DAP: A 507-GMACs/J 256-Core Domain Adaptive Processor for Wireless Communication and Linear Algebra Kernels in 12-nm FINFET

Kuan-Yu Chen, *Graduate Student Member, IEEE*, Chi-Sheng Yang, Yu-Hsiu Sun, Chien-Wei Tseng, *Graduate Student Member, IEEE*, Morteza Fayazi, *Graduate Student Member, IEEE*, Xin He, Siying Feng, Yufan Yue, *Graduate Student Member, IEEE*, Trevor Mudge, *Life Fellow, IEEE*, Ronald Dreslinski, *Senior Member, IEEE*, Hun-Seok Kim, *Senior Member, IEEE*, and David Blaauw, *Fellow, IEEE*

***Abstract*— We present domain adaptive processor (DAP), a programmable systolic-array processor designed for wireless communication and linear algebra workloads. DAP uses a globally homogeneous but locally heterogeneous architecture, uses decode-less reconfiguration instructions for data streaming, enables single-cycle data communication between functional units (FUs), and features lightweight nested-loop control for periodic execution. Our design demonstrates how configuration flexibility and rapid program loading enable a wide range of communication workloads to be mapped and swapped in less than a microsecond, supporting continually evolving communication standards such as 5G. A prototype chip of DAP with 256 cores is fabricated in a 12-nm FINFET process and has been verified. The measurement results show that DAP achieves 507 GMACs/J and a peak performance of 264 GMACs.**

***Index Terms*— 12-nm FINFET, accelerator, array processor, digital signal processing, domain-specific hardware, multicore, programmability, systolic array, wireless communication.**

I. Introduction

PROMPTED by the demise of Dennard's scaling [1], there is an increased use of accelerators to augment general-purpose processing. These accelerators provided a viable solution to bridge the gap between the growing computational demands and the transistor advancements [2], [3], [4], [5], [6]. However, the application space for customized designs which target a specific computational function is limited, and the gain of increasing the number of accelerators is constrained by on-chip resources. Eventually, this approach will hit the "accelerator wall" [7], [8]. Therefore, the tradeoff between performance, efficiency, and flexibility has become a key concern, especially for wireless communication workloads.

A. Wireless Communication

Crucial to modern society, wireless communications range from radar stations to wearable devices. Given their compute-intensive nature, they require cutting-edge performance, and their ubiquity necessitates high efficiency. Furthermore, with the frequent introduction of new protocols and the need to run various kernels concurrently, flexibility and interoperability are of paramount importance.

B. Limitations of Conventional Computing Platforms

CPUs and GPUs offer versatility but face challenges with performance and efficiency overheads. These overheads stem from operations such as memory transfer, instruction decoding, and unpredictable program execution (e.g., branching) [9]. Field-programmable gate arrays (FPGAs), which offer gate-level reconfigurability, are promising but come with non-trivial hardware overheads. In addition, FPGAs suffer from long reconfiguration times at microsecond scales [10].

Dedicated application-specific integrated circuits (ASICs), while highly efficient, feature a hardcoded datapath. This inflexibility makes them ill-suited for wireless communication tasks that require adaptability to frequently introduced standards and computational kernel modifications. As a result, the challenge would be finding a balance between efficiency and flexibility, coupled with rapid reprogrammability.

C. Kernel Characteristics

Wireless communication kernels focus on data streaming operations with minimal control flow. A single functional unit (FU) operates continuously or periodically for many cycles, streaming data to other FUs in a highly deterministic manner. The acceleration of such kernels is particularly well-suited for systolic-array architectures [11], [12], [13]. While these architectures achieve high efficiency, they traditionally have limited flexibility.

Manuscript received 4 October 2023; revised 3 March 2024 and 16 May 2024; accepted 26 July 2024. Date of publication 27 August 2024; date of current version 30 January 2025. This article was approved by Associate Editor Kathryn Wilcox. This work was supported in part by U.S. Government through the Defense Advanced Research Projects Agency (DARPA) Domain-Specific System On Chip (DSSoC) Program, under Award FA8650-18-2-7860. *(Corresponding author: Kuan-Yu Chen.)*

Kuan-Yu Chen and Xin He were with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA. They are now with Tenstorrent USA, Inc., Austin, TX 78735 USA (e-mail: knyuchen@umich.edu).

Chi-Sheng Yang was with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA. He is now with Apple Inc., Cupertino, CA 95014 USA.

Yu-Hsiu Sun and Chien-Wei Tseng were with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA. They are now with NVIDIA Corporation, Santa Clara, CA 95051 USA.

Morteza Fayazi, Yufan Yue, Trevor Mudge, Ronald Dreslinski, Hun-Seok Kim, and David Blaauw are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA.

Siying Feng was with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA. She is now with ARM Ltd., Austin, TX 78735 USA.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSSC.2024.3438758.

Digital Object Identifier 10.1109/JSSC.2024.3438758

0018-9200 © 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Michigan Library. Downloaded on February 13,2025 at 18:51:25 UTC from IEEE Xplore. Restrictions apply.

D. Proposed Solution

Addressing these challenges, we introduce domain adaptive processor (DAP) [14], a programmable fabric designed to execute a diverse range of wireless communication kernels with near-ASIC energy efficiency. Unlike existing designs that prioritize full programmability or optimization for a specific kernel, DAP focuses on the target domain. It achieves high performance and efficiency while offering the required flexibility. After a thorough study of the workloads, both the architecture and the microarchitecture are optimized. This results in co-optimized hardware implementations and mappings tailored to each kernel's unique characteristics. Moreover, multiple kernels can be mapped onto the fabric simultaneously, forming a complete workload. This reduces memory access and traffic in and out of the fabric, ensuring energy consumption and latency focus on computation. The proposed solution indirectly contributes to overall system efficiency by optimizing baseband processing, which, in turn, can lead to reduced energy consumption system wide. Enhancements in baseband processing efficiency can lower the computational overhead, potentially allowing for more energy-efficient data transmission strategies to be implemented in the system front-end.

The contributions of this work are as follows.

1) We design a systolic-array architecture consisting of lightweight processing elements (PEs) tailored for domain-specific computations in wireless communication.
2) We propose a custom instruction set architecture (ISA), co-designed with the architecture that maximize FUs' parallelism through data-level parallelism, instruction-level parallelism, and pipeline parallelism.
3) We demonstrate examples of mapping wireless communication and linear algebra kernels on DAP, ensuring high throughput and latency reduction.
4) We implement a prototype chip of DAP that includes 256 PEs, which is fabricated in a 12-nm FINFET technology. The performance and efficiency are evaluated by executing more than ten kernels on the prototype chip. Measurement results show that DAP maintains an efficiency to within 2.23× of fixed-function accelerators running the same kernels.

The previous works similar to DAP include Yuan and Marković [15], Cerqueira et al. [16], and Nagi et al. [17]. The former two approaches incorporate a network on chip (NoC) system to connect multiple cores in addition to systolic connections while the last one uses multi-layered interconnects. In contrast, DAP features a pure systolic connection between all units. This design choice results in a lighter architecture by eliminating the need for packet routers, switches, and arbitration, thanks to the deterministic nature of the target domain. The direct connection between cores and FUs more closely mimics an ASIC implementation, which is aligned with DAP's design goal. While Yuan and Marković [15] and Nagi et al. [17] also target wireless communication kernels, leveraging their high data-reuse nature, DAP showcases a broader range of kernel mappings. Conversely, Catena emphasizes on circuit-level optimizations such as power-gating and fine-grained clock-gating for improved energy efficiency. On the other hand, DAP's main concern lies on the microarchitectural level, aiming to maximize hardware reuse and minimize switching activity. A comprehensive discussion on reconfigurable accelerators, dataflow machines, and multicore architectures can be found in Section VII.

E. Article Structure

The remainder of this article is structured as follows. Section II delves into the DAP architecture, while Section III presents the microarchitectural optimizations. Section IV presents examples of kernels and workloads mapped onto DAP. Section V outlines the prototype chip implementation and experimental methodology. Section VI presents and analyzes the measured results.

II. Architecture

In this section, we delve into the architecture of DAP.

A. Design Characteristics

DAP is a programmable accelerator specifically designed to cater to the requirements of wireless communication workloads. It combines traditional techniques with innovative approaches. The major design features of DAP include the following.

1) *Systolic:* DAP's PEs are connected in a systolic array, facilitating direct interconnections. This design helps reduce data transfer latency, increase data reuse, and minimize energy consumption in interconnects.
2) *Programmable:* Each PE in DAP offers programmability in its datapath and FU operations. With a custom ISA, it can activate multiple FUs within a single PE in the same cycle, boosting parallelism and computation throughput.
3) *Concurrence:* Operating under a multiple instruction, multiple data (MIMD) paradigm, each PE functions autonomously. PEs can form kernels, and multiple kernels can run concurrently and independently. This design promotes multitasking and seamless integration of inputs and outputs from different kernels, optimizing data reuse and reducing memory access.

B. Architecture Overview

Fig. 1 shows a high-level view of DAP's architecture. The core of DAP are the PEs, interconnected with neighboring PEs in all eight directions to form a tight matrix. DAP is optimized for 32-bit fixed-point complex number computations, and it supports four different PE types equipped with FUs tailored for complex and real number arithmetic (Table I). These PEs group into locally heterogeneous clusters. However, these clusters are replicated across the fabric for uniformity. The custom ISA, detailed in Section II-D, facilitates the simultaneous activation of multiple FUs ("parallelism" column in Table I), ensuring high throughput.

For data transfers, DAP uses a systolic approach, eliminating the need for an additional NoC. Global scratchpad memories, positioned at the borders of each systolic array row, store input/output data and essential programming instructions for the PEs. Management units (MUs) connect the PE array

Authorized licensed use limited to: University of Michigan Library. Downloaded on February 13,2025 at 18:51:25 UTC from IEEE Xplore. Restrictions apply.

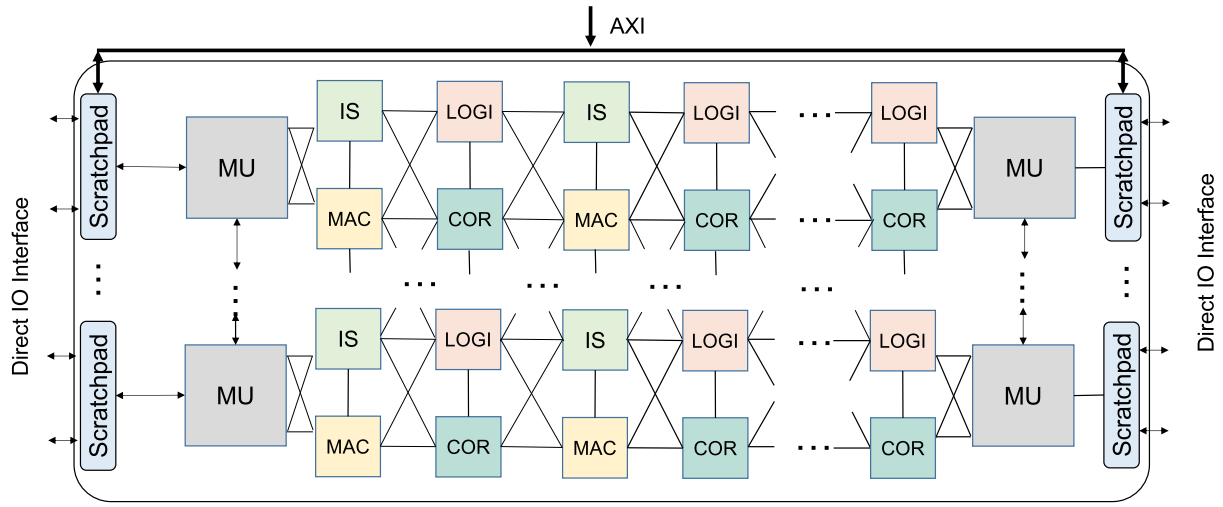


Fig. 1. DAP architecture overview. Four types of PEs with different FUs form a locally heterogeneous, globally homogeneous systolic array of 256 PEs. Scratchpad memory is connected to the PE array through MUs.

TABLE I
DIFFERENT TYPES OF PES PROVIDE VARIOUS AMOUNT OF PARALLELISM, CONTAIN DIFFERENT FUS, AND SUPPORT A WIDE RANGE OF COMPUTATION

Type	Function	Parallelism	Operation
MAC	MAC	6	Multiplication, Addition, Subtraction, Convolution
COR-DIV	Nonlinear	2	Rotation, Vectorization Division, Square Root
IS	Buffer	2	Indexed Access FIFO, FILO
Logical	Logical	2	Bitwise Logical Compare, RELU

and the global scratchpads. Each MU oversees two PE rows, ensuring seamless data movement, reliable data streams, and PE state configurations.

C. PE Overview

Each PE in the array can transition between four operational states. **LOAD** for loading programs into the instruction memory (IMEM), **EXECUTE** for program execution, **ROUTE** for acting as a programmable router, and **IDLE** where everything is clock-gated. In the **ROUTE** state, once the connections between ports are defined at the program's initiation, the architecture enables flexible dataflow throughout the systolic array. This design eliminates the need for repeated packet arbitrations and ensures efficient communication between non-adjacent PEs without the necessity of loading extensive programs solely for data transfer.

Fig. 2(a) displays the general architecture of the PEs. While the foundational structure remains consistent across PE types, the FUs vary. Non-FU components ensure flexibility and swift reconfiguration in the dataflow and control. The common units [Fig. 2(a)] are: State control, loop control, and the register unit. The state control unit, configured via a pipelined bus connected to the MU, stores the current operational state of the PE. Another essential component is the loop control, which acts as a program counter in sync with the instruction memory, and it is designed to support nested "for" loops.

The register unit manages the connections of the crossbar and internal PE datapaths, including data buffering, FU mode configuration, and crossbar settings' adjusting. This results in seamless, direct streaming between FUs, guaranteeing zero-cycle latency for inter-FU communications and significantly optimizing data movement while conserving buffering energy. The breakdown of area and power of different components in all PE types is shown in Table II, while the distribution of each type of PE within the PE array is shown in Table III.

The highlight of the architecture is the decoder-less FU activation paired with the dynamic routing reconfiguration. Combined with an efficient nested-loop program control, these elements substantially reduce overhead, demanding a minimal instruction memory footprint of only 128 entries per PE.

D. Stream Instruction

Fig. 2(b) portrays the intricate architecture of an MAC PE, detailing control signals and datapath. Communication kernels depend on data streaming operations with little or no control flow where a single FU is executed continuously or periodically for many cycles, streaming data into other FUs in a highly deterministic manner. The traditional cycle-upon-cycle instruction fetch and decode has been replaced with stream instructions that set the crossbar's data flow configuration and the operation of FUs (which are enabled or not) concurrently and, combined with loop control, determine the duration of the configuration. The instruction fields are directly copied into the control registers after the instruction is fetched from the instruction memory. Therefore, the instruction decode overhead is essentially eliminated.

All the operations, including loop control, activation, and configuration, are executed in the same cycle. Hence, a single instruction can stay in place for many cycles, greatly minimizing control overhead. Since instruction fields are directly copied into the control registers, instruction decode overhead is essentially eliminated and since embedded loop control greatly

Authorized licensed use limited to: University of Michigan Library. Downloaded on February 13,2025 at 18:51:25 UTC from IEEE Xplore. Restrictions apply.

TABLE II
AREA/POWER BREAKDOWN OF DIFFERENT COMPONENTS IN ALL TYPES OF PES

PE Type	MAC		IS		CORDIV		Logical		Average	
Distribution	Area (%)	Power (%)	Area (%)	Power (%)	Area (%)	Power (%)	Area (%)	Power (%)	Area (%)	Power (%)
Functional Units	35.4	85.2	67.9	61.8	53.8	78.2	11.7	47.2	49.6	80.5
PE Control	0.53	1.06	0.62	5.92	0.81	3.26	1.80	8.22	0.76	2.67
Loop Control	1.17	0.38	1.38	1.88	1.79	1.05	3.90	2.58	1.74	0.81
Datapath	45.8	10.6	17.5	20.6	27.0	12.1	45.7	28.4	29.3	11.0
Instruction Memory	17.1	2.76	12.6	9.80	16.6	5.39	36.9	13.6	18.6	5.02

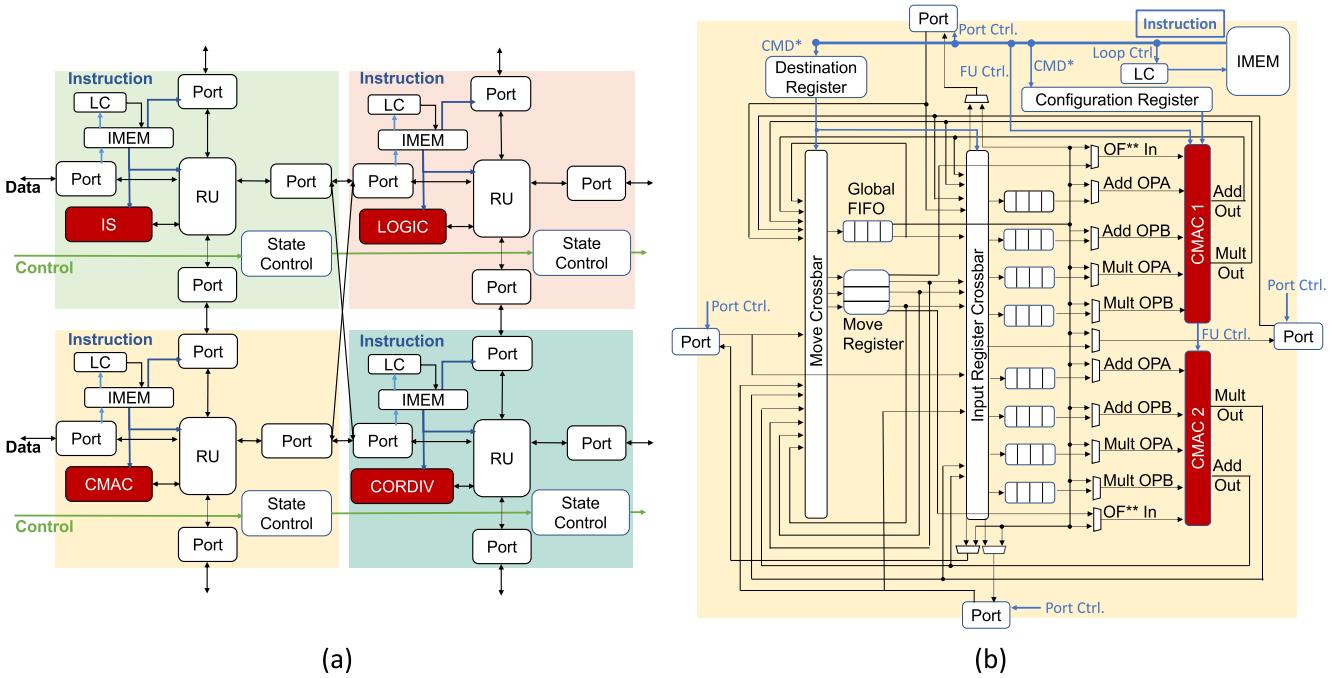


Fig. 2. (a) Overview of PE architecture of all four types of PE, the only difference is the FU type. (b) Detail dataflow (black lines) and control signal (blue lines) of an MAC-type PE.

TABLE III
DISTRIBUTION OF DIFFERENT TYPES OF PES IN THE PE ARRAY

PE Type	Area (%)	Power (%)
MAC	36.8	61.8
IS	31.9	11.1
CORDIV	24.3	20.2
Logical	7.00	6.90

TABLE IV
CONSTRUCT OF INSTRUCTION FIELDS IN DIFFERENT TYPES OF PE, ACTIVATION ACTIVATES PORTS AND FUS, LOOP CONSTRUCT FOR LOOPS IN THE PROGRAM, AND CONFIGURATION CONTROLS THE CROSSBAR AND FU MODES THROUGH THE REGISTER UNIT

PE Type	Activation	Loop	Configuration	Total
MAC	10 bits	2 bits	42 bits (4 in parallel)	54 bits
IS	8 bits	2 bits	17 bits (2 in parallel)	27 bits
COR-DIV	6 bits	2 bits	19 bits (2 in parallel)	27 bits
Logical	6 bits	2 bits	19 bits (2 in parallel)	27 bits

reduces program size. An entire DAP can be programmed hundreds of cycles (sub-μs), enabling on-the-fly kernel swapping (unlike FPGAs) for simultaneous support of multiple protocols that share PEs in a time-multiplexed fashion. The exact bitwidth of each field is specified in Table IV.

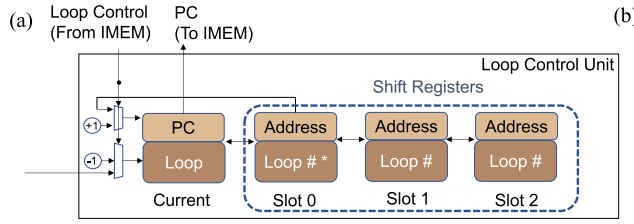
E. Detailed Dataflow

The dataflow's complexity extends beyond control signals due to potential interactions among ports, FUs, and various storage registers through two crossbars. This intricate connection allows DAP's dataflow to mimic fixed datapath hardware accelerators. During execution, data stream from one unit to another within a single cycle. Transferring data from an FU in one PE to another FU in a neighboring PE takes only two cycles, marking a significant enhancement in communication efficiency compared with NoC-based designs. Traditional CPUs typically implement several FUs and registers, leading to large, multi-port register files, resulting in significant power and area overhead. Instead, we restrict the data movement based on common compute patterns and handle connections between the FUs with two sequential crossbars that are pre-set with the stream instructions.

Each FU's inputs are directly connected to specific registers (or small queues with four entries in the MAC PE) thereby eliminating the need for multiple register outputs. The FU's outputs then connect to the 12-input to 12-output crossbar

Authorized licensed use limited to: University of Michigan Library. Downloaded on February 13,2025 at 18:51:25 UTC from IEEE Xplore. Restrictions apply.

(a)



*Loop # (slot 0) is updated with Loop, i.e. Loop # (slot 0) always equals to Loop
**Upon a right shift, Addr. # (slot 0) = PC, Loop # (slot 0) = input loop count i-1

(b)

Loop Control	PC_next	Loop_next	Shift Register Action
Start [i]	PC+1	i-1	Address / loop # registers shift right**
End (loop > 0)	Address (Slot 0)	Loop - 1	Loop # (slot 0) = Loop - 1
End (loop = 0 & slot 1 invalid)	PC+1	N/A	Address / loop # registers shift left
End (loop = 0 & slot 1 valid)	PC+1	Loop # (slot 1)	Address / loop # registers shift left
Other	PC+1	Loop	N/A

Fig. 3. (a) Loop control architecture that supports up to three nested "for" loops. (b) Table of operations for different "loop control" field in the instruction.

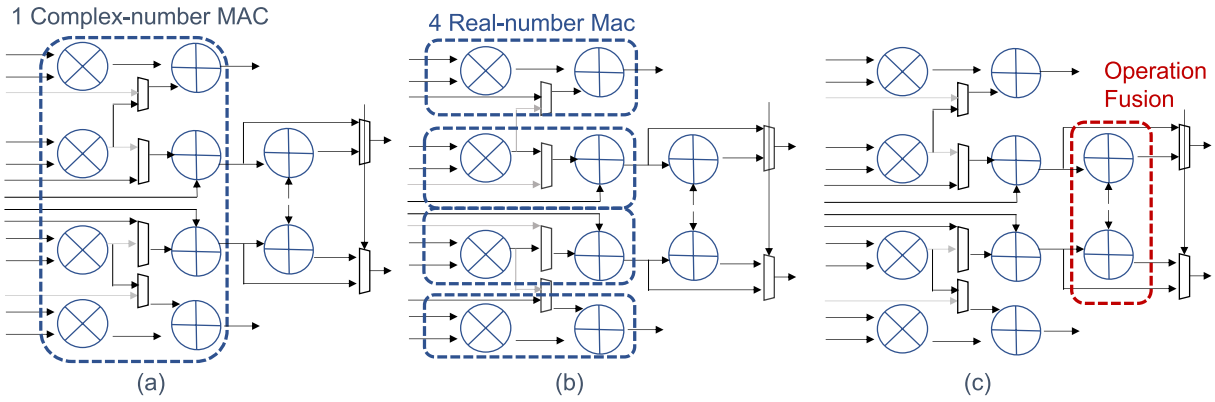


Fig. 4. Different operations of the CMAC unit in DAP, including (a) complex mode, (b) real mode, and (c) operation fusion.

which allows direct FU-to-FU streaming. In addition, data can be routed into Move registers or a global FIFO (four entry) through a smaller 16-to-4 crossbar. Global FIFO and Move registers are independent of the input register of FUs to reduce the complexity of crossbars. Data can be routed into Move registers or global FIFO through a smaller crossbar to allow additional data storage, data alignment, and data broadcasting to a selectable set of FU inputs.

III. Microarchitecture

This section discusses various microarchitecture optimizations that improve the performance and efficiency of DAP.

A. Loop Control

The implementation of nested "for" loops in DAP programs is supported by a loop control unit (Fig. 3) in each PE and 2-bit loop field (Table IV) in the instructions. Shift registers record the return address and the remaining number of loop iterations. Both the registers shift to the right when entering a new loop and shift to the left when the loop number reaches zero. We implemented three levels of shift registers, supporting three nested loops, which were found sufficient for a wide range of communication kernels. Infinite loop and looping a single instruction are also supported.

B. Dual-Mode CMAC Unit

There are two CMAC units (Fig. 4) inside each MAC PE. Each can be reconfigured as either one complex- number MAC [Fig. 4(a)] or four real-number MACs [Fig. 4(b)]. This feature greatly benefits real-value kernels such as 2-D convolution (2DConv) by providing 4× the number of MAC units (eight total for the MAC PE).

C. Operation Fusion

Fig. 4(c) illustrates the concept of operation fusion. Multipliers in the CMAC FU occupy two pipeline stages to meet the target frequency and adders can meet the timing constraints in a single stage. However, the multipliers continue to set the clock frequency, leaving adders with significant delay slack. We use this slack, by adding two operation-fused adders which execute in a single cycle with the CMAC adders, providing additional computation without impacting clock frequency.

Operation fusion is especially useful in supporting additions of three elements. For example, the bias addition and partial sum accumulation of 2DConv can be merged into a single CMAC unit. Operation fusion is also applied to the logical units for cascaded operations such as comparisons to further reduce latency.

D. Management Unit

The MU, which serves as a bridge between the scratchpad memory and the PE array, is shown in Fig. 5. Inside the MU, there are memory mapped registers, PE managers, and data transceivers. The memory mapped registers configure the data transceivers, PE manager, and data crossbar. The PE managers are connected to the control bus that is connected to the "state control" block in each PE. The PE managers can change the PE states, start or stop the PE, and program the routing direction when the PE is in ROUTING state. The data transceiver provides various address generation patterns and can modify data on the fly. Address generation includes temporal access to change the throughput and spatial access, such as bit reverse ordering for the fast Fourier transform (FFT). Data modification such as conjugation and real/imaginary part

Authorized licensed use limited to: University of Michigan Library. Downloaded on February 13,2025 at 18:51:25 UTC from IEEE Xplore. Restrictions apply.

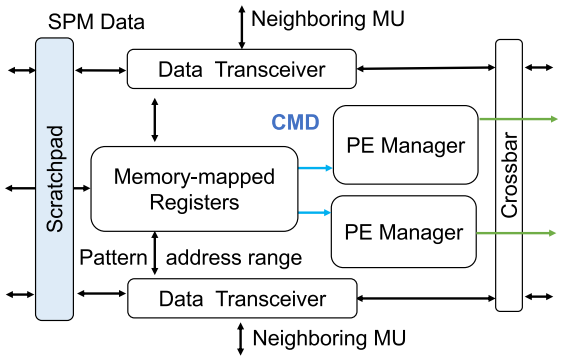


Fig. 5. DAP MU between the scratchpad memories and the PE array, including PE manager, data transceiver, and memory mapped register.

swapping is also supported. The crossbar provides flexible dataflow. For example, data from one transceiver can be multicast into multiple PEs, and multiple datastream can also be time-interleaved into the same PE.

IV. Mapping

This section presents examples of mapping various kernels and workloads on DAP to demonstrate its flexibility. The main considerations for mapping computations on DAP are as given below.

1) Ensuring high throughput for streaming computations.
2) Enabling back-to-back computation for kernels that operates on blocks of data with clear boundary (e.g., FFT).
3) Minimizing switching activity on datapath.

A. Max/Min Filter

A 1-D max/min filter implementation on DAP follows the divide-and-conquer algorithm proposed by Coltuc and Bolon [18]. Fig. 6 illustrates an example of a max filter with a window size of 3. The FILOs are implemented by the IS PEs, and the comparison is done by the logical PEs. Both the recursive and elementwise operations are realized by the flexible dataflow inside PEs. This mapping scheme is scalable for any window size and ensures the throughput of one output per cycle with the same number of PEs.

B. Two-Dimensional Convolution

Fig. 7 illustrates the mapping of 2DConv with one input and two kernels. Note that "kernel" here should not be confused with kernel mapping on DAP. Each kernel is assigned to a row of PEs, and an entire row in a kernel is stored inside one MAC PE, using the many real number MACs from the MAC units. This mapping reuses the row-stationery concept proposed by Chen et al. [19], [20]. The IS PEs are used as buffers for storing and transferring partial sums to the next stage for accumulation.

C. Orthogonal Frequency-Division Multiplexing

Fig. 8 shows the mapping of orthogonal frequency-division multiplexing (OFDM) on DAP. There are two phases in OFDM; packet detection is performed in phase 1, while FFT, channel estimation, and demodulation are computed in phase 2. During packet detection, the output of finite impulse response (FIR) is directly used as the input of auto-correlation without leaving the PE array, which reduces memory access. The summation and multiplication are all assigned to the MAC PEs, and the other PEs serve as routers. Upon packet detection, DAP is reprogrammed to reuse the same PEs for channel estimation, FFT, and symbol demodulation. The COR-DIV PE (Table I) is used for channel estimation, and the demodulation stage is fused with the last butterfly in FFT.

D. Multiple-Input Multiple-Output Minimum Mean Square Error

Fig. 9 shows the mapping of multiple-input multiple-output (MIMO) minimum mean square error (MMSE) detection. The first phase is to compute the MMSE matrix with a combination of kernels, including matrix multiplication, QR decomposition, and back substitution. Then, we reprogram the array to perform matrix–vector multiplication, which is the second phase. We take the MMSE matrix computed in the first phase and multiply it with the incoming vectors.

V. Prototype Chip and Testing Methodology

This section describes the implementation of the DAP prototype chip and presents an overview of the benchmarking methodology and evaluation criteria.

A. Prototype Chip Implementation

The prototype chip of DAP is implemented in a 12-nm FinFET process and occupies 21 mm^2 die area (Fig. 10). There are 256 PEs in the prototype chip, 64 for each type. Since all the connections between the PEs and the peripheral blocks are systolic, the system is highly scalable even if implemented in different technology nodes. The design is partitioned hierarchically with PEs as hard macros. An on-chip digital-controlled oscillator drives a single clock tree, capable of spanning clock frequencies from 40 MHz to 2 GHz. To measure the operating frequency, the system clock is routed through a 1024-divider and connected to an oscilloscope via an output pad. At 1.0 V, the system operates at 506 MHz, corresponding to 969 mW and 264 GMACs.

B. Baseline and Test Methodology

The prototype chip is evaluated against fixed-function accelerators (Anders et al. [21] and Desoli et al. [22]) to showcase its ability to provide generalizations without incurring significant overhead. In addition, DAP is also compared with programmable processors (Yuan and Marković [15], Cerqueira et al. [16], and Nagi et al. [17]) to demonstrate the effectiveness of domain specialization.

To guarantee a balanced comparison that takes into account varying technology nodes and performance benchmarks, a meticulous technology scaling at iso-throughput was applied based on SPICE simulations of FO4 energy and delay. The energy consumption is measured from a single power supply;

Authorized licensed use limited to: University of Michigan Library. Downloaded on February 13,2025 at 18:51:25 UTC from IEEE Xplore. Restrictions apply.

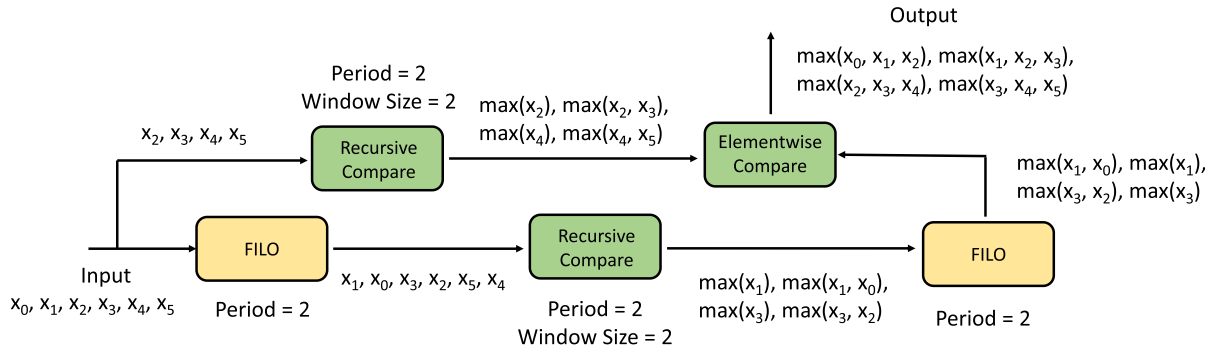


Fig. 6. Mapping of max filter with a window size of 3 on DAP, five PEs are used, including two IS PEs for data shuffling and three logical PEs for actual comparison.

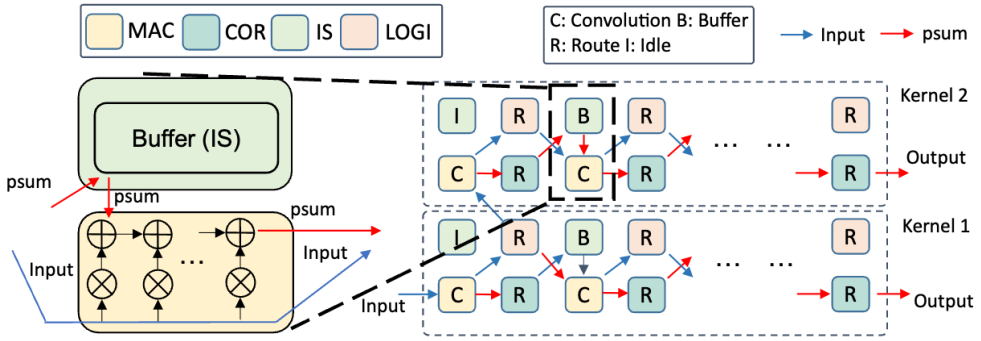


Fig. 7. Mapping of 2DConv on DAP, one input matrix with two kernels for convolution.

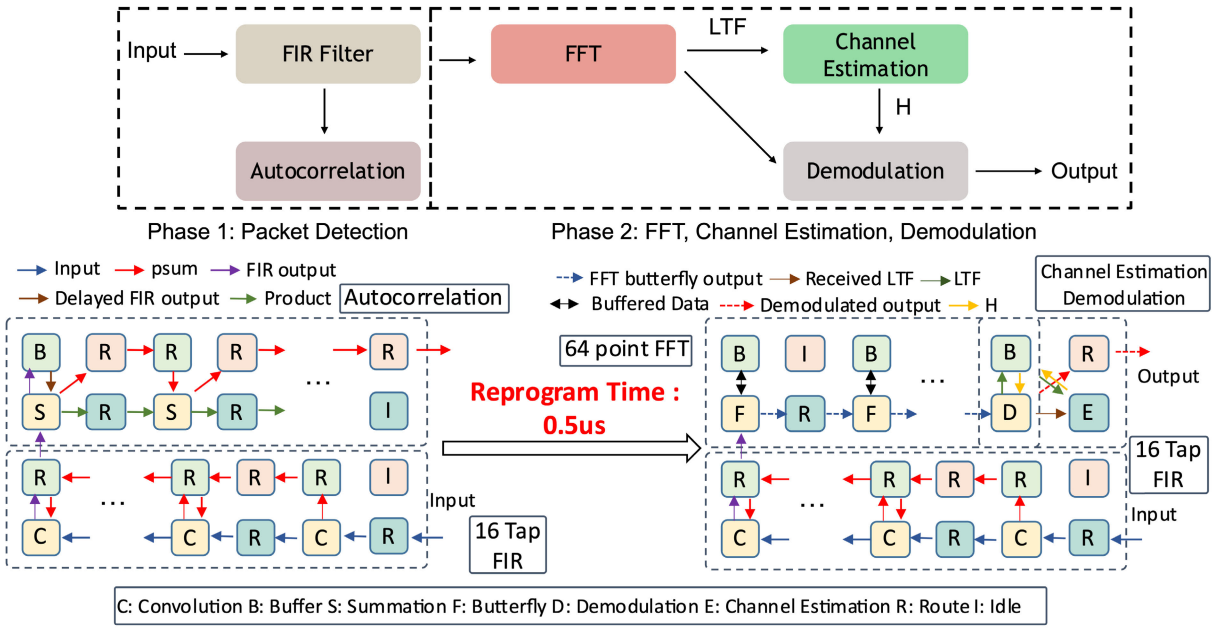


Fig. 8. Mapping of OFDM on DAP FIR and autocorrelation in phase 1 and FIR, FFT, channel estimation, and demodulation in phase 2.

all major portions of the test chip (core logic, on-chip SRAM, and clock network) are included in the measurements.

Over 15 computational kernels have been verified on DAP. However, for brevity and clarity, only those with well-defined benchmarks are showcased in Section VI. These kernels include a variety of operations common to the wireless communication domain, e.g., FIR, FFT, and QR decomposition. Other kernels include linear algebra kernels, 2Dconv, and general matrix multiplication (GeMM). The performance and efficiency of comprehensive workloads such as OFDM and MIMO MMSE detection were also measured. The detailed mapping of the kernels and workloads on DAP was covered in Section IV. In selecting kernels and workloads, the aim was to ensure a broad coverage across various FUs within different PEs. For instance, FIR primarily targets MAC operations, while kernels such as 2Dconv, FFT, and GeMM use the

Authorized licensed use limited to: University of Michigan Library. Downloaded on February 13,2025 at 18:51:25 UTC from IEEE Xplore. Restrictions apply.

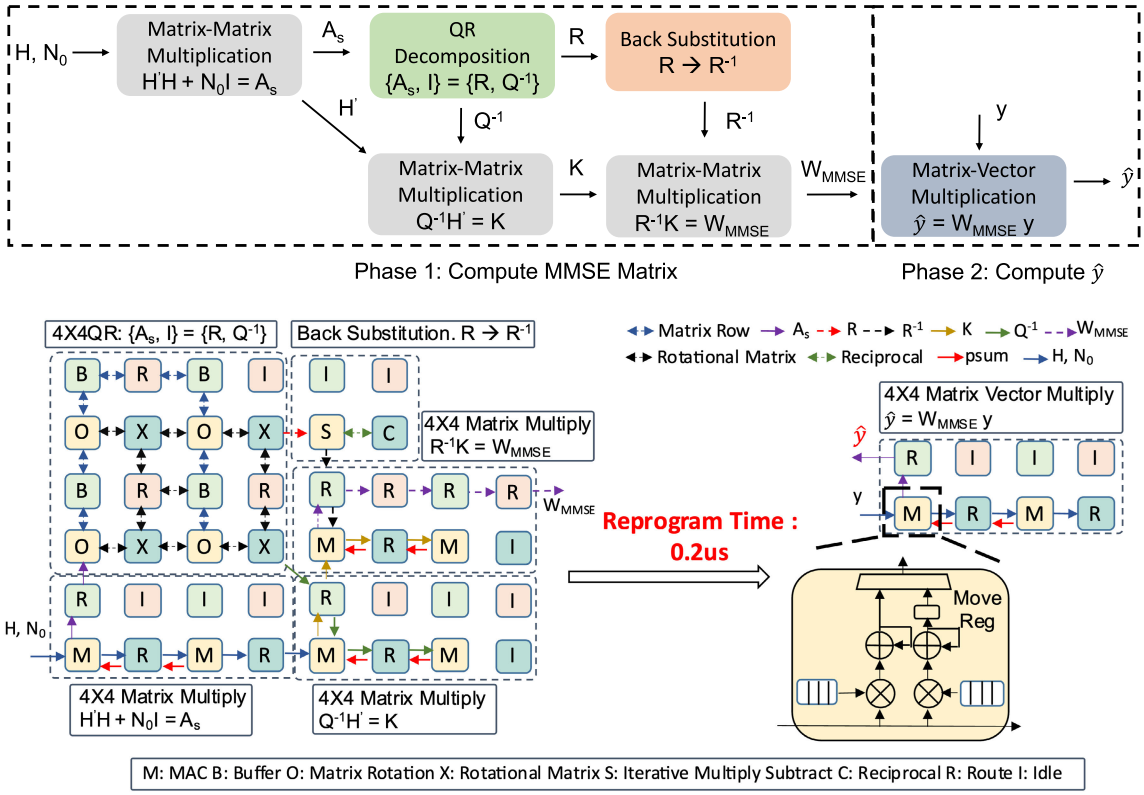
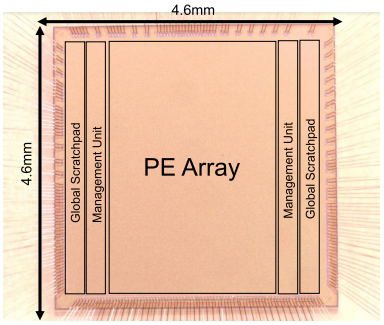


Fig. 9. Mapping of MIMO on DAP: GeMM, QR decomposition, back substitution in phase 1 and matrix–vector multiplication in phase 2.

IS PEs. Distinct operations such as QR decomposition and back substitution require CORDIC and divider operations. Traditional metrics such as "OPs/W" and "OPs" were deemed unsuitable due to the challenges of normalizing CORDIC and divider functionalities into "OP." Instead, different metrics tailored to each kernel's characteristics such as "MACs/J," "FFT/J," and "Matrix/J" are presented.

VI. Measurement

This section presents the measurement results of executing the kernels mentioned in Section V on DAP, followed by voltage–frequency scaling measurement, and comparison with the prior works discussed in Section V.



Technology	GF 12nm CMOS
Area (mm²)	21.16
Voltage (V)	0.42-1.0
Frequency (MHz)	8.55-506
Peak Efficiency (GMACs/J)	507
Peak Performance (GMACs)	264
On-Chip Memory (KB)	522

(b)

Fig. 10. (a) Micrograph of prototype chip and (b) summary table of DAP.

A. Measurement Results of Individual Kernels and Workloads

Table V summarizes the throughput and efficiency of different benchmarks on two operating points of DAP. The peak performance operating point was measured when supplied with a nominal voltage, and the other was assessed by scaling down the voltage, ensuring that adequate throughput is still achieved. These kernels include FIR, GeMM, 2DConv, FFT, QR decomposition, back substitution, OFDM, and MIMO.

FIR, GeMM, and 2DConv are similar kernels that largely involve MAC computations. However, their maximum throughput and efficiency differ due to individual dataflow and mapping characteristics. The first difference arises from the use of IS PEs. Both GeMM and 2DConv require IS to store matrix weights and partial sum, respectively, whereas FIR relies solely on MAC PEs. Second, utilization of FUs varies. FIR and 2DConv can map computations onto both CMAC units in an MAC PE, while GeMM is constrained to using just one, since the neighboring IS PE only stores one row. In other words, to compute the same amount of MAC operations per cycle, GeMM would require twice the number of PE, which accounts for the 40% drop in the peak efficiency when compared with FIR.

The performance and efficiency of FFT are close to 2DConv when measured in GMACs and GMACs/J since the two shared similar datapath and computation (Section IV). However, Gsamples/s and nJ/FFT are more appropriate metrics. QR decomposition, back substitution, and MMSE are iterative and more compute-intensive, resulting in lower throughput.

The number of PE required to compute each kernel and workload is also listed in Table V. However, the PEs that are

Authorized licensed use limited to: University of Michigan Library. Downloaded on February 13,2025 at 18:51:25 UTC from IEEE Xplore. Restrictions apply.

TABLE V
MEASUREMENT RESULTS OF KERNELS/WORKLOADS

Kernel / Workload		Peak Performance		Efficiency / Throughput Trade-off		Number of Required PEs*
		Throughput	Efficiency	Throughput	Efficiency	
FIR (GMACs, GMACs/J)		252.2	254.5	45.19	409	0.5 / tap
GeMM (GMACs, GMACs/J)		148.1	162.2	42.33	242.13	2 / row
2DConv (GMACs, GMACs/J)		231.7	250.5	58.35	375.63	2 / row
256 pt FFT (Gsamples/s, nJ/FFT)		4.41	53.96	0.974	31.6	13
4 X 4 QR (Mmatrix/s, nJ/matrix)		16.07	19.3	3.54	8.62	12
4 X 4 Back Substitution (Mmatrix/s, nJ/matrix)		107	2.69	15.67	1.24	2
OFDM (Gbits/s, Gbits/J) FIR, FFT, autocorrelation		46.46	59.57	9.896	108.74	19
MIMO	MMSE (Mmatrix/s, nJ/matrix)	1.95	178.5	0.33	82.89	20
	DMV (Gbits/s, Gbits/J)	213.12	310.68	34.87	576.76	2

* Routing PEs excluded.

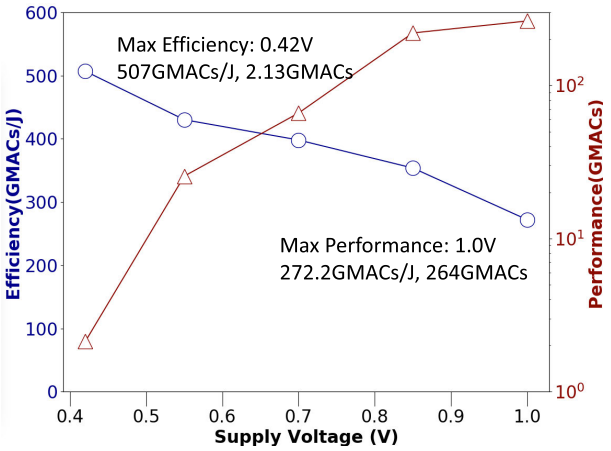


Fig. 11. DAP efficiency and performance at different supply voltage levels.

used for routing are excluded since the number depends on PE connection topology and shape of the mapped kernel.

B. Voltage–Frequency Scaling Analysis

Fig. 11 shows the peak performance and peak efficiency of DAP under different supply voltages. GMACs and GMACs/J are chosen as metrics instead of OPs. Therefore, non-linear functions such as CORDIC and division are excluded, and a non-terminating, artificial kernel that maximizes PE utilization is mapped on DAP for measurement. The prototype chip of DAP achieves a peak efficiency of 507 GMACs/J at 0.42 V and a peak performance of 264 GMACs at 1.0 V.

C. Comparison With Similar Prior Works

Table VI shows comparison of DAP with hardware accelerators Desoli et al. [22] and Anders et al. [21], on 2DConv and GeMM. All the bitwidths are 16 bits, and the listed results are scaled using iso-throughput and FO4 SPICE simulation to accommodate for the difference in voltage, frequency, and technology node. DAP maintains an efficiency to within 2.23× of when compared with hardware accelerators. This is due to the nature of FUs directly connected to each other, enabling DAP to form ASIC-like dataflow. Table VII shows comparison of DAP with previous works Yuan and Marković [15], Cerqueira et al. [16], and Nagi et al. [17], three programmable processors. Out of the four, DAP has the largest number of cores per die and the largest amount of average data memory per core. The prototype chip is more efficient than Catena, within 1.5× of the work of Yuan and Marković [15], and within 1.3× of the work of Nagi et al. [17] DAP reports the greatest number of mapped kernels.

TABLE VI
COMPARISON WITH DEDICATED ACCELERATORS, ANDERS ET AL. [21] AND DESOLI ET AL. [22]

	2D Convolution		GeMM	
	This Work	ISSCC'17 [22]*	This Work	VLSI'18 [21]**
Technology	12 nm	28 nm	12 nm	14 nm
Voltage (V)	0.65	0.575	0.65	0.9
Efficiency (GMACs/J)	375	810	242	541
Energy Gap	2.16×		2.23×	

all bitwidth are 16 bits
*Listed numbers are scaled to iso-throughput and technology
**Throughput cannot match DAP, compare with max throughput reported

TABLE VII
COMPARISON WITH PROGRAMMABLE ARCHITECTURES

	This Work	VLSI'14 [15]*	VLSI'19 [16]***	JSSC'23 [17]
Technology	12nm	40nm	65nm	16nm
Total Cores	256	16	16	784
Cores per Die	256	16	16	196
Data Memory per Core (KB)	1.47	None	0.5	0.03
Frequency Range (MHz)	8.55-506	25-500	0.1-20****	80-1100****
FIR (GMACs/J)	409, 0.65V	467**, 0.73V	26, 0.54V	543**, 0.42V
4x4 QR (nJ Matrix)	8.62, 0.65V	6.05**, 0.49V	N/A	N/A
256 Point FFT (nJ/FFT)	31.6, 0.65V	N/A	1855, 0.54V	N/A
Matrix Multiplication***** (GMACs/J)	422, 0.65V	N/A	N/A	551**, 0.42V
Max Throughput per Die (GMACs)	264	126	N/A	500****
Number of Kernels	17	5	5	4

all bitwidth are 16 bits
*includes only compute cores, interface and peripheral circuits
** Calculated from plot and scaled to iso-throughput and for technology
*** Throughput cannot match DAP, compare with max throughput reported.
**** estimated based on plots.
***** JSSC'23 [17] reports real number 8x8 matrix multiplication. DAP can perform both real and complex number matrix with larger sizes. VLSI'19 [16] can perform matrix multiplication, but efficiency cannot be calculated as throughput is not provided.

VII. RELATED WORKS

In addition to the prior works discussed in Sections I and VI, DAP is compared with programmable architecture for wireless

Authorized licensed use limited to: University of Michigan Library. Downloaded on February 13,2025 at 18:51:25 UTC from IEEE Xplore. Restrictions apply.

communication, reconfigurable hardware, multi-core processor with systolic connection, and spatial dataflow architecture.

A. Programmable Architecture for Wireless Communication

Various programmable architectures have been developed for the wireless communication domain [23], [24], [25], [26]. Some works run an entire workload concurrently while others focus on processing a single kernel at a time. Asynchronous array of simple processors (AsAP) and its related works [27], [28], [29] use fine-grained clock control for different processing units and demonstrate the mapping of an entire workload onto the compute fabric. SODA [30] is a multi-core DSP processor that captures algorithmic behavior with a wide single instruction, multiple data (SIMD) unit. Pedram et al. [31] and REVEL [32] highly emphasize on matrix factorization kernels that are hard to vectorize due to their inductive nature. REVEL proposed a hybrid systolic array architecture composed of simple systolic compute cores and more complex dataflow cores, while Pedram et al. [31] thoroughly analyzed the algorithms and mapped them onto a homogeneous architecture. DAP demonstrates mapping examples of both single kernels and entire workloads, providing fast reconfigurability and high efficiency.

B. Reconfigurable Hardware

The ASICs from [33], [34], and [35] leverage programmable interconnect in respective targeted applications. Smets et al. [33] used programmable routers to support multiple image processing kernels, while the ASIC implementations from [34] and [35] reconfigure between the multiplication and merge phases of an outer-product-based sparse matrix multiplication algorithm. These ASICs use reconfigurability to broaden the capabilities of a fixed-function design. In contrast, DAP supports a wider variety of computation kernels while retaining performance and efficiency.

C. Multi-Core Processor With Systolic Connection

The designs [23], [36], [37], [38] are multi-core processors providing spatial data transfer through systolic-like connection. All the connections in DAP are systolic and programmable without the need to implement a complicated NoC system. This enables DAP with performance efficiency characteristics close to equivalent ASIC implementations of different computation kernels.

D. Spatial Dataflow Architecture

DAP is also compared with spatial architecture with dataflow execution [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [62], [63]. Dataflow architectures [41], [64], [65], [66], [67] and hybrid dataflow von-Neumann architectures [42], [68] are different from von-Neumann execution and instead map dataflow graphs onto architecture explicitly while execution is triggered by data arrival. In comparison, DAP retains PC-based execution but forms dataflow-like connection between FUs in steady state of execution.

VIII. Conclusion

This work introduces DAP, a domain-specific processor that aims to accelerate the computation of wireless communication and linear algebra kernels. DAP is a programmable accelerator with a co-designed custom ISA tailored for the target domain. DAP maintains an efficiency to within 2.23× of the fixed-function accelerators that execute only a single kernel. Among programmable domain-specific processors, it has the highest number of reported mapped kernels while providing fast re-programmability. In short, DAP successfully balances performance, flexibility, and efficiency.

References

[1] R. Dennard, F. Gaensslen, H.-N. Yu, V. Rideout, E. Bassous, and A. LeBlanc, "Design of ion-implanted MOSFET's with very small physical dimensions," *IEEE J. Solid-State Circuits*, vol. SSC-9, no. 5, pp. 256–268, Oct. 1974.

[2] J. Cong, M. A. Ghodrat, M. Gill, B. Grigorian, K. Gururaj, and G. Reinman, "Accelerator-rich architectures: Opportunities and progresses," in *Proc. 51st ACM/EDAC/IEEE Annu. Design Autom. Conf.*, Jun. 2014, pp. 1–6.

[3] I. Magaki, M. Khazraee, L. V. Gutierrez, and M. B. Taylor, "ASIC clouds: Specializing the datacenter," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 178–190.

[4] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," *IEEE Micro*, vol. 32, no. 3, pp. 122–134, May 2012.

[5] M. B. Taylor, "Is dark silicon useful? Harnessing the four horsemen of the coming dark silicon apocalypse," in *Proc. Des. Autom. Conf.*, 2012, pp. 1131–1136.

[6] Y. S. Shao and D. Brooks, "Research infrastructures for hardware accelerators," *Synth. Lect. Comput. Archit.*, vol. 10, no. 4, pp. 1–99, Nov. 2015.

[7] A. Fuchs and D. Wentzlaff, "The accelerator wall: Limits of chip specialization," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2019, pp. 1–14.

[8] V. Dadu, J. Weng, S. Liu, and T. Nowatzki, "Towards general purpose acceleration by exploiting common data-dependence forms," in *Proc. 52nd Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2019, pp. 924–939.

[9] R. Hameed et al., "Understanding sources of inefficiency in general-purpose chips," in *Proc. 37th Annu. Int. Symp. Comput. Archit.* New York, NY, USA: Association for Computing Machinery, Jun. 2010, pp. 37–47, doi: 10.1145/1815961.1815968.

[10] K. Vipin and S. A. Fahmy, "FPGA dynamic and partial reconfiguration: A survey of architectures, methods, and applications," *ACM Comput. Surv.*, vol. 51, no. 4, pp. 1–39, Jul. 2018, doi: 10.1145/3193827.

[11] Kung, "Why systolic architectures?" *Computer*, vol. 15, no. 1, pp. 37–46, Jan. 1982.

[12] H. T. Kung, "Systolic communication," in *Proc. Int. Conf. Systolic Arrays*, 1988, pp. 695–703.

[13] O. Menzilcioglu, H. T. Kung, and S. W. Song, "Comprehensive evaluation of a two-dimensional configurable array," in *Proc. 19th Int. Symp. Fault-Tolerant Comput. Dig. Papers*, 1989, pp. 93–100.

[14] K.-Y. Chen et al., "A 507 GMACs/J 256-core domain adaptive systolic-array-processor for wireless communication and linear-algebra kernels in 12 nm FINFET," in *Proc. IEEE Symp. VLSI Technol. Circuits*, Jun. 2022, pp. 202–203.

[15] F.-L. Yuan and D. Marković, "A 13.1 GOPS/mW 16-core processor for software-defined radios in 40 nm CMOS," in *Proc. Symp. VLSI Circuits Dig. Tech. Papers*, Jun. 2014, pp. 1–2.

[16] J. P. Cerqueira, T. J. Repetti, Y. Pu, S. Priyadarshi, M. A. Kim, and M. Seok, "Catena: A 0.5-v sub-0.4-mW 16-core spatial array accelerator for mobile and embedded computing," in *Proc. Symp. VLSI Circuits*, 2019, pp. C54–C55.

[17] S. S. Nagi, U. Rathore, K. Sahoo, T. Ling, S. S. Iyer, and D. Markovic, "A 16-nm 784-core digital signal processor array, assembled as a 2×2 dielet with 10-μm pitch interdielet I/O for runtime multiprogram reconfiguration," *IEEE J. Solid-State Circuits*, vol. 58, no. 1, pp. 111–123, Jan. 2023.

Authorized licensed use limited to: University of Michigan Library. Downloaded on February 13,2025 at 18:51:25 UTC from IEEE Xplore. Restrictions apply.

[18] D. Coltuc and P. Bolon, "Very efficient implementation of max/min filters," in *Proc. IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing (NSIP)*, Antalya, Turkey, A. E. Çetin, L. Akarun, A. Ertüzün, M. N. Gurcan, and Y. Yardimci, Eds. İstanbul, Turkey: Bogaziçi University, Jun. 1999, pp. 520–523.

[19] 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.

[20] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 2, pp. 292–308, Jun. 2019.

[21] M. Anders et al., "2.9TOPS/W reconfigurable dense/sparse matrix-multiply accelerator with unified INT8/INTI6/FP16 datapath in 14NM tri-gate CMOS," in *Proc. IEEE Symp. VLSI Circuits*, Jun. 2018, pp. 39–40.

[22] G. Desoli et al., "A 2.9TOPS/W deep convolutional neural network SoC in FD-SOI 28 nm for intelligent embedded systems," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2017, pp. 238–239.

[23] B. Bohnenstiehl et al., "KiloCore: A 32-nm 1000-processor computational array," *IEEE J. Solid-State Circuits*, vol. 52, no. 4, pp. 891–902, Apr. 2017.

[24] A. K. Yeung and J. M. Rabaey, "A 2.4 GOPS data-driven reconfigurable multiprocessor IC for DSP," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 1995, pp. 108–109.

[25] A. T. Tran, D. N. Truong, and B. M. Baas, "A complete real-time 802.11a baseband receiver implemented on an array of programmable processors," in *Proc. 42nd Asilomar Conf. Signals, Syst. Comput.*, Oct. 2008, pp. 165–170.

[26] H. Zhang et al., "A 1-V heterogeneous reconfigurable DSP IC for wireless baseband digital signal processing," *IEEE J. Solid-State Circuits*, vol. 35, no. 11, pp. 1697–1704, Nov. 2000.

[27] Z. Yu et al., "An asynchronous array of simple processors for DSP applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2006, pp. 1696–1705.

[28] Z. Yu et al., "AsAP: An asynchronous array of simple processors," *IEEE J. Solid-State Circuits*, vol. 43, no. 3, pp. 695–705, Mar. 2008.

[29] D. N. Truong et al., "A 167-processor computational platform in 65 nm CMOS," *IEEE J. Solid-State Circuits*, vol. 44, no. 4, pp. 1130–1144, Apr. 2009.

[30] Y. Lin et al., "SODA: A low-power architecture for software radio," in *Proc. 33rd Int. Symp. Comput. Archit. (ISCA)*, 2006, pp. 89–101.

[31] A. Pedram, A. Gerstlauer, and R. A. van de Geijn, "Algorithm, architecture, and floating-point unit codesign of a matrix factorization accelerator," *IEEE Trans. Comput.*, vol. 63, no. 8, pp. 1854–1867, Aug. 2014.

[32] J. Weng, S. Liu, Z. Wang, V. Dadu, and T. Nowatzki, "A hybrid systolic-dataflow architecture for inductive matrix algorithms," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2020, pp. 703–716.

[33] S. Smets, T. Goedemé, A. Mittal, and M. Verhelst, "A 978 GOPS/W flexible streaming processor for real-time image processing applications in 22 nm FDSOI," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2019, pp. 44–46.

[34] S. Pal et al., "A 7.3 m output non-zeros/J sparse matrix-matrix multiplication accelerator using memory reconfiguration in 40 nm," in *Proc. Symp. VLSI Technol.*, Jun. 2019, pp. C150–C151.

[35] D.-H. Park et al., "A 7.3 m output non-zeros/J, 11.7 m output non-zeros/GB reconfigurable sparse matrix–matrix multiplication accelerator," *IEEE J. Solid-State Circuits*, vol. 55, no. 4, pp. 933–944, Apr. 2020.

[36] M. B. Taylor et al., "The raw microprocessor: A computational fabric for software circuits and general-purpose programs," *IEEE Micro*, vol. 22, no. 2, pp. 25–35, Mar. 2002.

[37] A. M. Jones and M. Butts, "TeraOPS hardware: A new massively-parallel MIMD computing fabric IC," in *Proc. IEEE Hot Chips 18 Symp. (HCS)*, Aug. 2006, pp. 1–15.

[38] S. Kim et al., "Versa: A 36-core systolic multiprocessor with dynamically reconfigurable interconnect and memory," *IEEE J. Solid-State Circuits*, vol. 57, no. 4, pp. 986–998, Apr. 2022.

[39] K. Sankaralingam et al., "TRIPS: A polymorphous architecture for exploiting ILP, TLP, and DLP," *ACM Trans. Archit. Code Optim.*, vol. 1, no. 1, pp. 62–93, Mar. 2004, doi: 10.1145/980152.980156.

[40] K. Sankaralingam et al., "Distributed microarchitectural protocols in the TRIPS prototype processor," in *Proc. 39th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*. Washington, DC, USA: IEEE Computer Society, Dec. 2006, pp. 480–491, doi: 10.1109/MICRO.2006.19.

[41] S. Swanson et al., "The WaveScalar architecture," *ACM Trans. Comput. Syst.*, vol. 25, no. 2, pp. 1–54, May 2007. [Online]. Available: https://doi-org.proxy.lib.umich.edu/10.1145/1233307.1233308

[42] F. Yazdanpanah, C. Alvarez-Martinez, D. Jimenez-Gonzalez, and Y. Etsion, "Hybrid dataflow/von-Neumann architectures," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 6, pp. 1489–1509, Jun. 2014.

[43] V. Govindaraju, C.-H. Ho, and K. Sankaralingam, "Dynamically specialized datapaths for energy efficient computing," in *Proc. IEEE 17th Int. Symp. High Perform. Comput. Archit.*, Feb. 2011, pp. 503–514.

[44] V. Govindaraju et al., "DySER: Unifying functionality and parallelism specialization for energy-efficient computing," *IEEE Micro*, vol. 32, no. 5, pp. 38–51, Sep. 2012.

[45] T. Nowatzki, V. Gangadhar, N. Ardalani, and K. Sankaralingam, "Stream-dataflow acceleration," in *Proc. 44th Annu. Int. Symp. Comput. Archit.*, Jun. 2017, pp. 416–429.

[46] B. Mei, S. Vernalde, D. Verkest, H. D. Man, and R. Lauwereins, "ADRES: An architecture with tightly coupled VLIW processor and coarse-grained reconfigurable matrix," in *Proc. 13th Int. Conf. Field Program. Log. Appl. (FPL)*, in Lecture Notes in Computer Science, vol. 2778, Lisbon, Portugal, P. Y. K. Cheung, G. A. Constantinides, and J. T. de Sousa, Eds. New York, NY, USA: Springer, Sep. 2003, pp. 61–70, doi: 10.1007/978-3-540-45234-8_7.

[47] K. Mai, T. Paaske, N. Jayasena, R. Ho, W. J. Dally, and M. Horowitz, "Smart memories: A modular reconfigurable architecture," in *Proc. 27th Int. Symp. Comput. Archit.*, Jun. 2000, pp. 161–171.

[48] F. Liu, S. Ghosh, N. P. Johnson, and D. I. August, "CGPA: Coarse-grained pipelined accelerators," in *Proc. 51st ACM/EDAC/IEEE Design Autom. Conf. (DAC)*. New York, NY, USA: Association for Computing Machinery, Jun. 2014, pp. 1–6, doi: 10.1145/2593069.2593105.

[49] W. Lee et al., "Space-time scheduling of instruction-level parallelism on a raw machine," *ACM SIGPLAN Notices*, vol. 33, no. 11, pp. 46–57, Oct. 1998, doi: 10.1145/291006.291018.

[50] E. Mirsky and A. DeHon, "MATRIX: A reconfigurable computing architecture with configurable instruction distribution and deployable resources," in *Proc. IEEE Symp. FPGAs Custom Comput. Mach. (FPGA)*, Apr. 1996, pp. 157–166.

[51] E. Caspi, M. Chu, R. Huang, J. Yeh, J. Wawrzynek, and A. DeHon, "Stream computations organized for reconfigurable execution (SCORE)," in *Proc. 10th Int. Workshop Field-Program. Log. Appl.* Berlin, Germany: Springer-Verlag, 2000, pp. 605–614.

[52] J. Balfour, W. J. Dally, D. Black-Schaffer, V. Parikh, and J. Park, "An energy-efficient processor architecture for embedded systems," *IEEE Comput. Archit. Lett.*, vol. 7, no. 1, pp. 29–32, Jan. 2008.

[53] J. Cong, H. Huang, C. Ma, B. Xiao, and P. Zhou, "A fully pipelined and dynamically composable architecture of CGRA," in *Proc. IEEE 22nd Annu. Int. Symp. Field-Program. Custom Comput. Mach.*, May 2014, pp. 9–16.

[54] H. Singh, M.-H. Lee, G. Lu, F. J. Kurdahi, N. Bagherzadeh, and E. M. C. Filho, "MorphoSys: An integrated reconfigurable system for data-parallel and computation-intensive applications," *IEEE Trans. Comput.*, vol. 49, no. 5, pp. 465–481, May 2000.

[55] R. Nagarajan, K. Sankaralingam, D. Burger, and S. W. Keckler, "A design space evaluation of grid processor architectures," in *Proc. 34th ACM/IEEE Int. Symp. Microarchitecture (MICRO)*. Washington, DC, USA: IEEE Computer Society, Dec. 2001, pp. 40–51.

[56] C. Ebeling, D. C. Cronquist, and P. Franklin, "RaPiD—Reconfigurable pipelined datapath," in *Proc. 6th Int. Workshop Field-Program. Log., Smart Appl.* Berlin, Germany: Springer-Verlag, 1996, pp. 126–135.

[57] D. C. Cronquist, P. Franklin, S. G. Berg, and C. Ebeling, "Specifying and compiling applications for RaPiD," in *Proc. IEEE Symp. FPGAs for Custom Comput. Mach.*, Apr. 1998, pp. 116–125.

[58] T. Miyamori and K. Olukotun, "REMARC: Reconfigurable multimedia array coprocessor," in *Proc. ACM/SIGDA 6th Int. Symp. Field Program. Gate Arrays*, Monterey, CA, USA, J. Cong and S. Kaptanoglu, Eds., 1998, p. 261, doi: 10.1145/275107.275164.

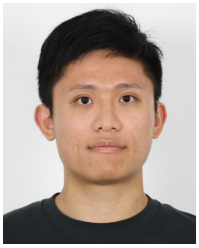
[59] M. Gao and C. Kozyrakis, "HRL: Efficient and flexible reconfigurable logic for near-data processing," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Mar. 2016, pp. 126–137.

Authorized licensed use limited to: University of Michigan Library. Downloaded on February 13,2025 at 18:51:25 UTC from IEEE Xplore. Restrictions apply.

[60] A. Carsello et al., "Amber: A 367 GOPS, 538 GOPS/W 16 nm SoC with a coarse-grained reconfigurable array for flexible acceleration of dense linear algebra," in *Proc. IEEE Symp. VLSI Technol. Circuits (VLSI Technol. Circuits)*, Jun. 2022, pp. 70–71.
[61] R. Prabhakar et al., "Plasticine: A reconfigurable accelerator for parallel patterns," *IEEE Micro*, vol. 38, no. 3, pp. 20–31, May 2018.
[62] M. B. Taylor, W. Lee, S. Amarasinghe, and A. Agarwal, "Scalar operand networks: On-chip interconnect for ilp in partitioned architectures," in *Proc. The 9th Int. Symp. High-Perform. Comput. Archit. (HPCA)*, 2003, pp. 341–353.
[63] R. D. Wittig and P. Chow, "OneChip: An FPGA processor with reconfigurable logic," in *Proc. IEEE Symp. FPGAs Custom Comput. Mach. (FPGA)*, Apr. 1996, pp. 126–135.
[64] E. A. Lee and D. G. Messerschmitt, "Synchronous data flow," *Proc. IEEE*, vol. 75, no. 9, pp. 1235–1245, Sep. 1987.
[65] J. B. Dennis and D. P. Misunas, "A preliminary architecture for a basic data-flow processor," *ACM SIGARCH Comput. Archit. News*, vol. 3, no. 4, pp. 126–132, Dec. 1974, doi: 10.1145/641675.642111.
[66] K. Arvind and R. S. Nikhil, "Executing a program on the MIT tagged-token dataflow architecture," *IEEE Trans. Comput.*, vol. 39, no. 3, pp. 300–318, Mar. 1990.
[67] G. Bilsen, M. Engels, R. Lauwereins, and J. A. Peperstraete, "Cyclo-static data flow," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, vol. 5, 1995, pp. 3255–3258.
[68] T. Nowatzki, V. Gangadhar, and K. Sankaralingam, "Exploring the potential of heterogeneous von neumann/dataflow execution models," *ACM SIGARCH Comput. Archit. News*, vol. 43, no. 3S, pp. 298–310, Jun. 2015, doi: 10.1145/2872887.2750380.

Kuan-Yu Chen (Graduate Student Member, IEEE) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2018, and the M.S.E. and Ph.D. degrees in electrical and computer engineering from the University of Michigan, Ann Arbor, MI, USA, in 2020 and 2024, respectively.

His research interests include digital circuit design, domain-specific programmable accelerators, and computer architecture.



Chi-Sheng Yang received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2018, and the M.S. degree in electrical and computer engineering from the University of Michigan, Ann Arbor, MI, USA, in 2020.

His research interests include VLSI design and computer architecture, especially in high-performance and high-efficiency accelerator design.

Yu-Hsiu Sun received the bachelor's degree from National Taiwan University, Taipei, Taiwan, in 2018, and the master's degree from the University of Michigan, Ann Arbor, MI, USA, in 2020.

His research focused on high-speed ASIC to ASIC/FPGA interface while working under Prof. Hun-Seok Kim in the University of Michigan. He then started his career at Qualcomm from 2021 developing 5G/LTE ASIC.

Chien-Wei Tseng (Graduate Student Member, IEEE) received the B.S. and M.S. degrees in electronics engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2011 and 2014, respectively, and the Ph.D. degree in electrical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2024.

His research interests include high-frequency analog/RF circuit design, clocking circuit design, and digital signal processing.

Morteza Fayazi (Graduate Student Member, IEEE) was born in Tehran, Iran, in 1994. He received the B.Sc. degree major in electrical engineering and minor in computer science from the Sharif University of Technology (SUT), Tehran, in 2017, and the M.S. degree in electrical engineering and computer science from the University of Michigan, Ann Arbor, MI, USA, in 2020, where he is currently pursuing the Ph.D. degree.

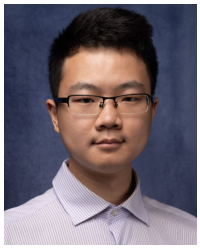
His research interests include circuit design automation and machine learning.

Xin He received the B.Eng. degree in software engineering from Sichuan University, Chengdu, China, in 2011, and the Ph.D. degree in computer science from the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing, China, in 2017.

He was an Assistant Research Scientist at the University of Michigan, Ann Arbor, MI, USA, until 2022. His research interests include computer architecture especially on application-specific acceleration, deep learning, approximate computing, and interconnection networks.

Siying Feng received the Ph.D. degree in computer engineering from the University of Michigan, Ann Arbor, Austin, MI, USA, in 2022, under the supervision of Prof. R. Dreslinski.

She is currently working as a Senior Engineer with Arm. Inc. Her research interests focused on application-specific and reconfigurable architecture design for workloads with irregular data access patterns.



Yufan Yue (Graduate Student Member, IEEE) received the dual B.S. degree from Shanghai Jiao Tong University, Shanghai, China, and the University of Michigan, Ann Arbor, MI, USA, in 2020. He is currently pursuing the Ph.D. degree in electrical and computer engineering from the University of Michigan.

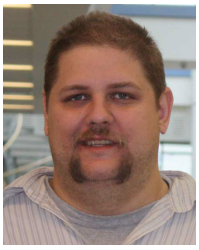
He was an Intern with Qualcomm Technologies Inc., San Diego, CA, USA, in Summer 2024, where he worked on 6G FEC Decoder Design. His research interest involves the design of reconfigurable multi-mode forward error correction accelerators and domain-specific hardware architecture, especially for wireless communication.

Authorized licensed use limited to: University of Michigan Library. Downloaded on February 13,2025 at 18:51:25 UTC from IEEE Xplore. Restrictions apply.

Trevor Mudge (Life Fellow, IEEE) received the Ph.D. degree in computer science from the University of Illinois at Urbana–Champaign, Champaign, IL, USA.

He is the Bredt Family Professor of computer science and engineering at the University of Michigan, Ann Arbor, MI, USA. He is the author of numerous articles on computer architecture, programming languages, VLSI design, and computer vision. He has chaired 59 Ph.D. theses in these areas.

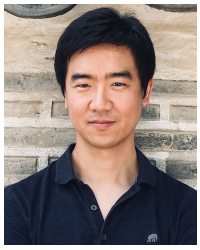
Dr. Mudge is a fellow of the ACM and a member of the IET and the British Computer Society. In 2014, he received the ACM/IEEE CS Eckert–Mauchly Award and the University of Illinois Distinguished Alumni Award.



Ronald Dreslinski (Senior Member, IEEE) received the B.S.E., M.S.E., and Ph.D. degrees from the University of Michigan, Ann Arbor, MI, USA, in 2001, 2003, and 2011, respectively.

He is currently an Associate Professor of computer science and engineering at the University of Michigan. His work focuses on hardware and circuit designs for a post-Moore's Law world.

Dr. Dreslinski received the 2015 IEEE Young Computer Architect Award.



Hun-Seok Kim (Senior Member, IEEE) received the B.S. degree in electrical engineering from Seoul National University, Seoul, South Korea, in 2001, and the Ph.D. degree in electrical engineering from the University of California at Los Angeles, Los Angeles, CA, USA, in 2010.

He is currently an Associate Professor with the University of Michigan, Ann Arbor, MI, USA. His research interests focus on system analysis, novel algorithms, and VLSI architectures for low-power/high-performance wireless communications, signal processing, computer vision, and machine learning systems.

Dr. Kim was a recipient of the DARPA Young Faculty Award in 2018 and the National Science Foundation (NSF) CAREER Award in 2019. He has served as an Associate Editor for IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, and IEEE SOLID STATE CIRCUITS LETTERS.

David Blaauw (Fellow, IEEE) received the B.S. degree in physics and computer science from Duke University, Durham, NC, USA, in 1986, and the Ph.D. degree in computer science from the University of Illinois at Urbana–Champaign, Champaign, IL, USA, in 1991.

Until August 2001, he worked for Motorola, Inc., Austin, TX, USA, and since August 2001, he has been at the faculty of the University of Michigan, Ann Arbor, MI, USA, where he is the Kensall D. Wise Collegiate Professor of EECS. He has published more than 600 articles, has received numerous best papers, and holds 65 patents. His research has covered ultralow-power digital and analog circuits for mm-sensors, hardware for neural networks in edge devices, and high-performance accelerators.

Dr. Blaauw has served on the IEEE International Solid-State Circuits Conference's technical program committee and received the 2016 SIA-SRC faculty award for lifetime research contributions to U.S. semiconductor industry.

Authorized licensed use limited to: University of Michigan Library. Downloaded on February 13,2025 at 18:51:25 UTC from IEEE Xplore. Restrictions apply.