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

Kuan-Yu Chen, Chi-Sheng Yang, Yu-Hsiu Sun, Chien-Wei Tseng, Morteza Fayazi, Xin He, Siying Feng,
Yufan Yue, Trevor Mudge, Ronald Dreslinski, Hun-Seok Kim, David Blaauw
University of Michigan, Ann Arbor, USA. knyuchen@umich.edu

2022 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits) | 978-1-6654-9772-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/VLSITechnologyandCir46769.2022.9830330

Abstract: We present DAP (Domain Adaptive Processor), an adaptive systolic-array-processor of 256 programmable cores in 12 nm CMOS for wireless communication workloads. DAP uses a globally homogeneous but locally heterogeneous architecture, decode-less reconfiguration instructions for data streaming, single-cycle data communication between functional units (FUs), and lightweight nested-loop control. We show how configuration flexibility and fast program loading allows a wide range of communication workloads to be mapped and swapped in sub-µs, supporting continually evolving communication standards such as 5G. DAP achieves 507 GMACs/J and a peak performance of 264 GMACs.

Introduction: With the increased use of accelerators to augment general purpose and GPGPU processing, the trade-off between efficiency and flexibility has become a key concern. The need for flexibility is particularly pertinent for wireless communication workloads where new standards are frequently introduced and require modification of computational kernels [1]. These workloads are characterized by data streaming which make them especially suitable for systolic-array architectures which can achieve high efficiency but traditionally have limited flexibility.

To address this issue, we propose DAP that implements a configurable systolic-array fabric designed to execute a wide range of wireless communication kernels with near-ASIC energy efficiency. Four types of Processing Elements (PEs), each with both complex and real number arithmetic FUs, are organized in locally heterogeneous clusters, with clusters replicated homogeneously across the full 256-PE fabric. Decode-less FU activation and routing reconfiguration instructions and a lightweight nested-loop program control minimize overhead and reduce the needed instruction memory to a mere 128-entries per PE. A light-weight single-port Register File Unit (RU) with dual crossbars enables efficient direct FU-to-FU streaming (zero cycle inter-FU latency) minimizing data-movement and buffering energy.

Architecture Overview: The top-level architecture (Fig. 1) consists of the systolic array of PEs, Management Units (MUs) for data movement into and out of the array, and 32 banks of 8kB scratchpad (256KB total) connected to an AXI bus for the external host interface. There are 4 types of PEs that perform different operations: complex multiply and accumulate (MAC), intelligent storage (IS), CORDIC and division (COR-DIV) and LOGICAL. The PEs support 32-bit fixed point complex numbers and communicate with their neighbors in all 8 directions and can be individually clock gated. Each PE has 4 states, LOAD for loading program into instruction memory (IMEM), EXECUTE for program execution, ROUTE for acting as a programmable router, and IDLE.

PE Architecture: Communication kernels rely 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. Hence, we eliminated traditional cycle upon cycle instruction fetch and decode and instead utilize *stream instructions* that simultaneously specifies the data flow configuration (crossbars and queues) and the operation of FUs (which are enabled or not) combined with loop control which specifies how many cycles the configuration stays in place (Fig 2). Hence, a single instruction can stay in place for many cycles, greatly minimizing control overhead and code size. Further, data is streamed from one FU, through a storage registers or queue, to another FU in a single cycle. Streaming data from a FU in one PE to a FU in neighbor PE takes two cycles, greatly improving PE to PE communication efficiency compared to network-on-chip architectures which must decode address-headers and execute routing algorithms. Since instruction fields are directly copied into the control registers, instruction decode overhead is essentially eliminated and since embedded loop control greatly reduces program size, an entire DAP can be programmed in 100's of cycles (sub-µs), allowing on-the-fly kernel swapping (unlike FPGAs), allowing simultaneous support of multiple protocols that share PEs in a time multiplexed fashion.

Implementation of nested loops use a loop control unit and 2-bit loop field (LC) in each instruction. Two shift registers record the return address and the remaining number of loop iterations. Both shift right when entering a new loop and shift left when the loop number reaches zero. We implemented 3 levels of shift registers, allowing for 3 nested loops which was found sufficient for a wide range of communication kernels.

A conventional CPU or GPGPU typically implements a large number of FUs and registers resulting a large, multi-port register file. In our design with 4 FUs per PE, this would result in a register file with over 12 ports, which would dominate the power and area consumption of the PE. Instead, we restrict the data movement based on common patterns for a wide set of kernels, and handle connections between the FUs with two sequential crossbars (Fig. 3, left) that are pre-set with the stream instructions. Each FU's inputs are directly connected to specific registers (or small queues with 4 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 which allows direct FU-to-FU streaming. In addition, data can be routed to move registers or a global FIFO (4 entry, Fig. 3, left) through a smaller 16-to-4 crossbar to allow additional data storage, data alignment, and broadcast to a selectable set of FU inputs.

The IS PE has a 2kB storage to address longer buffering, necessary in certain kernels such as FFT. The COR-DIV PE performs vector rotate, magnitude/angle computation, divide, and square root operations. There are two CMAC FU inside each MAC PE (Fig. 3, left) each of which can be reconfigured as either 1 complex-number MAC or 4 real-number MACs (Fig. 3, right). This feature greatly benefits real-valued kernels by providing 4x the number of MAC units (8 total for the MAC PE). Multipliers in the CMAC FU occupy two pipeline stages and adders a single stage. However, the multipliers continue to set the clock frequency leaving adders with significant delay slack. We utilize this slack, by adding 2 operation-fused adders (red box, Fig 3, right) which execute in a single cycle with the CMAC adders, providing additional computation without impacting clock frequency. The global scratchpads outside of the PE array and in the IS PEs can be configured into either complex or real number mode by multi-banking.

DAP supports multitasking in which the PE array runs different kernels simultaneously. A dataflow graph of the desired kernels is first broken down into DAP-supported FUs, and connections are then programmed into the intra- and inter-PE datapaths. Different kernels in the same workload can directly interface with each other without moving data in and out of the global scratchpad. Fig 4-6 shows example mappings of 2D convolution, OFDM, and MMSE MIMO detection. There are multiple phases in OFDM; packet detection, channel estimation, and demodulation. For packet detection, the output of FIR is directly used as the autocorrelation input without leaving the PE array. Upon packet detection, DAP can be reprogrammed within 0.5us to reuse the same PEs for channel estimation, FFT, and symbol demodulation (Fig. 5). DAP's ability to merge and seamlessly connect different kernels largely eliminates scratchpad access overhead. MIMO detection is combination of multiple kernels including QR decomposition, matrix multiplication, and back substitution. After computing the MMSE matrix and PEs are reprogramed (0.2us latency) to perform matrix vector multiplication. In this mapping, no IS PE is required since an entire matrix is stored in the queues connected to the CMAC.

DAP is fabricated in 12nm CMOS and occupies 21mm^2. A total of 17 kernels were mapped to DAP of which 7 more common ones are shown in Fig. 7 for space reasons. In addition, the performance and efficiency of an artificial kernel that maximizes PE utilization

are shown in Fig. 8. Fig. 9-10 compares DAP to both fixed-function accelerators and programmable processors. Technology scaling at iso-throughput was done based on SPICE simulations of FO4 energy and delay. 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 highest peak performance of 264 GMACs at 1.0 V, 506 MHz.

Acknowledgements: This work was sponsored in part by the U.S. Government under the DARPA DSSoC program, award #FA8650-18-2-7860.

References:
[1] Fang-Li Yuan *et al*., VLSI 2014. [2] J. P. Cerqueira *et al*., VLSI 2019.
[3] G. Desoli *et al*., ISSCC 2017. [4] M. Anders *et al*., VLSI 2018.

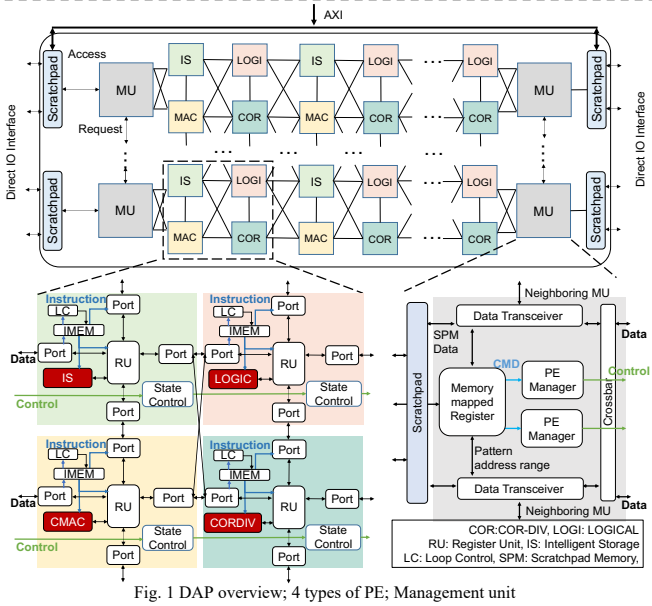


Fig. 1 DAP overview; 4 types of PE; Management unit

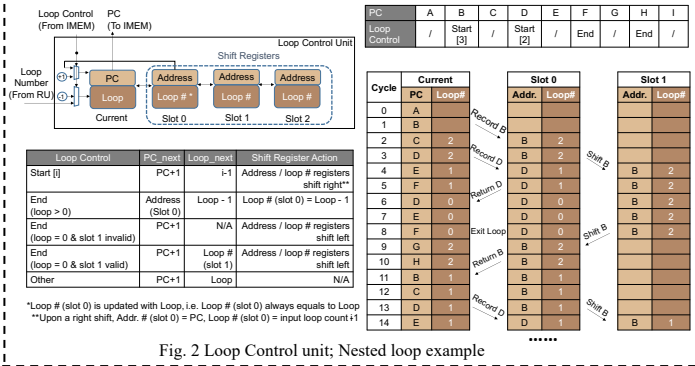


Fig. 2 Loop Control unit; Nested loop example

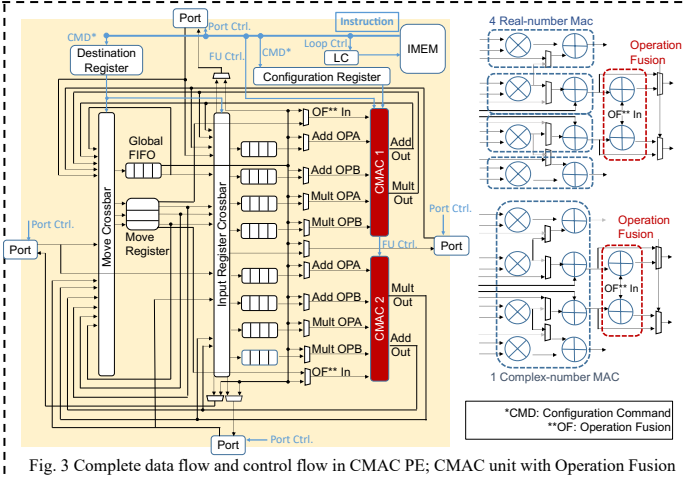


Fig. 3 Complete data flow and control flow in CMAC PE; CMAC unit with Operation Fusion

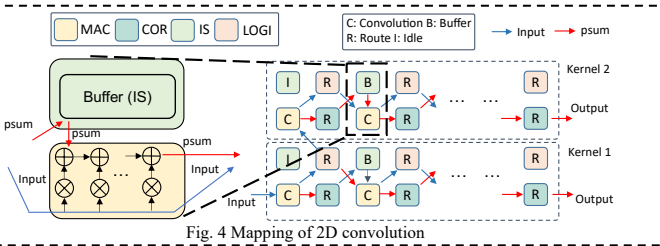


Fig. 4 Mapping of 2D convolution

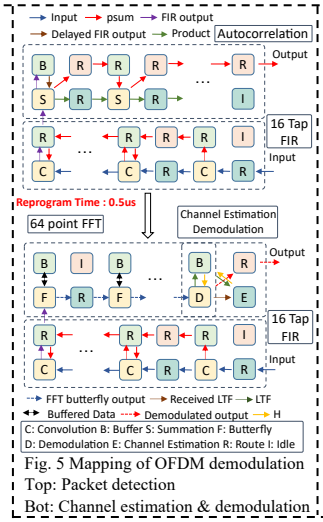


Fig. 5 Mapping of OFDM demodulation
Top: Packet detection
Bot: Channel estimation & demodulation

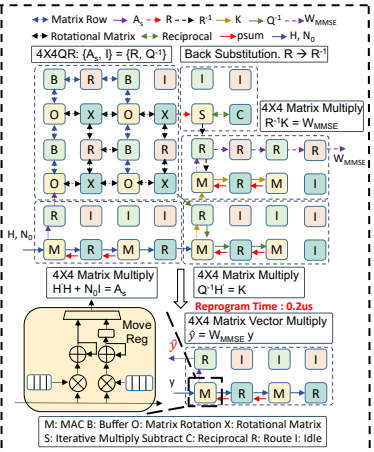


Fig. 6 Mapping of MIMO detection
Top: Calculating W_{MMSE}
Bot: Matrix Vector Multiplication

		Peak Performance		Efficiency / Throughput Trade-off		Number of Required PEs*
Kernel / Workload		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

Fig. 7 Measurement results of kernels / workloads *Routing PEs excluded

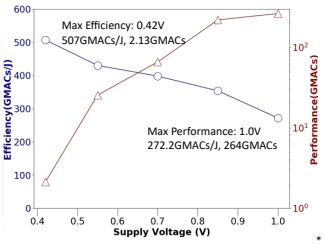


Fig. 8 DAP efficiency and performance at different supply voltage levels

	This Work	VLSI'14[1]*	VLSI'19[2]***
Technology	12 nm	40 nm	65 nm
Total Cores	256	16	16
Data memory per core (KB)	1.47	None	0.5
Frequency Range (MHz)	8.55-506	25-500	0.1-20****
FIR (GMACs/J)	409, 0.65V	467**, 0.73V	26, 0.54V
4X4 QR (nJ/Matrix)	8.62, 0.65V	6.05**, 0.49V	N/A
256 Point FFT (nJ/FFT)	31.6, 0.65V	N/A	1855, 0.54V
Number of Kernels	17	5	5

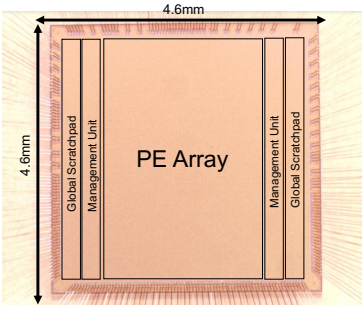
* includes only compute cores, interface and peripheral circuits
** Calculated from plot and scaled to iso-throughput and for technology
*** Throughput cannot match with this work, compare with max throughput reported. **** estimated based on plots. All bitwidth are 16 bit.

Fig. 9 Comparison with programmable architectures

	2D Convolution		GeMM	
	This Work	ISSCC'17[3]*	This Work	VLSI'18[4]**
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	2.16x		2.23x	

All bitwidth are 16 bit. * Listed numbers are scaled to iso-throughput and technology
** Throughput cannot match with this work, compare with max throughput reported

Fig. 10 Comparison with dedicated accelerators



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

Fig. 11 Chip micrograph and summary