Amber: Coarse-Grained Reconfigurable Array-Based SoC for Dense Linear Algebra Acceleration

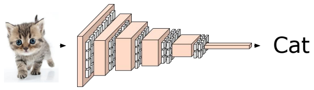
Kathleen Feng, Alex Carsello, Taeyoung Kong, Kalhan Koul, Qiaoyi Liu, Jackson Melchert, Gedeon Nyengele, Maxwell Strange, Keyi Zhang, Ankita Nayak, Jeff Setter, James Thomas, Kavya Sreedhar, Po-Han Chen, Nikhil Bhagdikar, Zachary Myers, Brandon D'Agostino, Pranil Joshi, Stephen Richardson, Rick Bahr,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

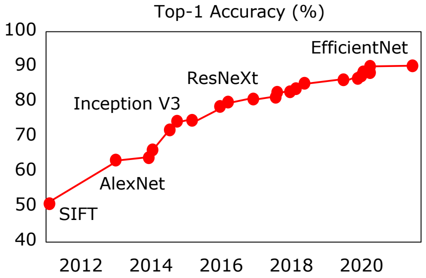
Stanford University

Hot Chips 34
August 22, 2022

Application-Specific Accelerators

- Dedicated hardware accelerators popular for imaging, vision, and machine learning (ML) applications
- Applications change rapidly → reconfigurable accelerators





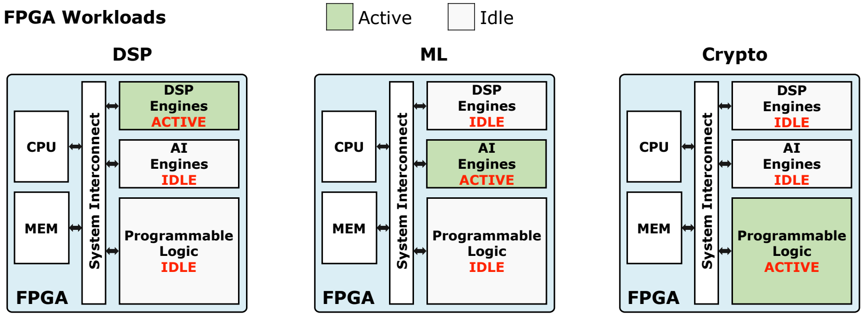
Accelerator optimized for ResNeXt not efficient for EfficientNet due to different layer types

↓

New ASICs required constantly to keep up with state of the art

Reconfigurable Accelerator Overheads

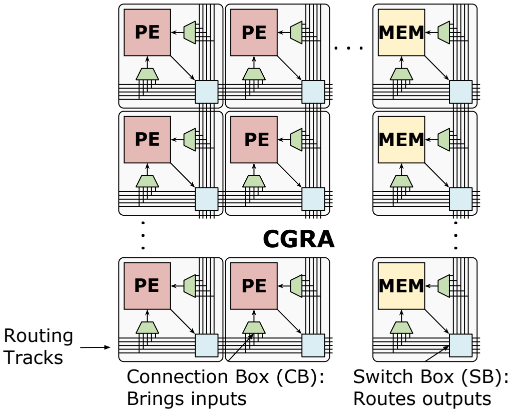
- Slow reconfiguration for repurposing idle resources
- Inefficient memory control logic
- Underutilized, costly compute units



Amber Architecture

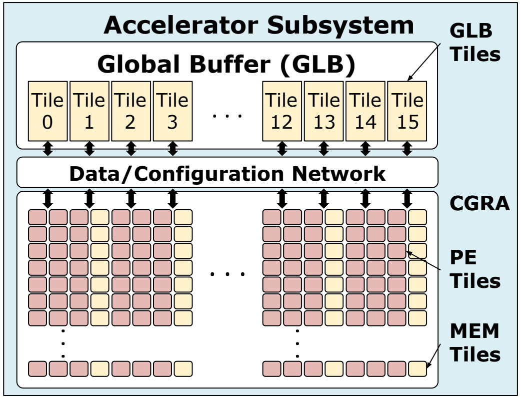
Coarse-grained configurable array (CGRA) for acceleration

- 384 processing elements (PEs): supports INT16/BFloat16 operations, 64B register file
- 128 memory elements (MEMs): 4KB SRAM with internal streaming memory controllers

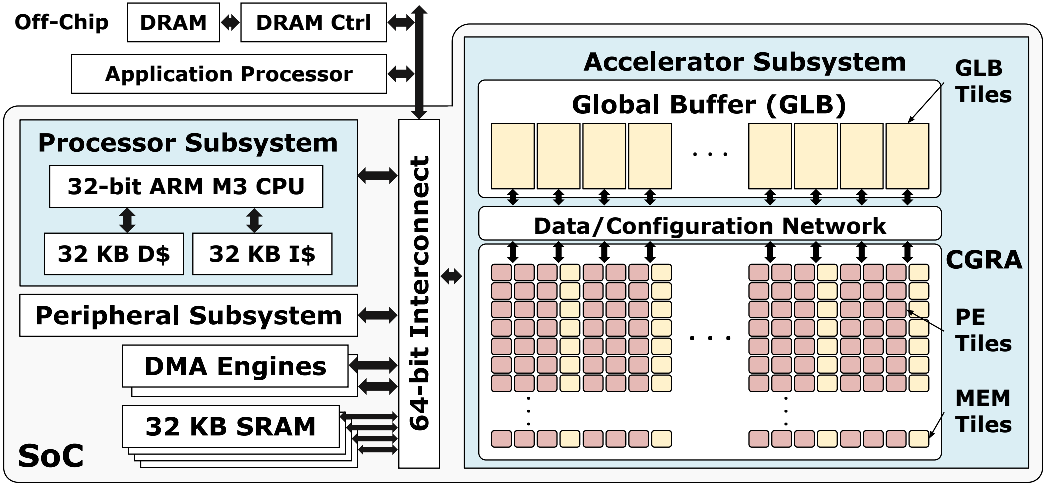

PE
PE
MEM
PE
PE
MEM
CGRA
PE
PE
MEM
Routing Tracks
Connection Box (CB): Brings inputs
Switch Box (SB): Routes outputs

Amber Architecture

- Each column has same title type
  - Every fourth column is memory
- Global buffer (GLB): streams data and bitstreams to the CGRA
  - 16 tiles: each with two 128KB SRAM banks, load and store units



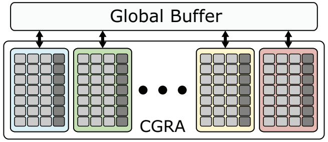
Amber Architecture



Contributions

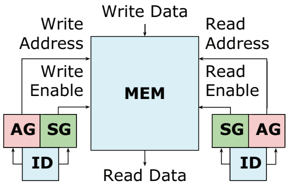
Configuration

Fast dynamic partial reconfiguration that runs up to eight kernels in parallel



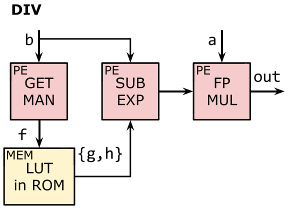
Memory

Efficient streaming memories for affine patterns



Compute

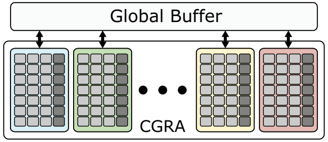
Low-overhead complex arithmetic support



Contributions

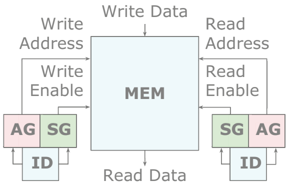
Configuration

Fast dynamic partial reconfiguration that runs up to eight kernels in parallel



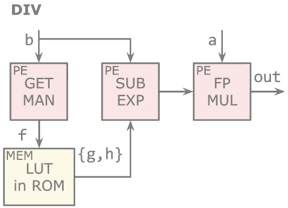
Memory

Efficient streaming memories for affine patterns



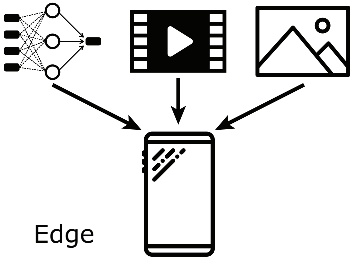
Compute

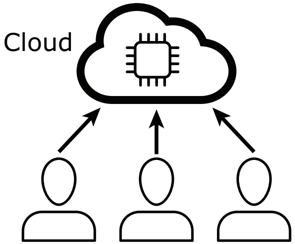
Low-overhead complex arithmetic support



Maximizing Resource Utilization

- Reconfigurable accelerators frequently need to switch applications
    - Edge devices run multiple kernels on limited resources
    - Multiple users share the same hardware in the cloud
- Fast configuration is key to prevent resources from sitting idle

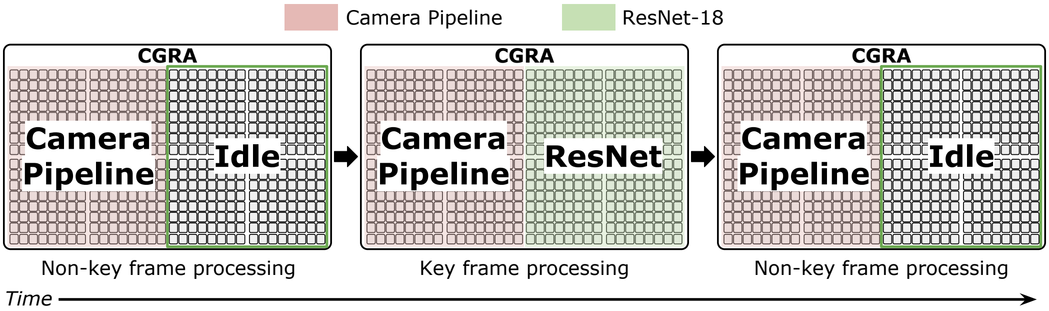




Maximizing Resource Utilization

In a stream of images:

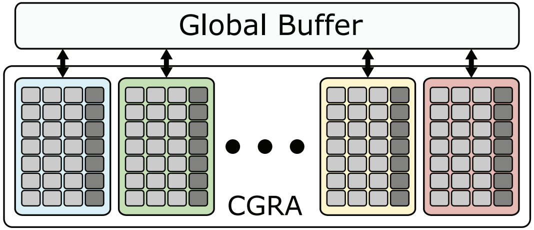
- Every frame processed by a camera pipeline
- Only key frames got through ResNet-18 for object detection



Dynamic Partial Reconfiguration

Dynamic partial reconfiguration (DPR) enables repurposing of unused tiles for additional computation during runtime

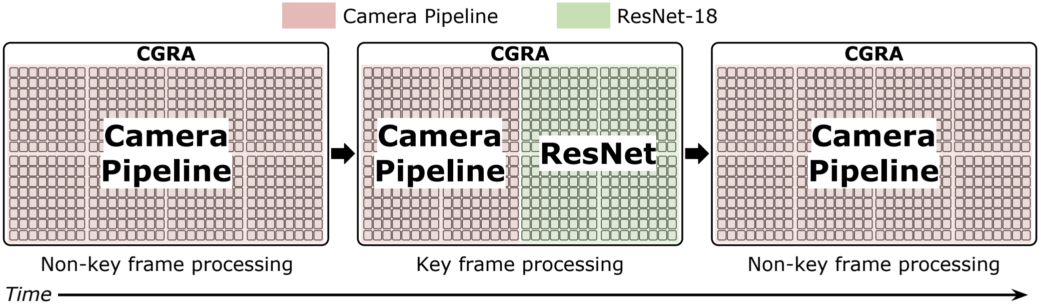
- First CGRA reconfiguration network specialized for high performance DPR
- Supports up to eight different kernels



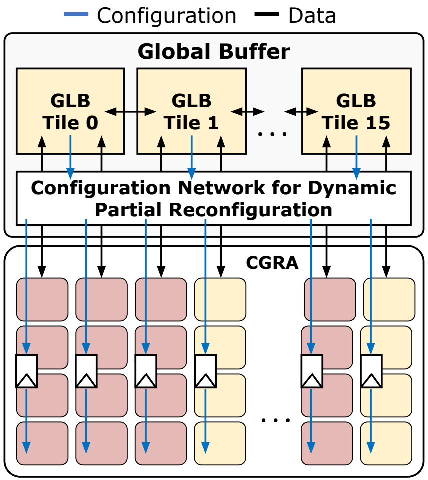
Dynamic Partial Reconfiguration

In a stream of images:

- Every frame processed by a camera pipeline
- Only key frames got through ResNet-18 for object detection



Dynamic Partial Reconfiguration in the GLB



- Achieves high configuration throughput using
  - **Parallel** GLB tiles
  - **Pipelined** configuration network
- Low area overhead by **sharing storage** between application and configuration data

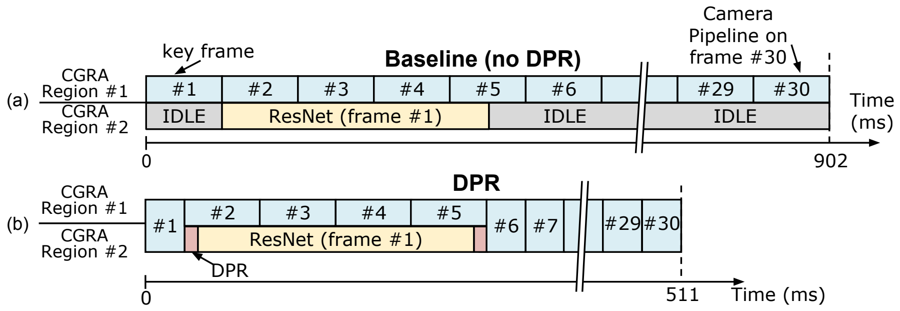
Performance Benefits of DPR in Amber

Configures full array in 3.5μs

- 36.5$\times$ more configuration throughput than FPGA

	Max. Freq.	Interface Bitwidth	Config Energy	Peak Throughput
Amber DPR	520 MHz	448 bit	57.4 pJ/config	29.1 GB/s
Amber AXI-Lite	660 MHz	32 bit	39454.5 pJ/config	44 MB/s
FPGA (Xilinx ICAP)	200 MHz	32 bit	–	800 MB/s

Camera Pipeline and ResNet with DPR

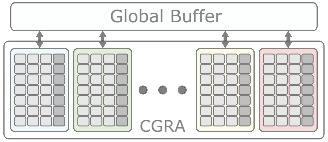


	Frequency (MHz)	Execution Time (ms/30 frames)	Energy (J/30 frames)	Energy-Delay Product (J·s/30 frames)
Baseline	200	902	217	196
DPR	200	511 (-**43.3%**)	154 (-**29.0%**)	78 (-**60.2%**)

Contributions

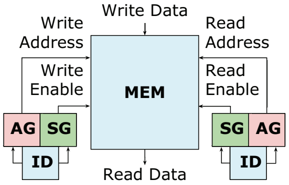
Configuration

Fast dynamic partial reconfiguration that runs up to eight kernels in parallel



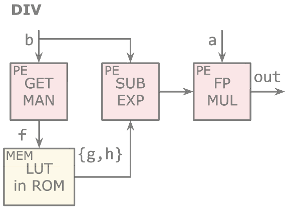
Memory

Efficient streaming memories for affine patterns



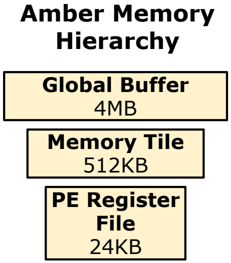
Compute

Low-overhead complex arithmetic support



On-Chip Streaming Memories for Affine Patterns

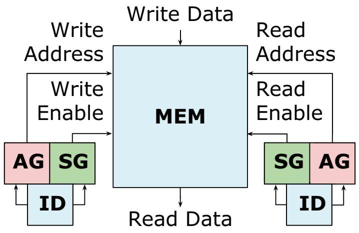
- Accelerators commonly use direct memory access engines
  - Too general and have high area/energy overheads
- Amber's on-chip memory controllers are specialized for affine access patterns seen in dense linear algebra applications
- Used in all levels of the memory hierarchy
  - Global buffer
  - MEM
  - PE register file



On-Chip Streaming Memories for Affine Patterns

Affine pattern:

```
for y in 0:ry
    for x in 0:rx
        addr = sx*x + sy*y + offset
```



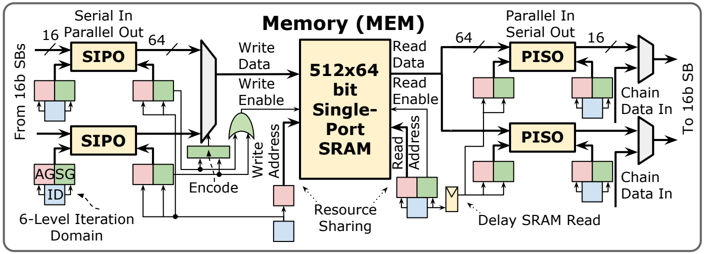
1. Iteration domain (ID): specifies range of memory operations
2. Address generator (AG): computes affine addresses from a set of strides and ID values
3. Schedule generator (SG): produces read/write enables, similarly to AG

- Parameters extracted from application by the compiler

Streaming Memory Optimizations in MEM

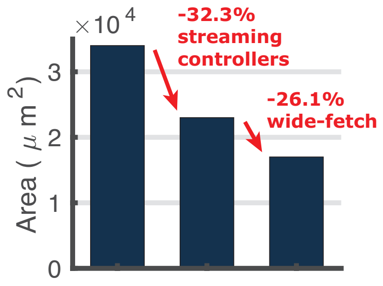
Streaming memory in MEM has further optimizations:

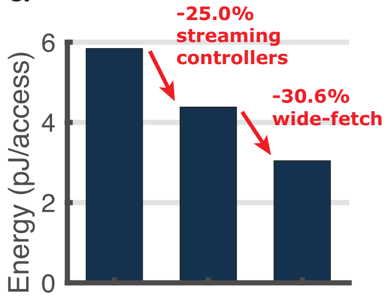
1. Wide-fetch SRAM: lower access energy per byte (0.81 pJ vs 1.65 pJ for single-fetch SRAM)
2. Resource sharing of ID/AG/SGs to reduce area
3. Recurrence relations: eliminates multiplier when calculating affine patterns



Streaming Memory Optimizations in MEM

- Save another 26.1% area and 30.6% energy using wide-fetch SRAM
- Overall, save 50% in area and 48% in energy

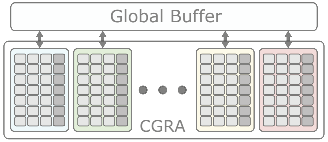




Contributions

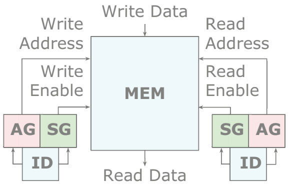
Configuration

Fast dynamic partial reconfiguration that runs up to eight kernels in parallel



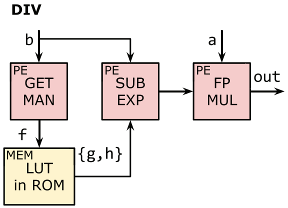
Memory

Efficient streaming memories for affine patterns



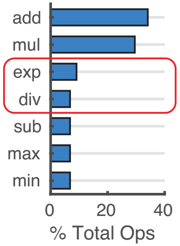
Compute

Low-overhead complex arithmetic support

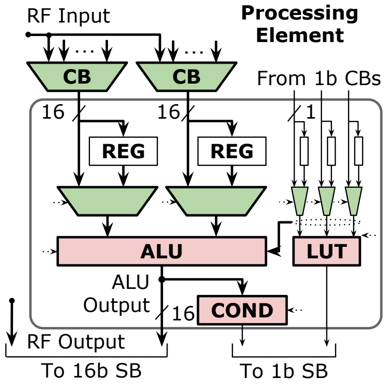


Complex Arithmetic Operations

- Image processing and computer vision kernels require complex arithmetic operations but are infrequently used
  - BFloat16 division, natural logarithm, sine, exponential
  - 15% of operations in non-local means (nlmeans) are complex
- How can we support complex operations?
  - Offload to CPU (slow)
  - Dedicated hardware in each PE (expensive)
  - A compromise?



Low-Overhead Complex Arithmetic in Amber

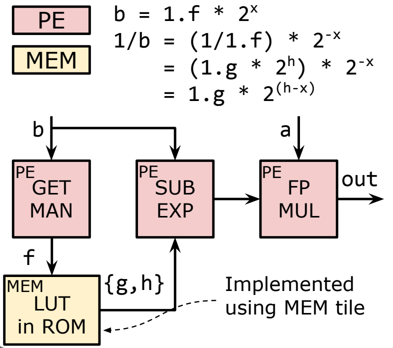


ALU Operations

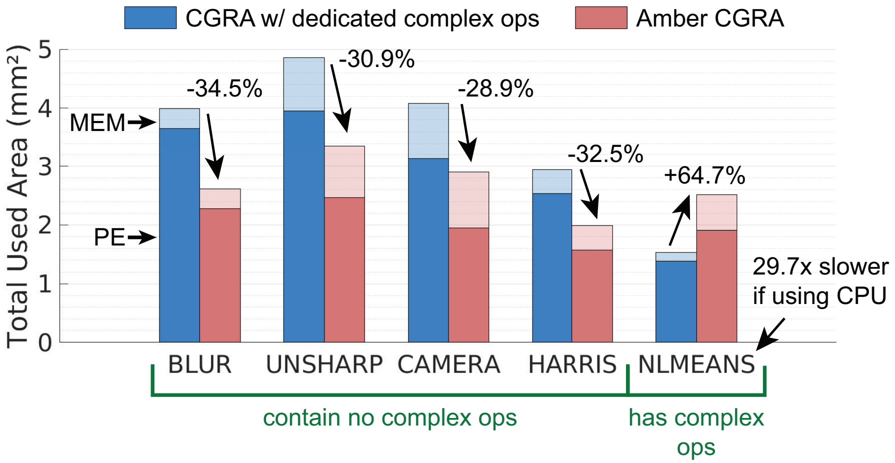
INT/BIT	**BFloat**
ADD	ADD
SUB	SUB
ADC	CMP
SBC	MUL
ABS	GETMAN
GTE	ADDIEXP
LTE	SUBEXP
SEL	EXP2F
MUL	F2INT
SHR	GETFR
SHL	INT2F
OR	
AND	
XOR	

DIV out = a/b = a * (1/b)

$$b = 1.f * 2^{x}$$
$$1/b = (1/1.f) * 2^{-x}$$
$$= (1.g * 2^{h}) * 2^{-x}$$
$$= 1.g * 2^{(h-x)}$$

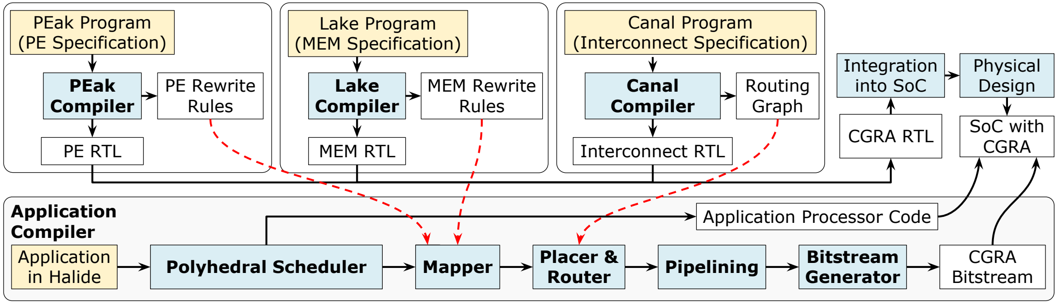


Low-Overhead Complex Arithmetic in Amber



Agile Accelerator-Compiler Design Flow

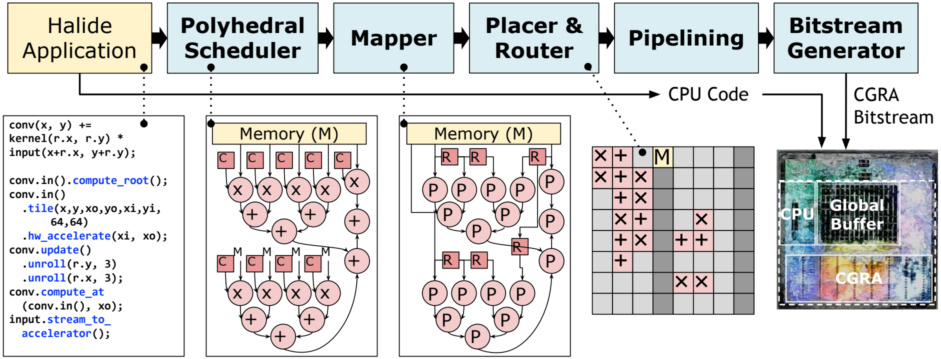
- Domain-specific language-based hardware generation flow
- Automatically updates application compiler flow to run applications



Application Flow

End-to-end compiler maps Halide applications onto CGRA

- 12.4× faster than Vivado FPGA compiler



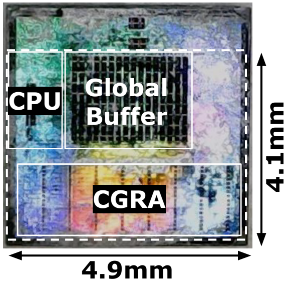
```
conv(x, y) +=
kernel(r.x, r.y) *
input(x+r.x, y+r.y);

conv.in().compute_root();
conv.in()
  .tile(x,y,xo,yo,xi,yi,
      64,64)
  .hw_accelerate(xi, xo);
conv.update()
  .unroll(r.y, 3)
  .unroll(r.x, 3);
conv.compute_at
  (conv.in(), xo);
input.stream_to_
  accelerator();
```

C = constant, P = PE, M = memory, R = shift register

Comparison with State of the Art

1.7× better energy efficiency with 36.7× throughput



	This Work	VLSI 2019 Whatmough	ISSCC 2021 Schmidt	VLSI 2019 Rovinski
Architecture	SoC with CGRA	SoC with FPGA	Multicore Vector CPU	Multicore CPU
Node	TSMC 16nm	TSMC 16nm	TSMC 16nm	TSMC 16nm
Area (mm^2)	20.1	25	24.01	15.25
Precision	BF16, INT16-64	FP16-64, INT16-64	FP16-64, INT32-64	INT32
SRAM (MiB)	4.5	9.025	4.5	3.875
Voltage (V)	0.84-1.29	0.5-1.0	0.55-1.0	0.60-0.98
Freq (MHz)	955	$>$ 1000	1440	10-1400
Peak GOPS	367	10-56.2	368.4	695
GOPS/W	538	312.4	209.5	93.04

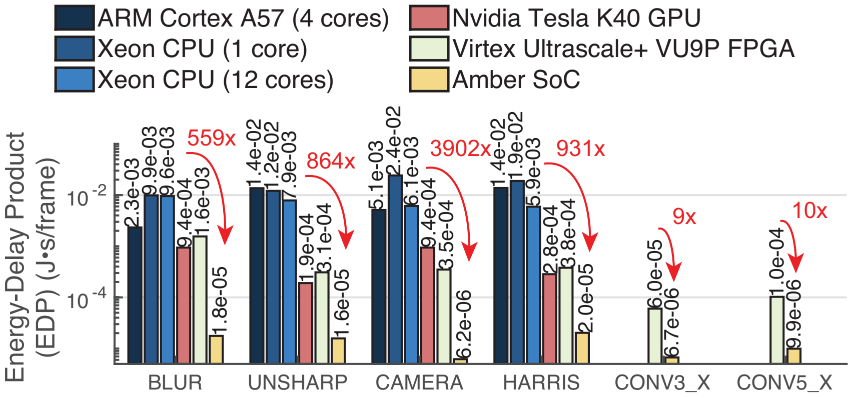
Benchmark Application Suite

Benchmark apps written in Halide, compared against CPU, GPU, FPGA:

- Image processing
  - Blur: image blur
  - Unsharp: enhances local contrast by smoothing an image
  - Camera pipeline: processes raw data from an image sensor into a color image
- Computer vision
  - Harris: detects corners
- Machine learning
  - ResNet-18: image classification

Results: Energy-Delay Product

Amber up to 3902×, 152×, and 88× better EDP than CPU, GPU, and FPGA



Summary of Key Contributions

- Amber is an SoC designed for flexible and efficient acceleration for image processing, computer vision, and machine learning
  - **Configuration**: fast dynamic partial reconfiguration at runtime
  - **Memory**: efficient streaming memories for affine patterns
  - **Compute**: low-overhead complex arithmetic operation support
- Automatic end-to-end compiler maps applications onto Amber
- Amber achieves 3902×, 152×, and 88× better EDP over CPU, GPU, and FPGA, respectively
- Enables efficient domain, rather than single application, acceleration

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