

Modeling FPGA Performance for Image and Audio Processing

David Durst



Andrew Adams



Shoaib Kamil



Dillon Huff



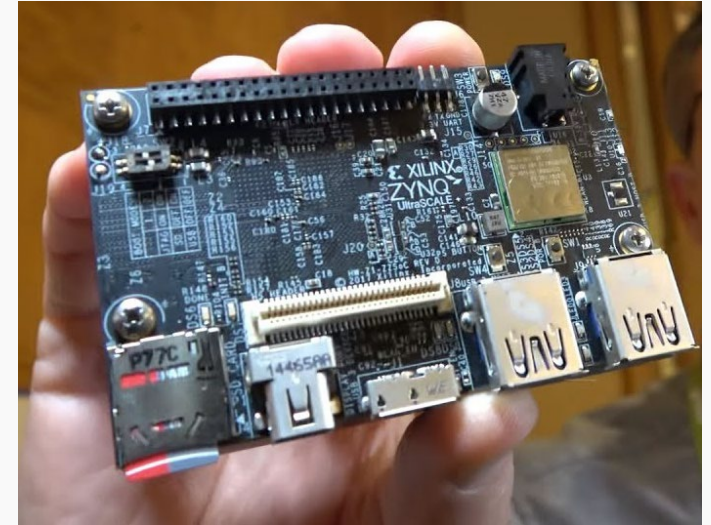
Kayvon Fatahalian



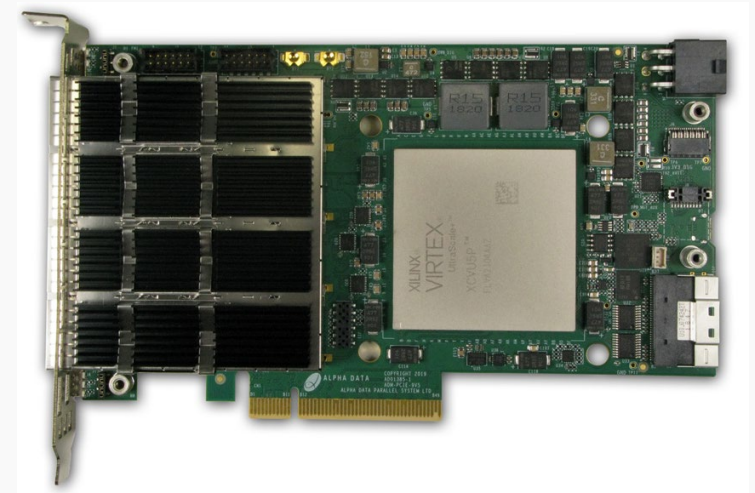
Pat Hanrahan



My Summer: Modeling FPGA Performance At Adobe



AHA



Apps Explore Space of Bottlenecks

Application	Hypothesized Bottleneck	Hypothesis Confirmed?	Why?
Compositing	Memory Bandwidth		
Stencil Chain	Compute Throughput		
Exposure Fusion	Cache Organization		
U-Net	Compute Throughput		
Audio ConvNet	Compute Latency		

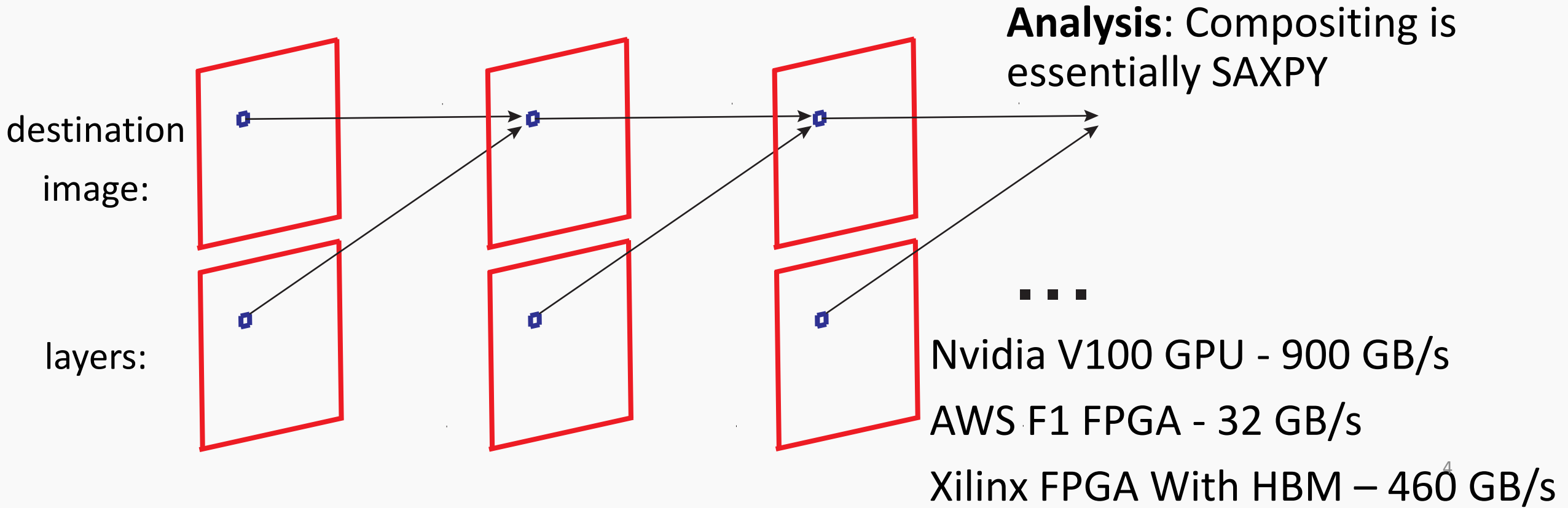


Image Compositing

Hypothesis Confirmed – Memory Bandwidth

Compositing – a sequence of operations where image data is combined

Hypothesis: bandwidth-bound app, best on processor with most memory bandwidth



Compositing Hypothesis Confirmed – Just Bandwidth

Application	Hypothesized Bottleneck	Hypothesis Confirmed?	Why?
Compositing	Memory Bandwidth	Yes	Compositing is similar to SAXPY
Stencil Chain	Compute Throughput		
Exposure Fusion	Cache Organization		
U-Net	Compute Throughput		
Audio ConvNet	Compute Latency		

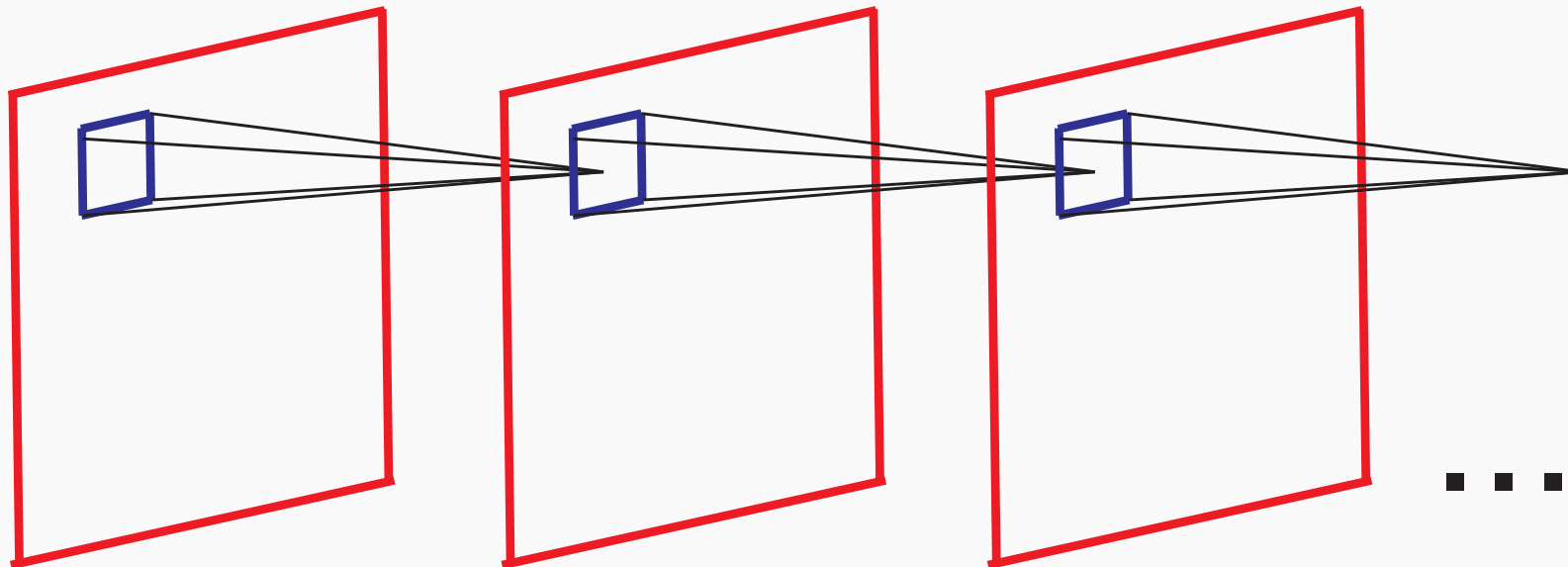


Stencil Chain

More Compute, Less Memory Traffic

Stencil Chain – sequence of non-compositing image processing operations like blurs and other filters that repeatedly modify a single input image

Hypothesis: compute-bound app, best on processor with most compute



GPU Has More Compute, So GPU Should Win, Right?

Processor	Peak Compute (UInt16 TOps)	Memory Bandwidth (GB/s)	TDP (W)
Nvidia V100	5 (CUDA Cores) / 56 (Tensor Cores)	900	300
AWS F1 FPGA (VU9P)	1.5	32	225

Stencil Chain

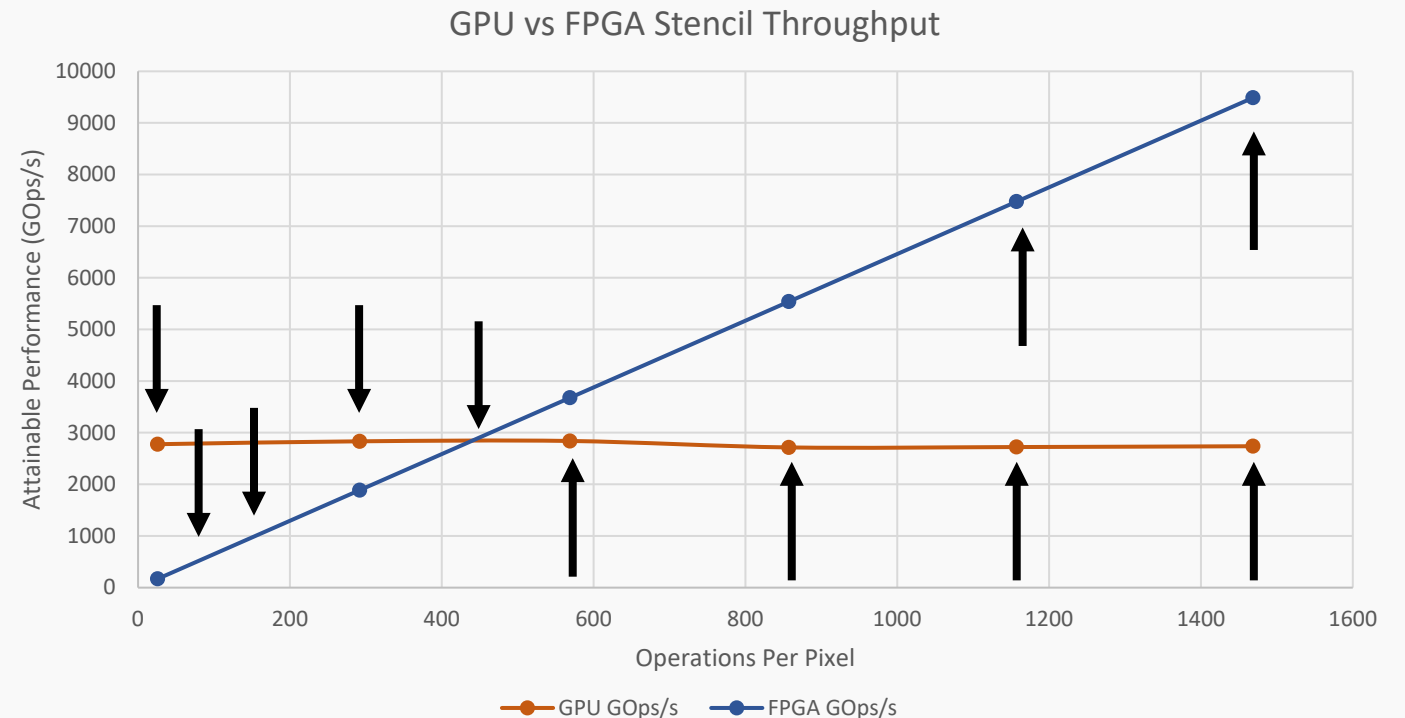
FPGA Wins A Supposedly Compute-Bound Task

Repeated applications of
5x5 blurs to 1536x2560
image on Nvidia V100 and
AWS F1

Experimental GPU Results,
Modeled FPGA Results

Dots are at 1, 11, 21, 31, 41,
51 blurs

FPGA outperforms GPU at
~15 blurs



Huh? GPU Has A Lot More Compute. How Can The FPGA Win A Compute-Bound Task?

Processor	Peak Compute (UInt16 TOps)	Memory Bandwidth (GB/s)	TDP (W)
Nvidia V100	5 (CUDA Cores) / 56 (Tensor Cores)	900	300
AWS F1 FPGA (VU9P)	1.5	32	225

Why pick parallelism for FPGA model based on memory bandwidth if app is compute bound?

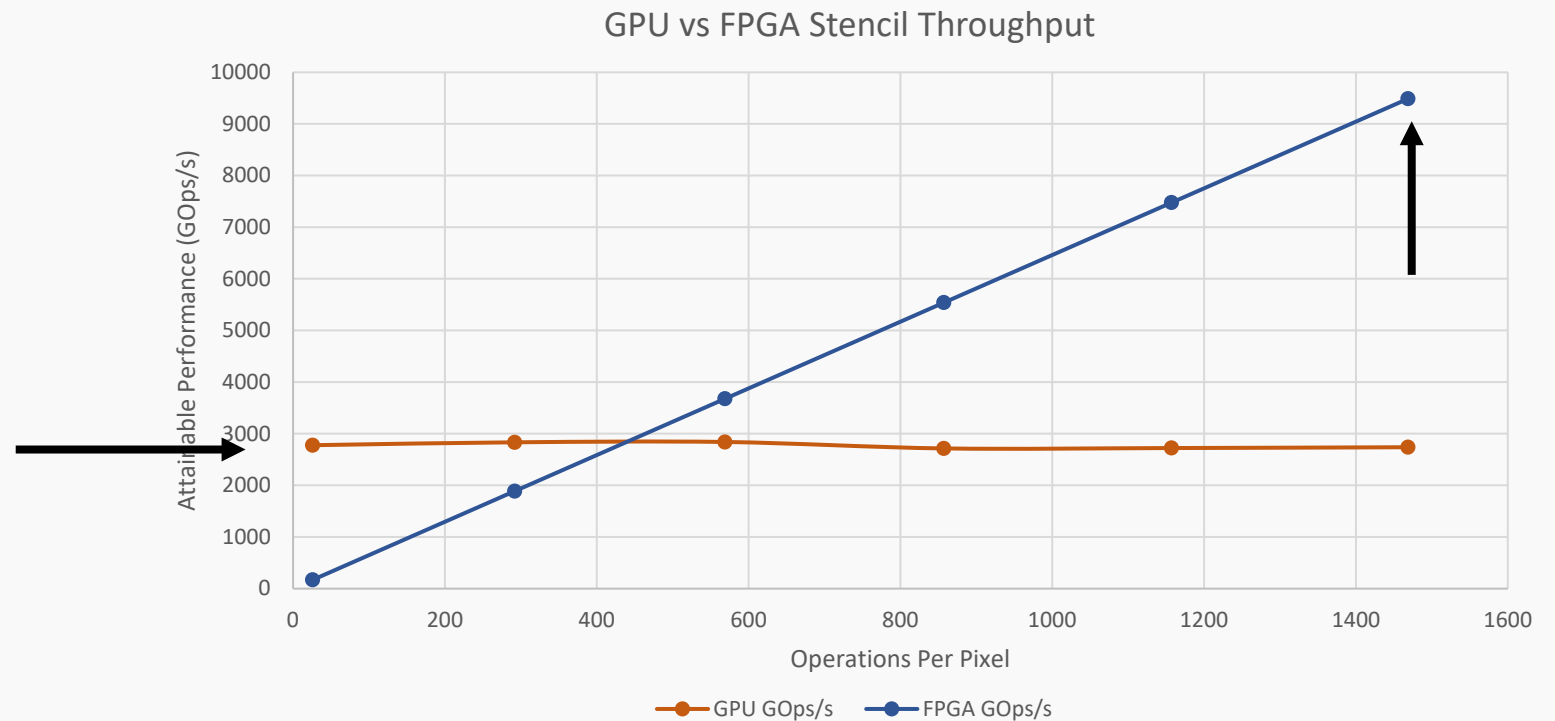
5x5 Stencil Chain on 5MP Image Has Different Bottlenecks on GPU and FPGA

FPGA

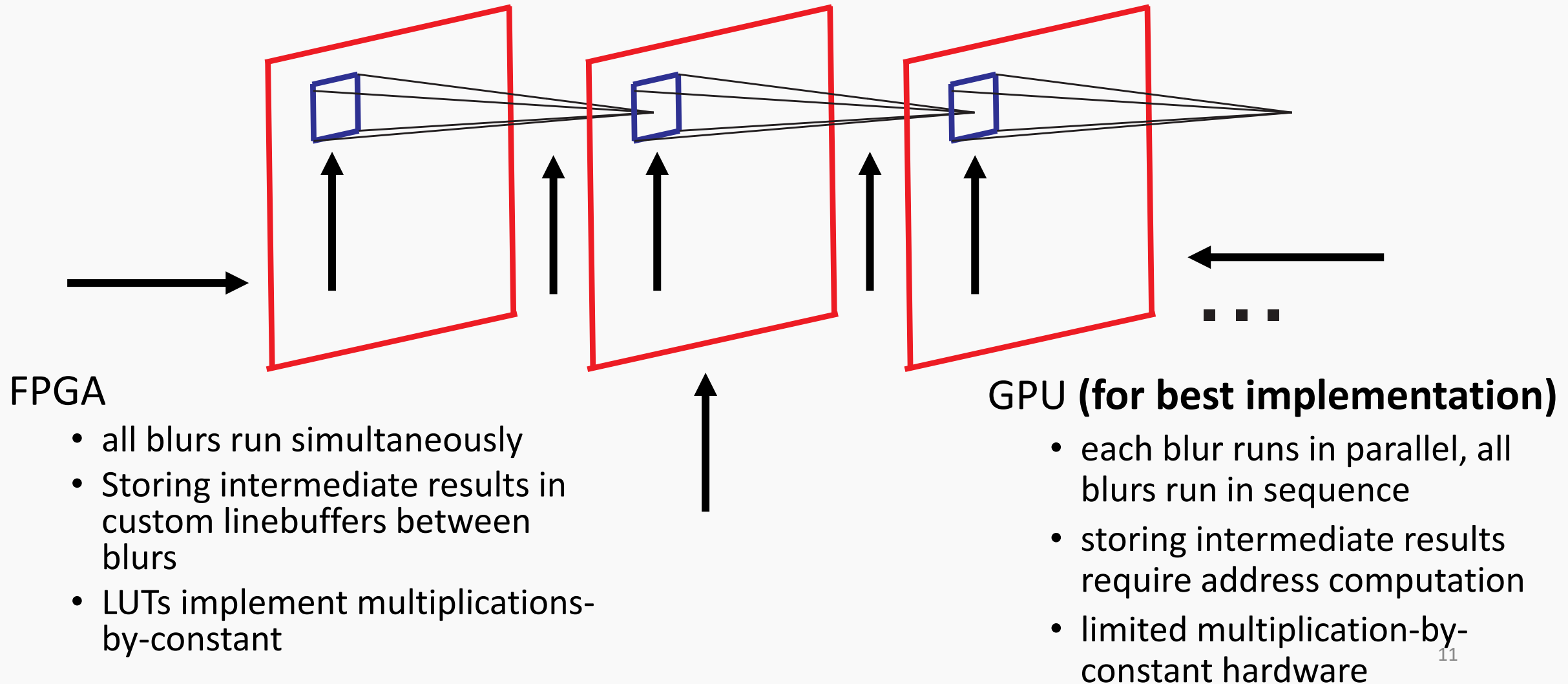
- saturating memory bandwidth at any number of stages
- compute limits number of stages

GPU

- 2.8 TOps observed performance for all numbers of stages is 60% of peak
- 5 TOps peak GPU performance less than peak 9.5 TOps peak FPGA performance

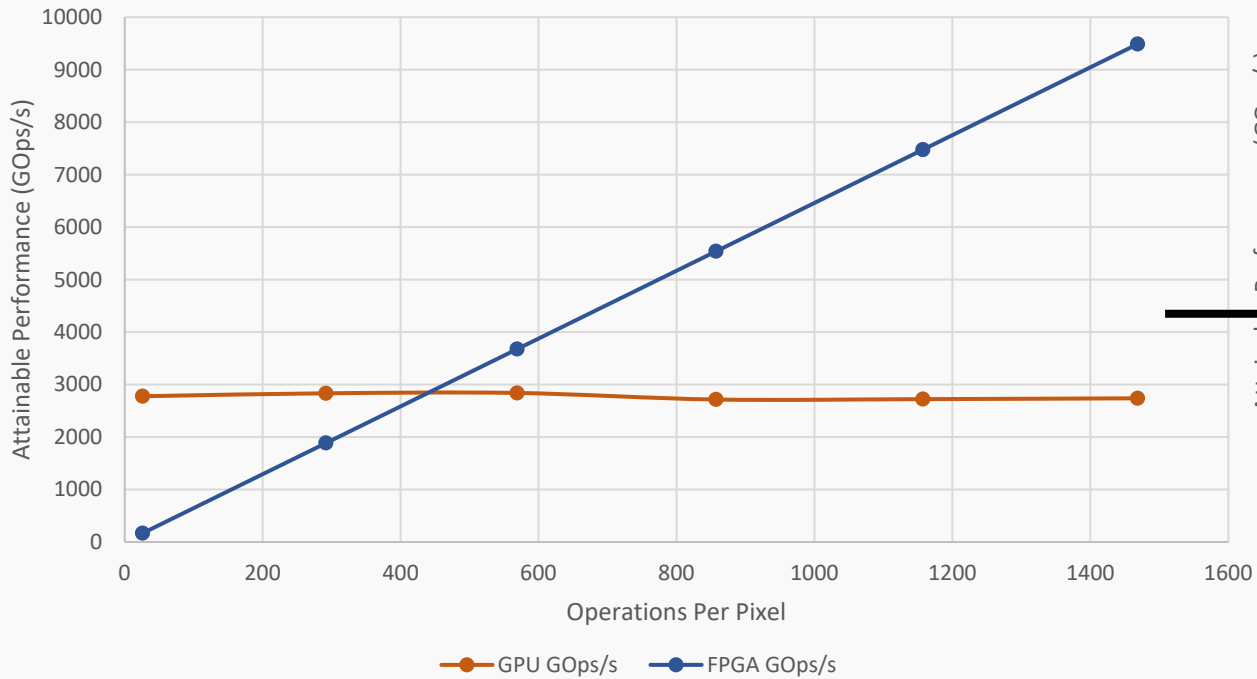


FPGA's Custom Memories and ALUs Enable Greater Peak Compute



When GPU At 88% of Peak Compute, Still Far Less Than FPGA

GPU vs FPGA Stencil Throughput, 26 Ops Per Stage




GPU vs FPGA Stencil Throughput, 78 Ops Per Stage



Stencil Hypothesis Rejected – Custom Memories and ALUs

Application	Hypothesized Bottleneck	Hypothesis Confirmed?	Why?
Compositing	Memory Bandwidth	Yes	Compositing is similar to SAXPY
Stencil Chain	Compute Throughput	Yes (GPU) No (FPGA)	More efficient ALU usage due to custom memories, More ALUs due to ALU specialization
Exposure Fusion	Cache Organization		
U-Net	Compute Throughput		
Audio ConvNet	Compute Latency		

Stencil Hypothesis Rejected – Custom Memories and ALUs

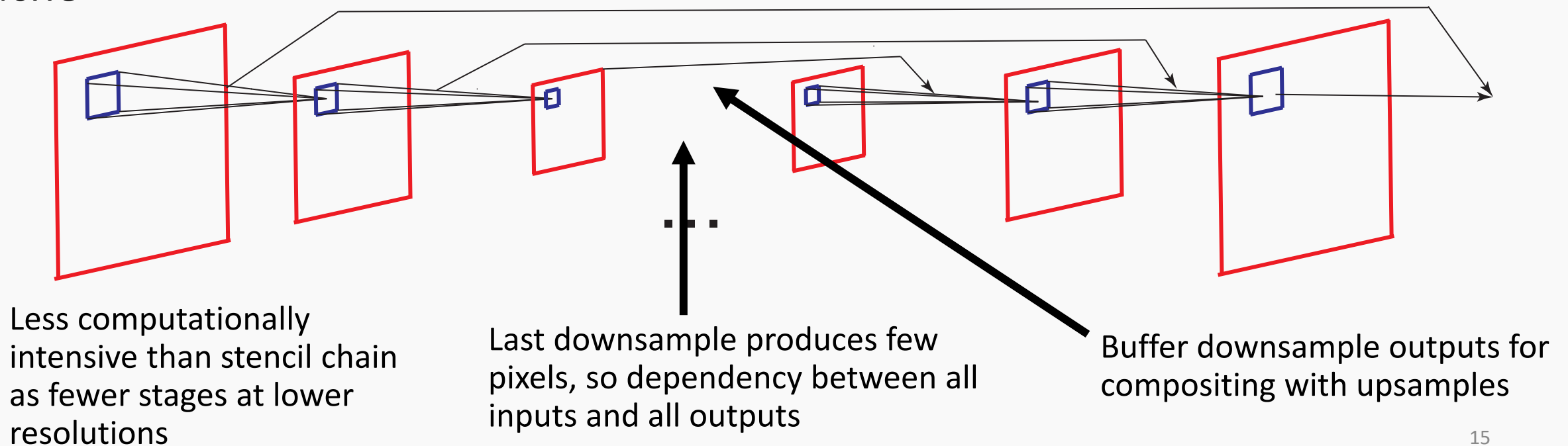
Application	Hypothesized Bottleneck	Hypothesis Confirmed?	Why?
Compositing	Memory Bandwidth	Yes	Compositing is similar to SAXPY
Stencil Chain	Compute Throughput	Yes (GPU) No (FPGA)	More efficient ALU usage due to custom memories, More ALUs due to ALU specialization
Exposure Fusion	Cache Organization		
U-Net	Compute Throughput		
Audio ConvNet	Compute Latency		

Exposure Fusion

Less Compute, More Memory Traffic

Exposure Fusion – pyramid of downsamples and upsamples with compositing operations at each resolution

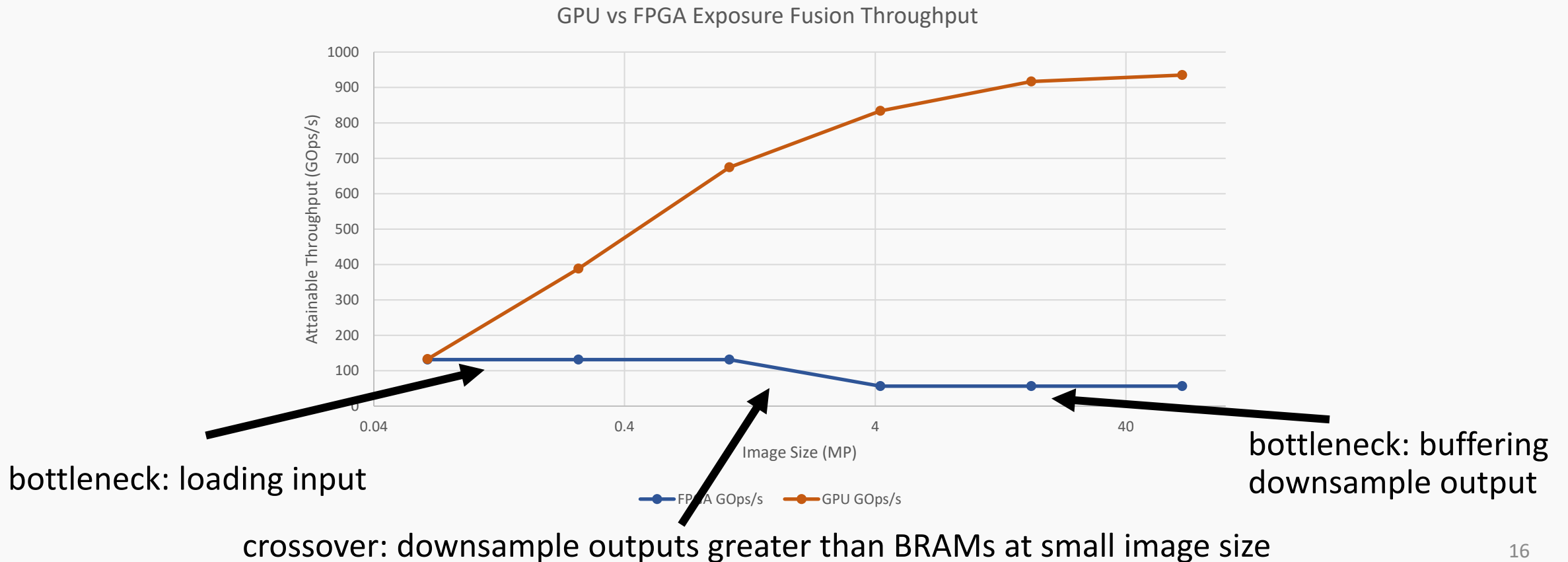
Hypothesis: cache-limited app, best on processor that keeps intermediate tiles in cache



Exposure Fusion


No Fusion, Only Acceleration Approach Is More Bandwidth

Pyramid creates all-to-all dependency between input and output pixels, so large working set and lots of DRAM traffic



Exposure Fusion Hypothesis Rejected – Working Set Too Big For Caching

Application	Hypothesized Bottleneck	Hypothesis Confirmed?	Why?
Compositing	Memory Bandwidth	Yes	Compositing is similar to SAXPY
Stencil Chain	Compute Throughput	Yes (GPU) No (FPGA)	More efficient ALU usage due to custom memories, More ALUs due to ALU specialization
Exposure Fusion	Cache Organization	No	All-to-all dependency in pyramid creates large working set, lots of DRAM traffic, preventing good hardware solution
U-Net	Compute Throughput		
Audio ConvNet	Compute Latency		

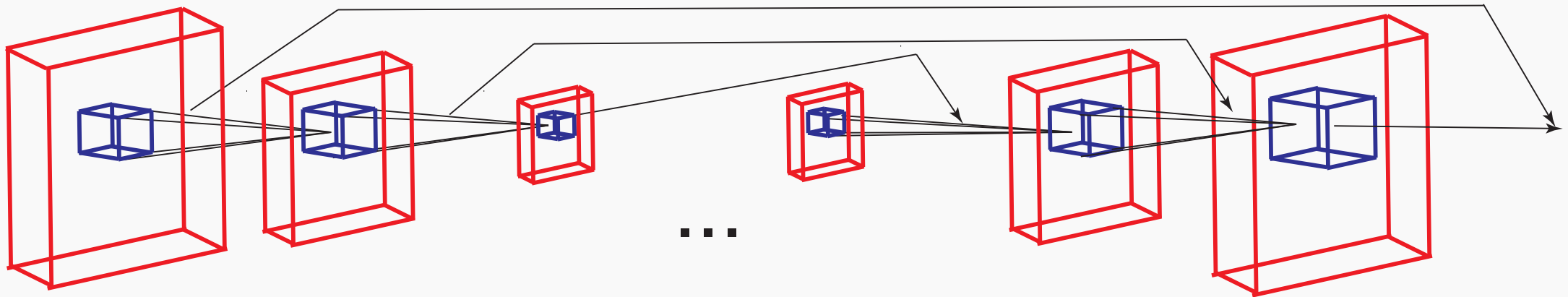


U-Net

Exposure Fusion, But With More Compute

U-Net* – pyramid of neural network convolution layers at different resolutions. Unlike exposure fusion, expensive matrix multiplication at each resolution

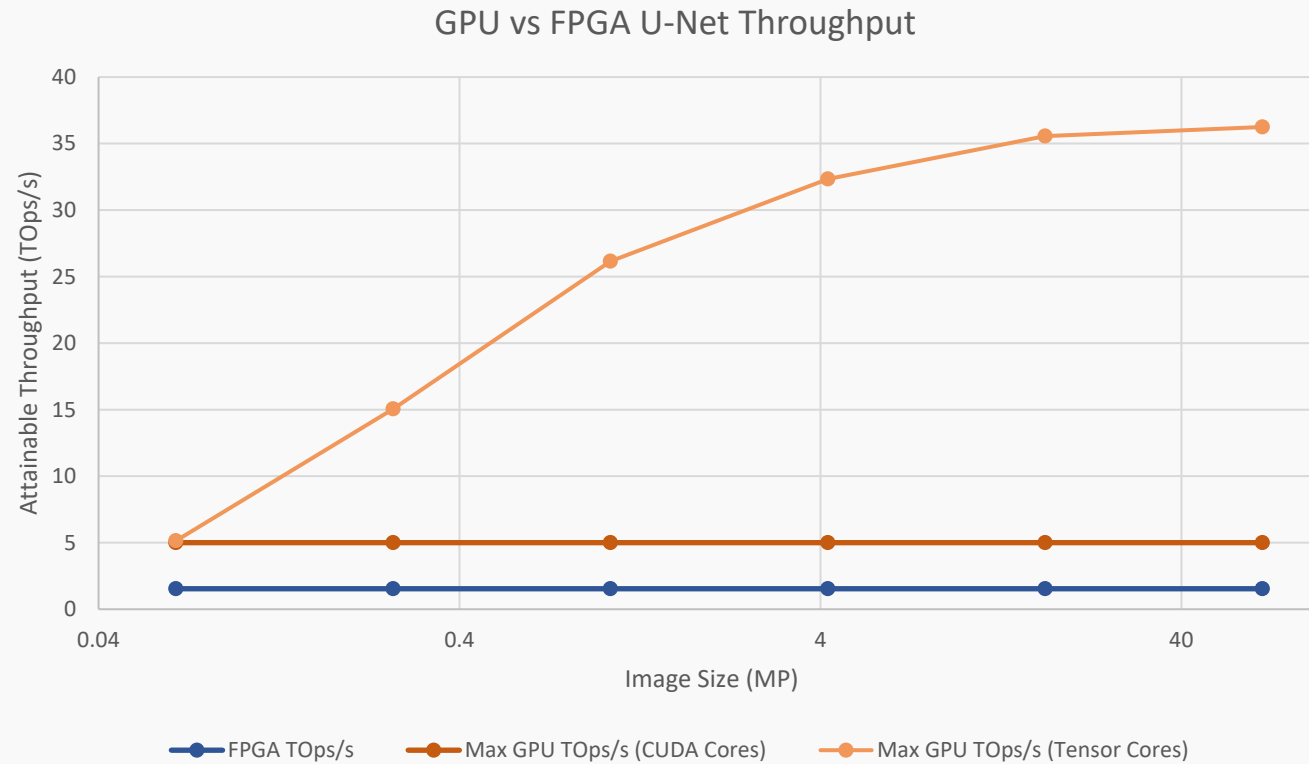
Hypothesis: compute-limited app, FPGA DSPs within 4x of CUDA cores in peak compute performance




*Technically Exposure Fusion memory traffic with U-Net compute. This comparison intended to explore pyramid structure with more compute. 18

U-Net

GPU has ASIC Optimized For Matrix Multiplication



U-Net Hypothesis Partially Accepted – More Compute Available Than Expected

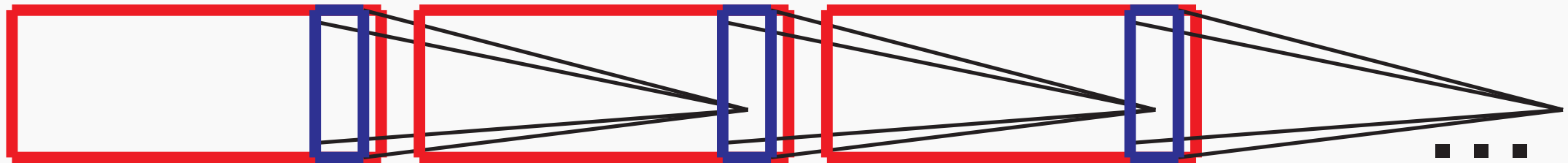
Application	Hypothesized Bottleneck	Hypothesis Confirmed?	Why?
Compositing	Memory Bandwidth	Yes	Compositing is similar to SAXPY
Stencil Chain	Compute Throughput	Yes (GPU) No (FPGA)	More efficient ALU usage due to custom memories, More ALUs due to ALU specialization
Exposure Fusion	Cache Organization	No	All-to-all dependency in pyramid creates large working set, lots of DRAM traffic, preventing good hardware solution
U-Net	Compute Throughput	Kinda Yes, Kinda No	Compute maps well to matrix multiplication ASICs in GPU
Audio ConvNet	Compute Latency		

Audio ConvNet

Latency-Sensitive, Mobile Application

Audio ConvNet – repeated matrix multiplication to 1D stream of samples grouped into time slices.

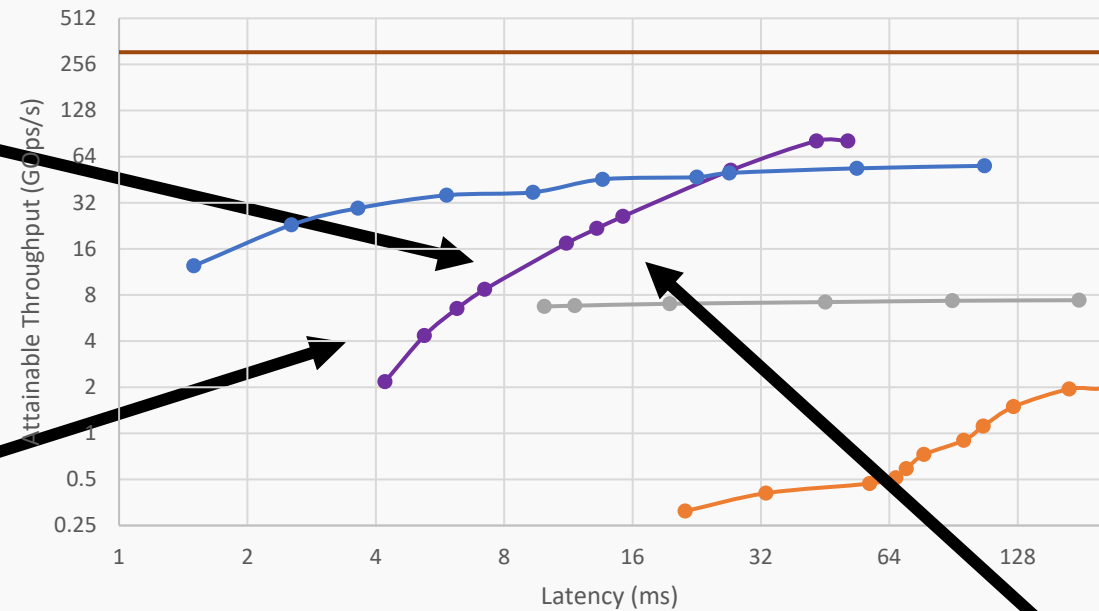
Hypothesis: latency-limited app, FPGA DSPs enable energy-efficient, low latency matrix multiplication



Audio ConvNet

It's All About the DSPs and Memory

Halide Audio ConvNet Latency vs Throughput



Fantastic peak performance

Performance comes from DSPs

AI Tensor Core packaged with Hexagon 695 DSP is estimated to have 5 TOps performance

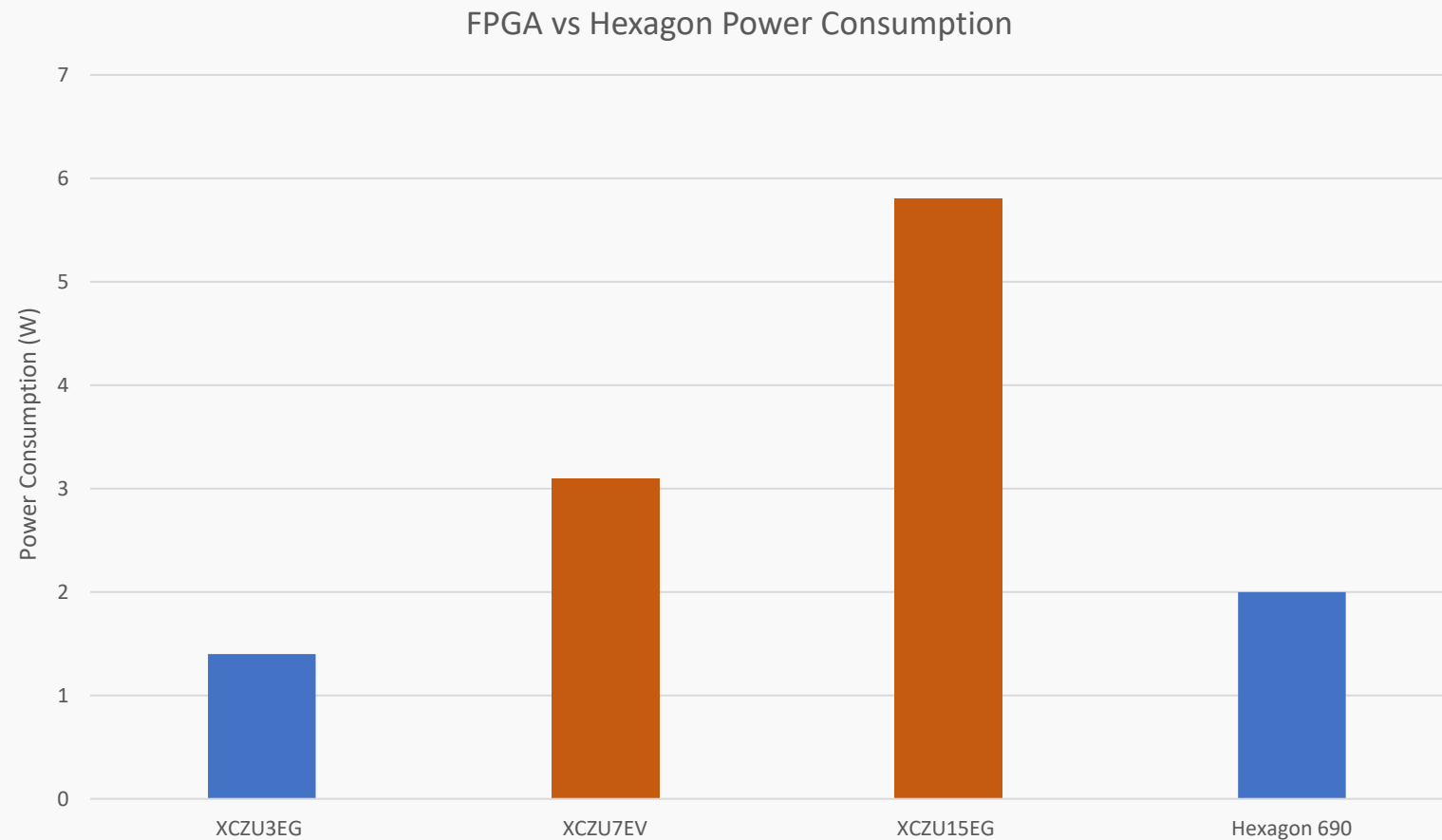
Compute vs Latency Pareto Curve

All mobile processors except Intel i9

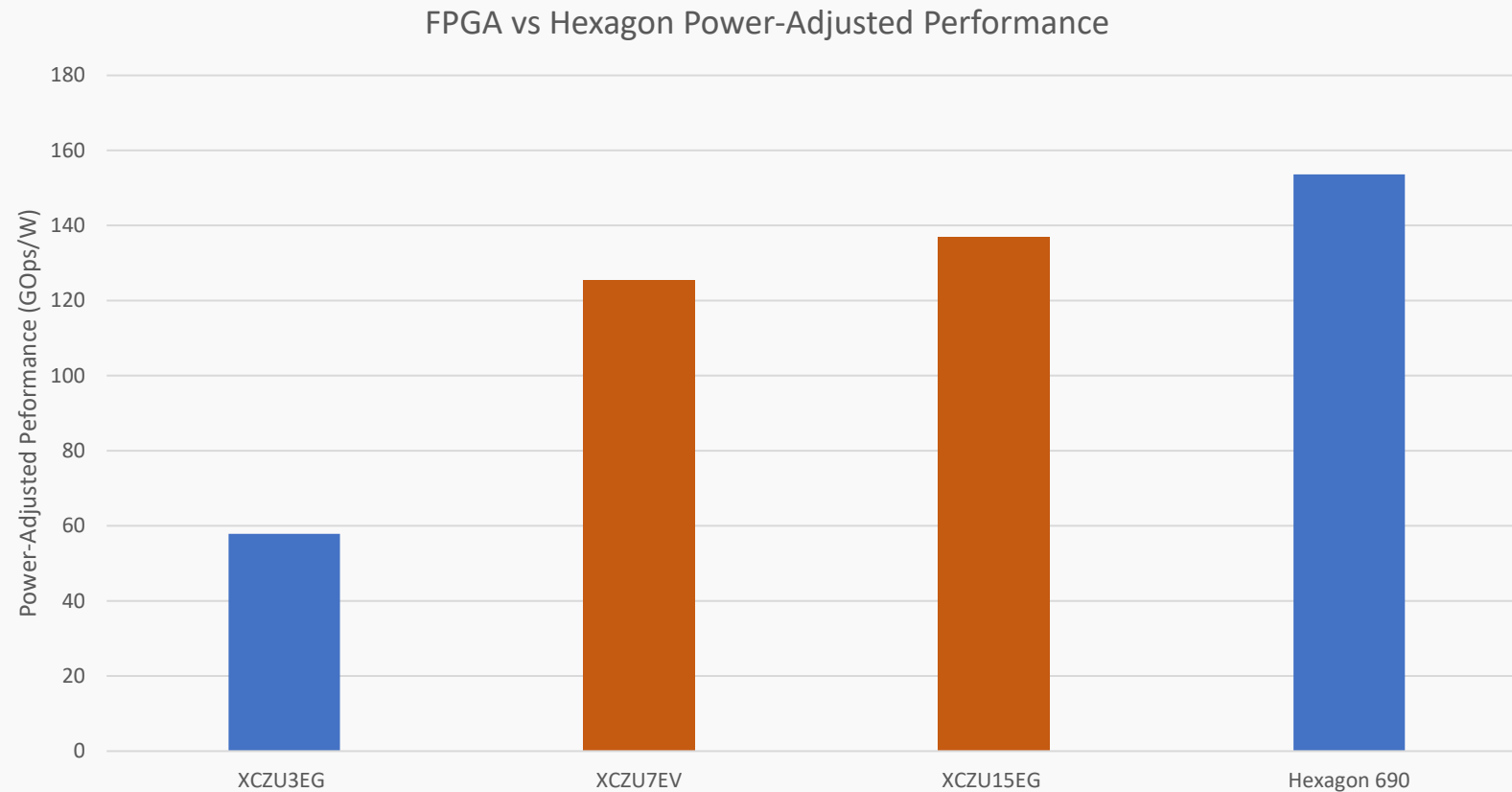
- ARM Cortex A53 (single core)
- Xilinx ZU3EG
- Hexagon 690 (peak performance)
- ARM Cortex A75 (single core)
- Intel i9-9960X (single core)

Latency comes from memory, not compute

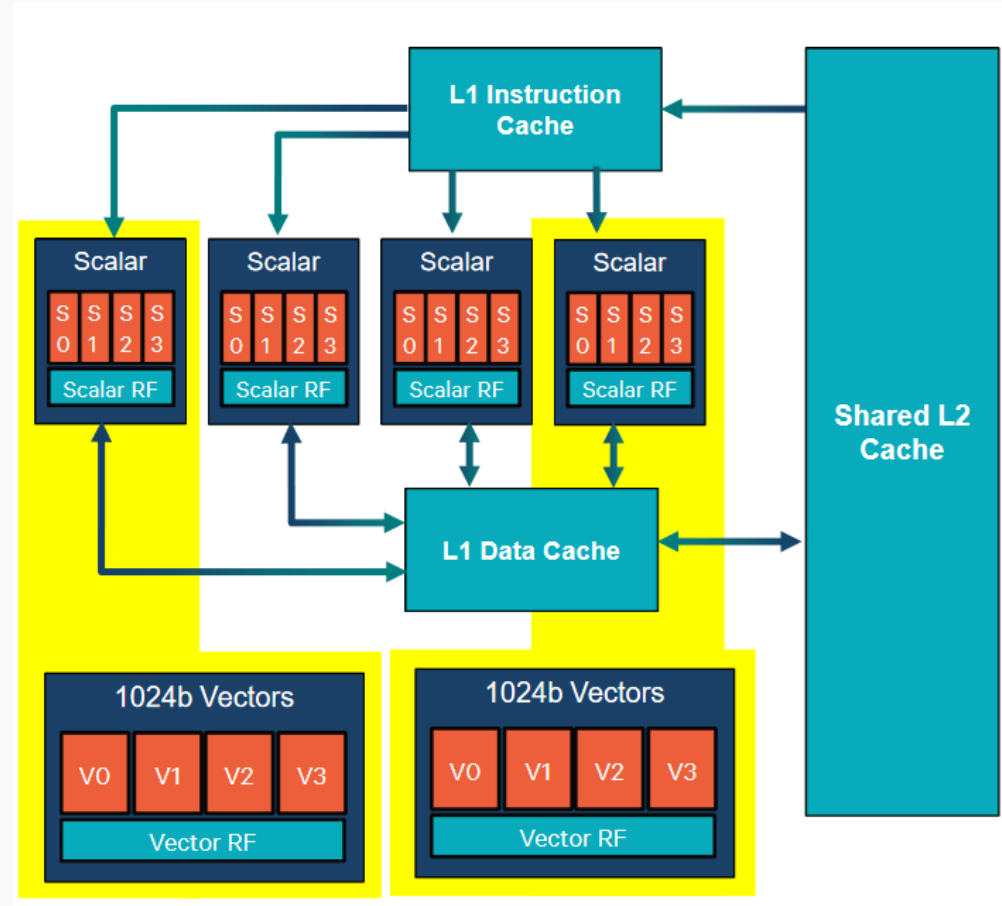
Only the XCZU3EG and Hexagon 690 Are Mobile Accelerators



Hexagon 690 Provides Competitive Performance Per Watt



Energy Efficient For Streaming Applications Since Hexagon 690 Is Standard Vector Processor

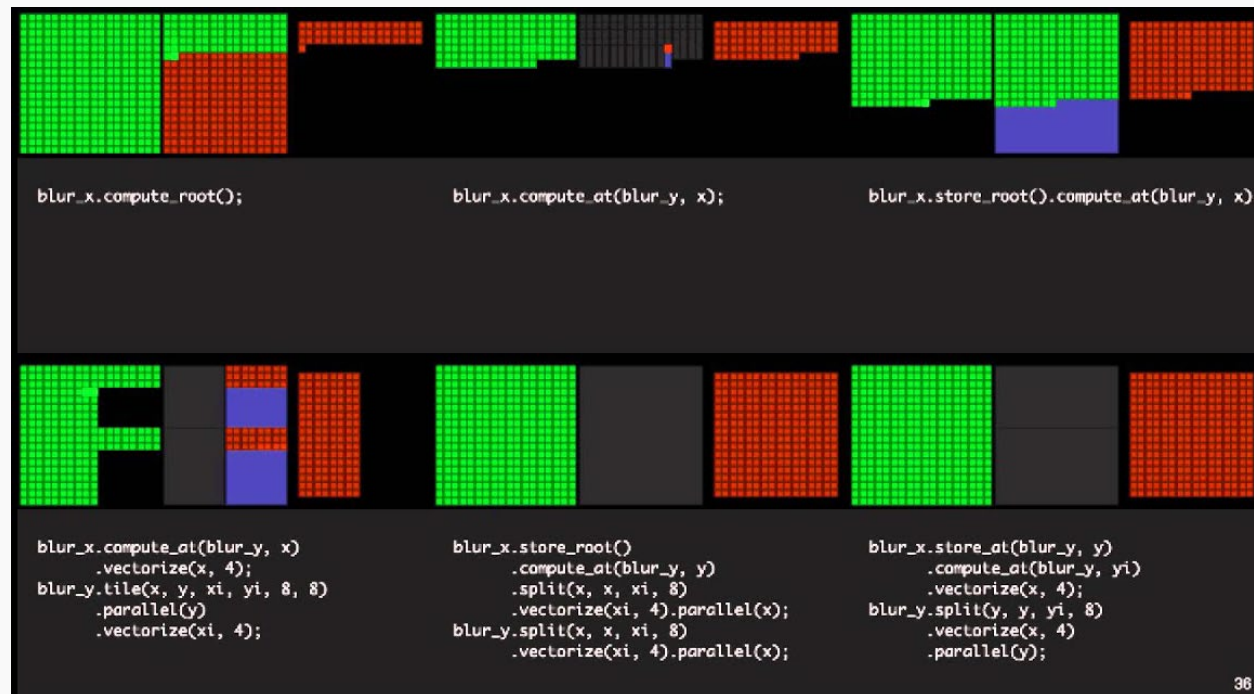


Audio ConvNet Hypothesis Rejected – Latency From Memory

Application	Hypothesized Bottleneck	Hypothesis Confirmed?	Why?
Compositing	Memory Bandwidth	Yes	Compositing is similar to SAXPY
Stencil Chain	Compute Throughput	Yes (GPU) No (FPGA)	More efficient ALU usage due to custom memories, More ALUs due to ALU specialization
Exposure Fusion	Cache System	No	All-to-all dependency in pyramid creates large working set, lots of DRAM traffic, preventing good hardware solution
U-Net	Compute Throughput	Kinda Yes, Kinda No	Compute maps well to matrix multiplication ASICs in GPU
Audio ConvNet	Compute Latency	No	Latency comes from memory system streaming weights; non-reconfigurable vector processors exist that provide energy efficient performance

AHA Enabled These Analyses, But We Need More Work On HLS Scheduling/RTL PnR Time

1. Thank you AHA for building the tools that enabled my analyses.
 1. <https://github.com/David-Durst/Halide/tree/durst-fpga>
2. But, we're not done, compilation too slow for experimental performance measurement instead of modeling



Memory-Bound Apps Are Rock

Matrix-Multiplication-Based, Compute-Bound Apps Are Hard Place

