.5 V/1.6 MHz with the rest of the SoC powered down. The MobileNet CNN dissipates 1.9 mW on the CCA at 0.4 V/25 MHz. The DSP filtering task on eFPGA consumes 2.4 mW at 0.4 V/10 MHz. Finally, the Crypto AES encode on eFPGA consumes 1.6 mW at 0.42 V/7.5 MHz.

In summary, we find that on this fixed-throughput real-time workload, the gains achieved compared directly with the A53 are different to the energy comparison at variable throughput (Section VI-E), because A53 often struggles to meet even modest real-time throughput requirements and therefore requires higher supply voltage to increase frequency. On the other hand, specialized hardware has much higher performance headroom and can operate at V_{MIN} to reduce power while sustaining sufficient frequency.

G. Comparison With Previous Work

This article demonstrates a heterogeneous SoC with four distinct compute clusters on a single SoC, highlighting the relative trade-offs in throughput, efficiency, and programming model. We are not aware of any similar heterogeneous SoCs against which to compare our work, however, here we briefly compare the four compute clusters individually with corresponding state-of-the-art single-accelerator articles. Hence, Table IV gives a high-level comparison with some recent publications in the same 16-nm technology.

In terms of CPU-based accelerators, the Celerity 496-core CPU array [52] and Hwacha [53] vector unit both achieve higher throughput than we target for IoT applications. However, although both these chips demonstrate high throughput, their energy efficiency is somewhat limited and lower than achieved in our work by the eFPGA, CCA, and AON clusters.

Comparing the CCA (Section III and AON with a recent 16-nm DNN accelerator [54], the latter demonstrates higher throughput (four TOPS) than we target here for IoT applications. However, the energy efficiency at nearly 1 TOPS/W is similar to that of the CCA cluster in our work, and the AON cluster significantly exceeds this at 4.84 TOPS/W, but is heavily specialized for small FC models.

VII. CONCLUSION

This article describes a 16-nm SoC for IoT applications, comprising four main compute clusters: an AON subsystem with self-contained DNN accelerator, a dual-core Arm Cortex-A53 cluster, an eFPGA cluster, and a cache-coherent datapath accelerator. The range of compute specialization allows for very low power consumption in AON mode (1.1 mW), while providing both higher performance and broader workload support from activating the other compute clusters. The eFPGA cluster demonstrates increases in energy efficiency of 5.5–28.9× after offloading from the A53 CPUs. The cache-coherent datapath accelerators operating in concert from the A53 L2 cache via ACP increase energy efficiency on the MobileNetV1 workload by 55× compared to A53. Finally, the overall efficiency range for GEMM compared to the dual-core A53 baseline spans 3.1× (A53 with SIMD), 16.5× (eFPGA), 54.9× (CCA), and up to 256× (AON).

ACKNOWLEDGMENT

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. The authors are grateful to Arm and FlexLogix for providing IP.

REFERENCES

[1] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, "UNPU: An energy-efficient deep neural network accelerator with fully variable weight bit precision," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 173–185, Jan. 2019.

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

[3] H. Li, M. Bhargava, P. N. Whatmough, and H.-S.-P. Wong, "On-chip memory technology design space explorations for mobile deep neural network accelerators," in *Proc. 56th Annu. Design Autom. Conf.*, Jun. 2019, pp. 1–6.

[4] J. Lee, J. Lee, D. Han, J. Lee, G. Park, and H.-J. Yoo, "An energy-efficient sparse deep-neural-network learning accelerator with fine-grained mixed precision of FP8–FP16," *IEEE Solid-State Circuits Lett.*, vol. 2, no. 11, pp. 232–235, Nov. 2019.

[5] D. Bankman, L. Yang, B. Moons, M. Verhelst, and B. Murmann, "An always-on 3.8 μJ/86% CIFAR-10 mixed-signal binary CNN processor with all memory on chip in 28-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 158–172, Jan. 2019.

[6] M. E. Sinangil *et al.*, "A 7-nm Compute-in-Memory SRAM macro supporting multi-bit input, weight and output and achieving 351 TOPS/W and 372.4 GOPS," *IEEE J. Solid-State Circuits*, vol. 56, no. 1, pp. 188–198, Jan. 2021.

[7] P. N. Whatmough, S. K. Lee, D. Brooks, and G.-Y. Wei, "DNN engine: A 28-nm timing-error tolerant sparse deep neural network processor for IoT applications," *IEEE J. Solid-State Circuits*, vol. 53, no. 9, pp. 2722–2731, Sep. 2018.

[8] J. Song *et al.*, "7.1 An 11.5 TOPS/W 1024-MAC butterfly structure dual-core sparsity-aware neural processing unit in 8 nm flagship mobile SoC," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2019, pp. 130–132.

[9] Y. Jiao *et al.*, "7.2 A 12 nm programmable convolution-efficient neural-processing-unit chip achieving 825 TOPS," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 136–140.

[10] C.-H. Lin *et al.*, "7.1 A 3.4-to-13.3 TOPS/W 3.6 TOPS dual-core deep-learning accelerator for versatile AI applications in 7 nm 5G smartphone SoC," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 134–136.

[11] J. Oh *et al.*, "A 3.0 TFLOPS 0.62V scalable processor core for high compute utilization AI training and inference," in *Proc. IEEE Symp. VLSI Circuits*, Jun. 2020, pp. 1–2.

[12] I. Bratt, "Arm's first-generation machine learning processor," in *Proc. IEEE/ACM SIGARCH Hot Chips A, Symp. High Perform. Chips*, May 2018.

[13] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*. [Online]. Available: http://arxiv.org/abs/1704.04861

[14] P. N. Whatmough *et al.*, "A 16 nm 25 mm^2 SoC with a 54.5x flexibility-efficiency range from dual-core arm cortex-A53 to eFPGA and cache-coherent accelerators," in *Proc. Symp. VLSI Circuits*, Jun. 2019, pp. C34–C35.

[15] P. N. Whatmough *et al.*, "SMIV: A 16 nm SoC with efficient and flexible DNN acceleration for intelligent IoT devices," in *Proc. IEEE/ACM SIGARCH Hot Chips A, Symp. High Perform. Chips*, 2018.

[16] D. Greenhill *et al.*, "3.3 A 14 nm 1 GHz FPGA with 2.5D transceiver integration," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2017, pp. 54–55.

[17] F.-L. Yuan, C. C. Wang, T.-H. Yu, and D. Marković, "A multi-granularity FPGA with hierarchical interconnects for efficient and flexible mobile computing," *IEEE J. Solid-State Circuits*, vol. 50, no. 1, pp. 137–149, Jan. 2015.

[18] J. Fowers *et al.*, "A configurable cloud-scale DNN processor for real-time AI," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2018, pp. 1–14.

[19] P. N. Whatmough, C. Zhou, P. Hansen, S. K. Venkataramanaiah, J. Seo, and M. Mattina, "FixyNN: Efficient hardware for mobile computer vision via transfer learning," in *Proc. 2nd SysML Conf.*, Palo Alto, CA, USA, 2019.

[20] P. Milder, F. Franchetti, J. C. Hoe, and M. Püschel, "Computer generation of hardware for linear digital signal processing transforms," *ACM Trans. Design Autom. Electron. Syst.*, vol. 17, no. 2, pp. 1–33, Apr. 2012.

[21] B. Khailany *et al.*, "A modular digital VLSI flow for high-productivity SoC design," in *Proc. 55th Annu. Design Autom. Conf.* New York, NY, USA: ACM, Jun. 2018, pp. 1–6, doi: 10.1145/3195970.3199846.

[22] M. Sadri, C. Weis, N. Wehn, and L. Benini, "Energy and performance exploration of accelerator coherency port using Xilinx Zynq," in *Proc. 10th FPGAworld Conf. (FPGAworld)*. New York, NY, USA: ACM, 2013, pp. 1–8, doi: 10.1145/2513683.2513688.

[23] S. L. Xi, Y. Yao, K. Bhardwaj, P. Whatmough, G.-Y. Wei, and D. Brooks, "SMAUG: End-to-end full-stack simulation infrastructure for deep learning workloads," *ACM Trans. Archit. Code Optim.*, vol. 17, no. 4, pp. 1–26, Nov. 2020, doi: 10.1145/3424669.

[24] S. K. Lee, P. N. Whatmough, D. Brooks, and G.-Y. Wei, "A 16-nm always-on DNN processor with adaptive clocking and multi-cycle banked SRAMs," *IEEE J. Solid-State Circuits*, vol. 54, no. 7, pp. 1982–1992, Jul. 2019.

[25] S. Kodali, P. Hansen, N. Mulholland, P. Whatmough, D. Brooks, and G.-Y. Wei, "Applications of deep neural networks for ultra low power IoT," in *Proc. IEEE Int. Conf. Comput. Design (ICCD)*, Nov. 2017, pp. 589–592.

[26] *Arm Socrates*. Accessed: Sep. 1, 2021. [Online]. Available: https://developer.arm.com/tools-and-software/ip-configuration-tools/socrates

[27] R. Seelam. (2019). *I/O Design Flexibility With the FPGA Mezzanine Card (FMC)*. [Online]. Available: https://www.xilinx.com/support/documentation/white_papers/wp315.pdf

[28] P. N. Whatmough, M. Donato, G. G. Ko, S. K. Lee, D. Brooks, and G.-Y. Wei, "CHIPKIT: An agile, reusable open-source framework for rapid test chip development," *IEEE Micro*, vol. 40, no. 4, pp. 32–40, Jul. 2020.

[29] T. Jia, Y. Ju, and J. Gu, "A dynamic timing enhanced DNN accelerator with compute-adaptive elastic clock chain technique," *IEEE J. Solid-State Circuits*, vol. 56, no. 1, pp. 55–65, Jan. 2021.

[30] K. Ando *et al.*, "BRein memory: A single-chip binary/ternary reconfigurable in-memory deep neural network accelerator achieving 1.4 TOPs at 0.6 W," *IEEE J. Solid-State Circuits*, vol. 53, no. 4, pp. 983–994, Apr. 2018.

[31] B. Moons, D. Bankman, L. Yang, B. Murmann, and M. Verhelst, "BinarEye: An always-on energy-accuracy-scalable binary CNN processor with all memory on chip in 28 nm CMOS," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Apr. 2018, pp. 1–4.

[32] J. K. Kim, P. Knag, T. Chen, and Z. Zhang, "A 640M pixel/s 3.65mW sparse event-driven neuromorphic object recognition processor with on-chip learning," in *Proc. IEEE Symp. VLSI Circuits Tech. Papers*, Kyoto, Japan, Jun. 2015, pp. C50–C51.

[33] P. A. Merolla *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, Aug. 2014. [Online]. Available: http://science.sciencemag.org/content/345/6197/668

[34] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," Univ. Massachusetts, Amherst, Amherst, MA, USA, Tech. Rep. 07-49, Oct. 2007.

[35] P. Price, W. M. Fisher, J. Bernstein, and D. S. Pallett, "The DARPA 1000-word resource management database for continuous speech recognition," in *Proc. Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, vol. 1, Apr. 1988, pp. 651–654.

[36] R. Chavarriaga *et al.*, "The opportunity challenge: A benchmark database for on-body sensor-based activity recognition," *Pattern Recognit. Lett.*, vol. 34, no. 15, pp. 2033–2042, Jan. 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167865512004205

[37] J.-L. Reyes-Ortiz, L. Oneto, A. Samà, X. Parra, and D. Anguita, "Transition-aware human activity recognition using smartphones," *Neurocomputing*, vol. 171, pp. 754–767, Jan. 2016, doi: 10.1016/j.neucom.2015.07.085.

[38] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[39] F. N. Buhler, P. Brown, J. Li, T. Chen, Z. Zhang, and M. P. Flynn, "A 3.43 TOPS/W 48.9 pJ/pixel 50.1 nJ/classification 512 analog neuron sparse coding neural network with on-chip learning and classification in 40 nm CMOS," in *Proc. Symp. VLSI Circuits*, Jun. 2017, pp. C30–C31.

[40] B. Moons and M. Verhelst, "A 0.3-2.6 TOPS/W precision-scalable processor for real-time large-scale ConvNets," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Jun. 2016, pp. 1–2.

[41] J. Zhang, Z. Wang, and N. Verma, "In-memory computation of a machine-learning classifier in a standard 6T SRAM array," *IEEE J. Solid-State Circuits*, vol. 52, no. 4, pp. 915–924, Apr. 2017.

[42] A. Biswas and A. P. Chandrakasan, "Conv-RAM: An energy-efficient SRAM with embedded convolution computation for low-power CNN-based machine learning applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 488–490.
[43] T. Stadtmann, C. Latotzke, and T. Gemmeke, "From quantitative analysis to synthesis of efficient binary neural networks," in *Proc. 19th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2020, pp. 93–100.
[44] S. Yin, Z. Jiang, J.-S. Seo, and M. Seok, "XNOR-SRAM: In-memory computing SRAM macro for binary/ternary deep neural networks," *IEEE J. Solid-State Circuits*, vol. 55, no. 6, pp. 1733–1743, Jun. 2020.
[45] *Performance Benchmarking Embedded FPGAs*. Accessed: Sep. 1, 2021. [Online]. Available: https://flex-logix.com/wp-content/uploads/2019/04/2018-05-performance-benchmarking-embedded-FPGAs-r1p2.pdf
[46] *OpenCores*. Accessed: Sep. 1, 2021. [Online]. Available: https://opencores.org
[47] *Ne10 Library*. Accessed: Sep. 1, 2021. [Online]. Available: https://projectne10.github.io/Ne10/
[48] *Eigen Library*. Accessed: Sep. 1, 2021. [Online]. Available: https://gitlab.com/libeigen/eigen
[49] *Tiny-AES-C Library*. Accessed: Sep. 1, 2021. [Online]. Available: https://github.com/kokke/tiny-AES-c
[50] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, vol. 1. Red Hook, NY, USA: Curran Associates, 2012, pp. 1097–1105.
[51] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Representation*, 2015.
[52] A. Rovinski *et al.*, "Evaluating celerity: A 16-nm 695 Giga-RISC-V Instructions/s manycore processor with synthesizable PLL," *IEEE Solid-State Circuits Lett.*, vol. 2, no. 12, pp. 289–292, Dec. 2019.
[53] C. Schmidt *et al.*, "4.3 An eight-core 1.44 GHz RISC-V vector machine in 16 nm FinFET," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2021, pp. 58–60.
[54] B. Zimmer *et al.*, "A 0.32–128 TOPS, scalable multi-chip-module-based deep neural network inference accelerator with ground-referenced signaling in 16 nm," *IEEE J. Solid-State Circuits*, vol. 55, no. 4, pp. 920–932, Apr. 2020.

Sae Kyu Lee (Member, IEEE) received the B.S. degree in electrical engineering from Seoul National University, Seoul, South Korea, in 2006, the M.S. degree in electrical and computer engineering from The University of Texas at Austin, Austin, TX, USA, in 2008, and the Ph.D. degree from Harvard University, Cambridge, MA, USA, in 2016.

He was previously with Intel Corporation, Austin, TX, and Advanced Micro Devices, Boxborough, MA, where he worked on mobile microprocessor design. He is currently with IBM T.J. Watson Research Center, Yorktown Heights, NY, USA. His research interests include energy-efficient accelerator design for machine learning applications and very large scale integration (VLSI) design for efficient on-chip power delivery solutions.

Paul N. Whatmough (Member, IEEE) received the B.Eng. degree (Hons.) from Lancaster University, Lancaster, U.K., in 2003, the M.Sc. degree (with distinction) from the University of Bristol, Bristol, U.K., in 2004, and the Ph.D. degree from University College London, London, U.K., in 2012.

From 2005 to 2008, he was with Philips/NXP Research Laboratories, Redhill, U.K., researching hardware architecture and signal processing for software defined radio from 2005 to 2008. From 2008 to 2015, he was with the Silicon Research and Development Group, ARM Ltd., Cambridge, U.K., working on topics including DSP hardware accelerators, variation tolerance, and system-on-chip (SoC) supply voltage noise. From 2015 to 2017, he was a Research Associate with Harvard University, Cambridge, MA, USA. He currently leads research on hardware for machine learning with Arm Research, Boston, MA, and is a part-time Associate with the School of Engineering and Applied Sciences, Harvard University. He has coauthored the book *Deep Learning for Computer Architects* (Morgan & Claypool, 2017).

Dr. Whatmough is a member of the IET. He was a recipient of the IET Student Project Award in 2003, the IEEE Communications Chapter Award in 2004, and multiple best paper awards. He has served on the technical program committees of numerous conferences in the fields of solid-state circuits, computer architecture, and machine learning.

Marco Donato (Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from the Università di Roma La Sapienza, Rome, Italy, in 2008 and 2010, respectively, and the Ph.D. degree in electrical sciences and computer engineering from Brown University, Providence, RI, USA, in 2016.

From 2017 to 2020, he was a Post-Doctoral Research Associate with the John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Tufts University, Medford, MA. His research interests include novel design methodologies targeting energy-efficient and reliable circuits and architectures for emerging computing paradigms.

Glenn G. Ko (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical and computer engineering from the University of Illinois at Urbana-Champaign, Urbana, IL, USA, in 2004, 2006, and 2017, respectively.

He was previously with Samsung Electronics, Suwon, South Korea, where he worked on mobile application processor system-on-chips. He also spent time with Qualcomm, San Diego, CA, USA, and IBM T.J. Watson Research Center, Yorktown Heights, NY, USA, working on machine learning accelerator architectures and deep learning kernels. He is currently a Research Associate with the Department of Electrical Engineering and Computer Science, School of Engineering and Applied Sciences (SEAS), Harvard University, Cambridge, MA. He is also the CEO of Stochastic, Inc., Cambridge, MA. His research interests include machine learning algorithms, computer architecture, and integrated circuits.

David Brooks (Fellow, IEEE) received the B.S. degree in electrical engineering from the University of Southern California, Los Angeles, CA, USA, in 1997, and the M.A. and Ph.D. degrees in electrical engineering from Princeton University, Princeton, NJ, USA, in 1999 and 2001, respectively.

He is currently the Haley Family Professor of computer science with the School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA. His current research interests include resilient and power-efficient computer hardware and software design for high-performance and embedded systems.

Dr. Brooks was a recipient of several honors and awards including the ACM Maurice Wilkes Award and ISCA Influential Paper Award.

Gu-Yeon Wei (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, USA, in 1994, 1997, and 2001, respectively.

He is a Robert and Suzanne Case Professor of electrical engineering and computer science with the Paulson School of Engineering and Applied Sciences (SEAS), Harvard University, Cambridge, MA, USA. His research interests span multiple layers of a computing system: mixed-signal integrated circuits, computer architecture, and design tools for efficient hardware. His research efforts focus on identifying synergistic opportunities across these layers to develop energy-efficient solutions for a broad range of systems from flapping-wing microrobots to machine learning hardware for Internet of Things (IoT) devices to large-scale servers.