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/VLSITechnologyandCir46769.2022.9830509

Amber: A 367 GOPS, 538 GOPS/W 16nm SoC with a Coarse-Grained Reconfigurable Array for Flexible Acceleration of Dense Linear Algebra

Alex Carsello*†, Kathleen Feng*, Taeyoung Kong, Kalhan Koul, Qiaoyi Liu, Jackson Melchert, Gedeon Nyengele, Maxwell Strange, Keyi Zhang, Ankita Nayak, Jeff Setter, James Thomas, Kavya Sreedhar, Po-Han Chen, Nikhil Bhagdikar, Zachary Myers, Brandon D'Agostino, Pranil Joshi, Stephen Richardson, Rick Bahr, Christopher Torng, Mark Horowitz, Priyanka Raina

Stanford University, CA, USA; *Equal Contribution; †Email: ajcars@stanford.edu

Abstract

Amber is a system-on-chip (SoC) with a coarse-grained reconfigurable array (CGRA) for acceleration of dense linear algebra applications such as machine learning (ML), image processing, and computer vision. It achieves a peak energy efficiency of 538.0 INT16 GOPS/W and 483.3 BFloat16 GFLOPS/W. We maximize CGRA utilization and minimize reconfigurability overhead through (1) dynamic partial reconfiguration of the CGRA that enables higher resource utilization by allowing multiple applications to run at once, (2) efficient streaming memory controllers supporting affine access patterns, and (3) low-overhead transcendental and complex arithmetic operations. Compared to a CPU, a GPU, and an FPGA, Amber achieves up to 3902x, 152x, and 88x better energy-delay product (EDP).

Introduction

Imaging, vision, and ML applications have used dedicated hardware accelerators to increase performance and energy efficiency. However, application changes quickly result in outdated accelerators. Reconfigurable accelerators adapt to these changes but are difficult to compile to and have high area and energy overheads due to (1) slow reconfiguration for repurposing idle resources, (2) inefficient memory control logic, and (3) underutilized, costly compute units. Amber is a reconfigurable SoC that overcomes these challenges with (1) fast dynamic partial reconfiguration that runs up to eight kernels in parallel, (2) efficient streaming memories for affine patterns, and (3) low-overhead complex arithmetic support.

Amber has a CGRA, a global buffer (GLB), and an ARM Cortex M3 CPU (Fig. 1). The CGRA has 384 INT16/ BFloat16 processing elements (PEs), each with a 64B register file (RF), and 128 memories (MEMs), each with 4KB SRAM. Tiles communicate through statically configured 16- and 1-bit interconnects. The GLB streams data and bitstreams to the CGRA and has 16 tiles, each with two 128KB SRAMs and load/store units. The CPU manages application execution.

Dynamic Partial Reconfiguration of the CGRA

Imaging, vision, and ML applications have multiple tasks which may not all be active at a time; for example, every frame in an image stream is processed by a camera pipeline (CP), while only key frames go through ResNet-18 for object detection. Thus, resources used for ResNet will sit idle (Fig. 4a). Idle resources can be repurposed at runtime, but configuration interfaces, such as AXI-Lite and those in FPGAs, switch applications slowly. Amber demonstrates the first CGRA configuration network specialized for high performance dynamic partial reconfiguration (DPR), capable of configuring the full array in 3.5μs and supporting up to eight distinct kernels simultaneously. For low overhead, high-bandwidth configuration, Amber shares GLB storage between application and configuration data and uses the GLB's parallel data ports to drive a high-speed, pipelined configuration network (+1.46% area overhead) (Fig. 2). DPR provides a 662x speedup (7280M configs/s vs. 11M configs/s) and 687x lower configuration energy (57.4 pJ/config vs. 39454.5 pJ/config) over AXI-Lite. Amber DPR has 36.4x higher peak throughput compared to DPR on an FPGA (Fig. 3). DPR enables higher utilization of the CGRA whereas the eFPGA SoC from [1] sees its efficiency limited by utilization. Allocating all resources to CP during non-key frames (Fig. 4) leads to a 1.77x speedup and 2.51x lower EDP (Fig. 5).

On-Chip Streaming Memories for Affine Patterns

Direct memory access engines are commonly used to stream data to the accelerator but are too general and have high area and energy overheads. Since dense linear algebra applications demonstrate affine access patterns, we specialize the memory controllers to match these patterns, while still maintaining flexibility. All on-chip memories (GLB, MEMs, RFs) have optimized internal affine memory controllers, whose parameters are extracted from the application by our compiler (Fig. 6). Each controller has an iteration domain (ID) to specify the range of memory operations, an address generator (AG) to compute affine addresses from a set of strides and ID values, and a schedule generator (SG) to similarly produce read/write enables. The memory controllers (1) fetch data from a wide-fetch SRAM, which has lower access energy per byte (0.81 pJ vs. 1.65 pJ for single-fetch SRAM), (2) eliminate multipliers by using a recurrence relation to compute affine patterns, and (3) share ID/AG/SGs to reduce area. Internal memory controllers save 32.3% area and 25.0% power vs. using PEs for address generation and eliminate sending memory requests across the interconnect as seen in [2]. A wide-fetch SRAM (512 64-bit vs. 2048 16-bit words) further reduces area by 26.1% and power by 30.6%, including the extra logic to support wide-fetch (Fig. 7).

Low-Overhead Complex Arithmetic Operations

Applications sometimes require, but infrequently use, complex arithmetic operations like BFloat16 division, log, sine, and exponential. For example, only 15% of the operations in non-local means (NLM) are complex. Using a CPU with floating point support to perform these operations results in a significant performance penalty (29.7x slower for NLM). Instead of supporting these operations entirely within one PE, the Amber PE uniquely contains supplementary operations (+0.2% area overhead) that implement complex operations by combining multiple tiles, saving 37.8% in area (Fig. 8). For example, division is equivalent to multiplying, using one PE, the dividend by the reciprocal of the divisor. The reciprocal is found using one PE to get the mantissa, one MEM to lookup the reciprocal of the mantissa, and one PE to subtract the original exponent from the reciprocal of the mantissa's exponent. A CGRA with dedicated complex operations in each PE has 39% lower total area utilization for NLM compared to Amber but 45% higher on average for applications that do not have complex operations (Fig. 9). Amber is 2.3x more energy efficient compared to [3], which uses longer latency functional units for complex operations.

Results

We designed an end-to-end compiler to map applications written in the high-level Halide language [4] onto Amber (Fig. 10). Due to the coarser granularity, our compiler is 12.4x faster than Vivado FPGA compiler (1.2 vs. 14.9 minutes for FPGA). Amber is fabricated in TSMC 16nm FinFET technology. It operates at 0.84-1.29V and up to 955 MHz (Fig. 11). It achieves a peak energy efficiency of 538.0 INT16 GOPS/W and 483.3 BFloat16 GFLOPS/W at 580 MHz/0.84V and 560 MHz/0.86V, respectively, 1.7x better than eFPGA and with 36.7x throughput [1] (Fig. 12). Amber achieves

Authorized licensed use limited to: Stanford University. Downloaded on November 18,2026 at 04:30:13 UTC from IEEE Xplore. Restrictions apply.

132-864x, 559-3902x, 292-988x, 12-152x, and 18-88x better EDP vs. Cortex A57, 1-core and 12-core Xeon CPUs, Tesla K40 GPU, and Virtex Ultrascale+ FPGA, respectively (Fig. 13). For ResNet-18 layers, Amber uses up to 9.4x less energy and has up to 10.3x lower EDP than the FPGA.

Conclusion

Amber is a reconfigurable SoC for flexible and efficient acceleration of imaging, vision, and ML applications. The demonstrated techniques – dynamic partial reconfiguration, efficient streaming memories, and low-overhead complex arithmetic – achieve efficient *domain* (rather than single application) *acceleration*.

Acknowledgement: DARPA DSSoC, AHA Center, Stanford SystemX Alliance.

References: [1] P. N. Whatmough *et al*., VLSI'19. [2] A. Rovinski *et al*., VLSI'19. [3] C. Schmidt *et al*., ISSCC'21. [4] R. Mullapudi *et al*., ACM TOG'16.

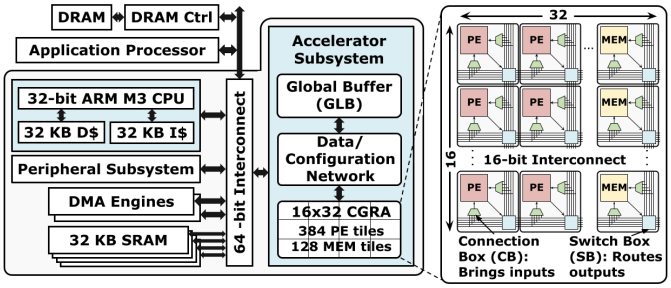


Fig. 1: Amber SoC with a CGRA having 384 PE and 128 MEM tiles.

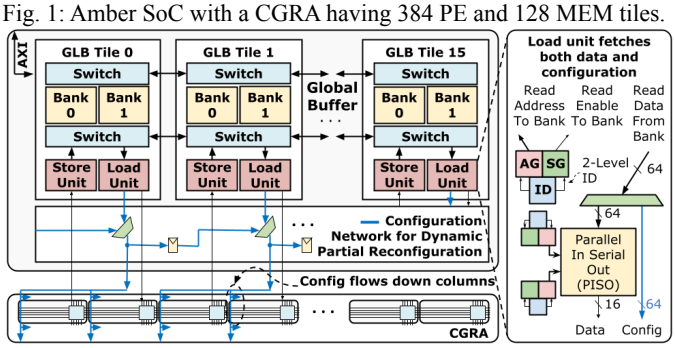


Fig. 2: Dynamic partial reconfiguration (DPR) support in the GLB. AG/SG/ID generate memory addresses and control signals (Fig. 6).

	Max. Frequency	Interface Bitwidth	Peak Throughput
CGRA DPR	520MHz	448bit	29.1GB/s
CGRA AXI-Lite	660MHz	32bit	44MB/s
FPGA (Xilinx ICAP)	200MHz	32bit	800MB/s

Fig. 3: Peak throughput of CGRA DPR is up to 36.4x that of FPGA.

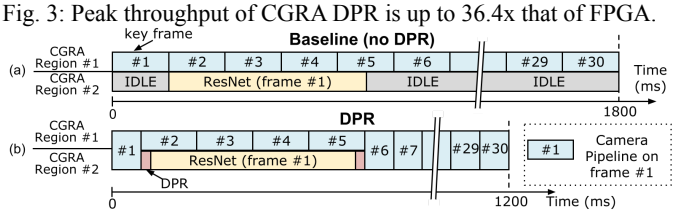


Fig. 4: Execution timeline of running camera pipeline and ResNet on CGRA (a) without DPR and (b) with DPR.

For Camera Pipe+ResNet	Frequency (MHz)	Execution Time (ms/30 frames)	Energy (J/30 frames)	EDP (J·s/30 frames)
Baseline	200	902	217	196
DPR	200	511 (-43.3%)	154 (-29.0%)	78 (-60.2%)

Fig. 5: Energy/performance benefits of DPR vs. baseline (no DPR).

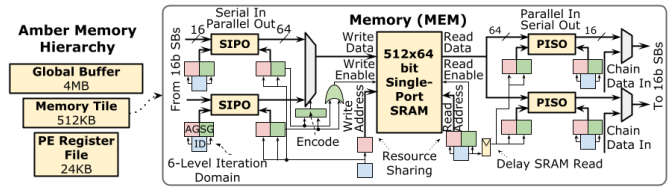


Fig. 6: Streaming memory hierarchy and implementation.

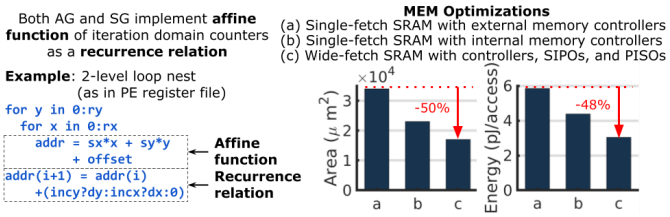


Fig. 7: Memory controller access patterns and area/energy benefits.

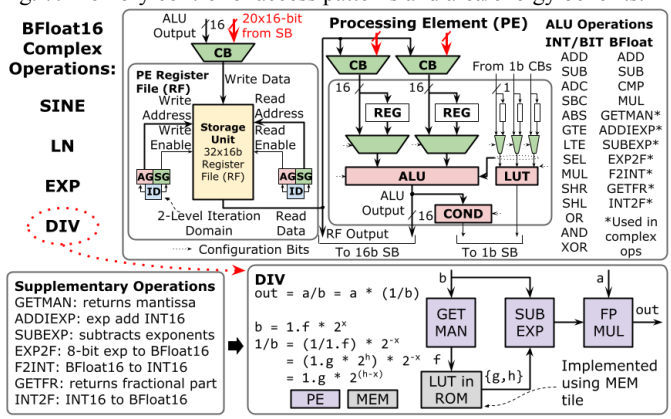


Fig. 8: Low-overhead, distributed complex arithmetic operations.

	A: Amber CGRA		B: CGRA w/ dedicated complex ops in PE							
	BLUR		UNSHARP		CAMERA		HARRIS		NLM	
	A	B	A	B	A	B	A	B	A	B
# PE tiles	280		303		240		194		235	106
# MEM tiles	14		37		39		17		25	6
Used mm²	2.62	4.00	3.36	4.86	2.90	4.08	1.99	2.95	2.52	1.53
% Change	+47%		+45%		+41%		+48%		-39%	

Fig. 9: Area utilization tradeoffs for complex arithmetic support.

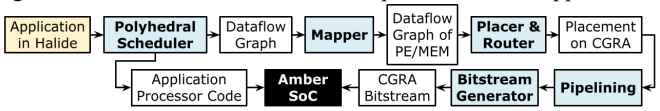


Fig. 10: Automatic end-to-end flow for compiling applications.

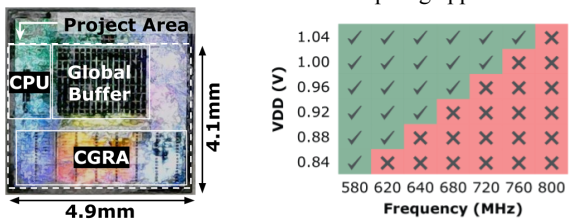
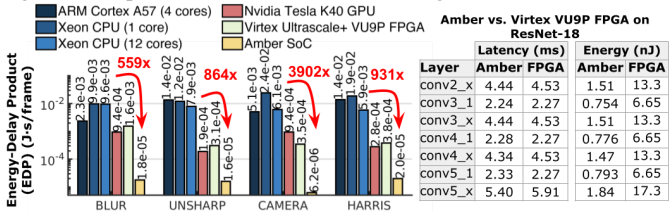


Fig. 11: Die photo (left) and shmoo plot (right).

	This Work	Whatmough VLSI'19	Schmidt ISSCC'19	Rovinski VLSI'19
Architecture	SoC with CGRA	SoC with FPGA	Multicore Vector CPU	Manycore CPU with Mesh NoC
Technology	16 nm FinFET	16 nm FinFET	16 nm FinFET	16 nm FinFET
Die Area (mm²)	20.1	25	24.01	15.25
Numeric Formats Supported	BFloat16, INT16: CGRA; INT16/32/64: ARM-M3	FP16/32/64, INT32/64: A53; INT22: eFPGA; INT16: CCA	FP16/32/64, INT32/64	INT32
# Cores	384 PEs, 1 ARM-M3	2 ARM-A53, 1 ARM-M0, 8.76 K 6-in LUTs, 80 DSPs	8 application cores, 1 management core	496 cores
Total SRAM (MiB)	4.5	9.025	4.5	3.875
Voltage (V)	0.84 @ 580 MHz 1.29 @ 955 MHz	0.5 - 1.0	0.55 @ 339 MHz 1.0 @ 1.44 GHz	0.60 @ 500 MHz 0.98 @ 1.4 GHz
Frequency (MHz)	955	> 1000	1440	10 - 1400
Peak Performance	367 INT16 GOPS 342 BFloat16 GFLOPS	~56.2 GOPS @ 79.4 GOPS/W 10 GOPS @ 312.4 GOPS/W	368.4 FP16 GFLOPS	695 INT32 GOPS
Peak Energy Efficiency	538.0 INT16 GOPS/W 483.3 BFloat16 GFLOPS/W	312.4 GOPS/W on eFPGA	209.5 FP16 GFLOPS/W	93.04 INT32 GOPS/W

Fig. 12: Comparison vs. state-of-the-art configurable accelerators.



Amber vs. Virtex VU9P FPGA on ResNet-18

	Latency (ms)		Energy (nJ)	
Layer	Amber	FPGA	Amber	FPGA
conv2_x	4.44	4.53	1.51	13.3
conv3_1	2.24	2.27	0.754	6.65
conv3_x	4.44	4.53	1.51	13.3
conv4_1	2.28	2.27	0.776	6.65
conv4_x	4.34	4.53	1.47	13.3
conv5_1	2.33	2.27	0.793	6.65
conv5_x	5.40	5.91	1.84	17.3

Fig. 13: Measured energy-delay product compared to CPU, GPU, and FPGA. ResNet-18 comparison with FPGA, with the same compute parallelization and compute-bound schedules.

Authorized licensed use limited to: Stanford University. Downloaded on November 18,2026 at 04:30:13 UTC from IEEE Xplore. Restrictions apply.