Integration of Open-Source Vortex GPU into ESP

Michael Lippe, Biruk Seyoum, Gabriele Tombesi, Joseph Zuckerman, and Luca P. Carloni

June 28, 2026

OSCAR 2026

Columbia University
IN THE CITY OF NEW YORK

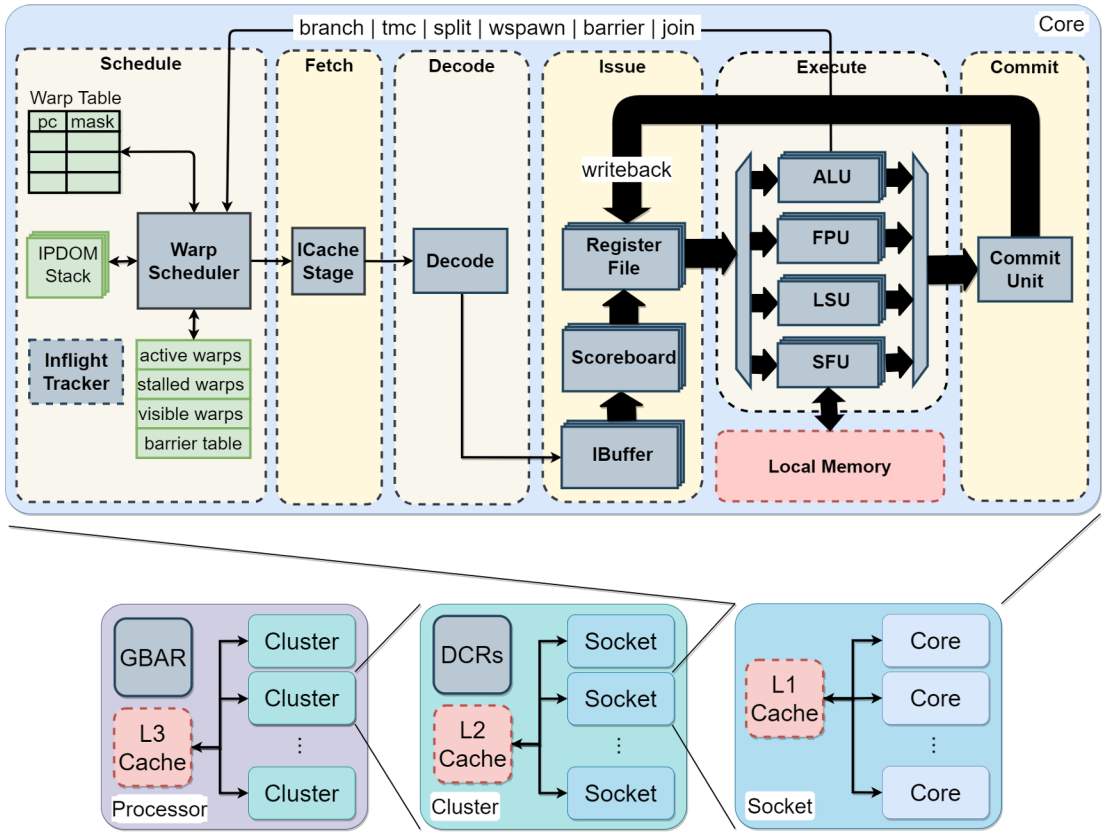
CS@CU Computer Science

State of GPU Space

- The advent of generative AI and increasing prevalence of Machine Learning makes GPUs and GPGPU workloads more important than ever
- GPU space is currently dominated by proprietary closed-source designs
- Open-source GPU implementations and their integration into other open-source projects are scarce
- This poses a significant barrier to accelerate GPGPU workloads in open-source SoC designs

Vortex GPU

- Vortex is one of the few readily available, tested, and maintained open-source GPU designs [1]
- Aim to integrate Vortex into ESP platform to enable seamless integration of an open-source GPU into custom SoC designs
- Vortex serves as a stress test for ESP's existing SoC infrastructure in terms of supporting larger, more compute and memory intensive accelerators

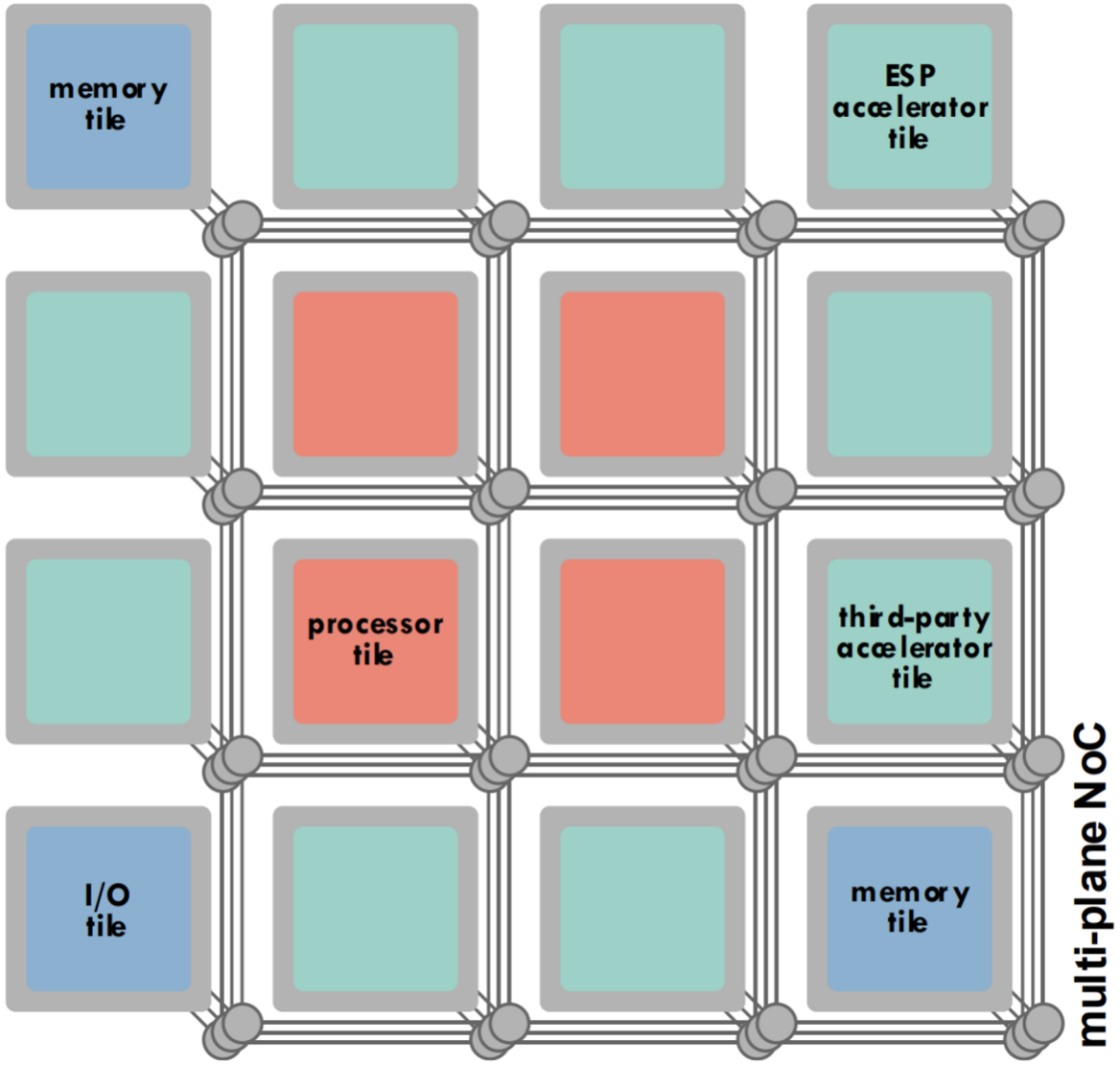


[1] B. Tine et al. MICRO '21 | [2] Vortex Github Repository

ESP Architecture

- Tile Based SoC Architecture
- SPARC and RISC-V CPUs
- Multi-Plane NoC
- Distributed Memory

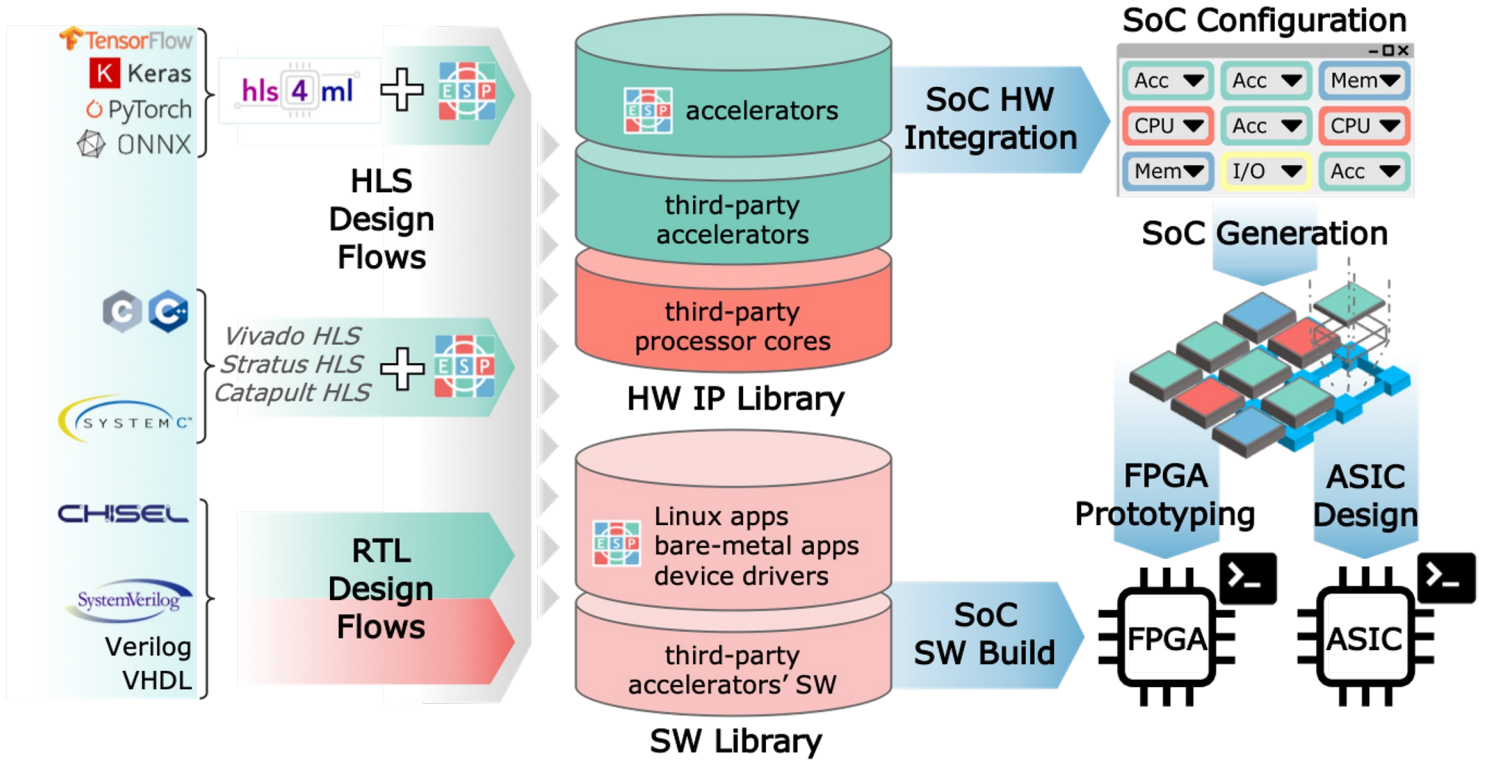
The ESP architecture is a **scalable**, **modular** and **heterogeneous distributed** system, giving processors and accelerators similar weight in the SoC



Source: P. Mantovani et al. ICCAD '20

ESP Methodology

- **SoC Methodology:**
  - Mix and match floor-plan with GUI
  - FPGA and ASIC flows
- **Accelerator Design:**
  - HLS and RTL design flows
  - Support for third party accelerators

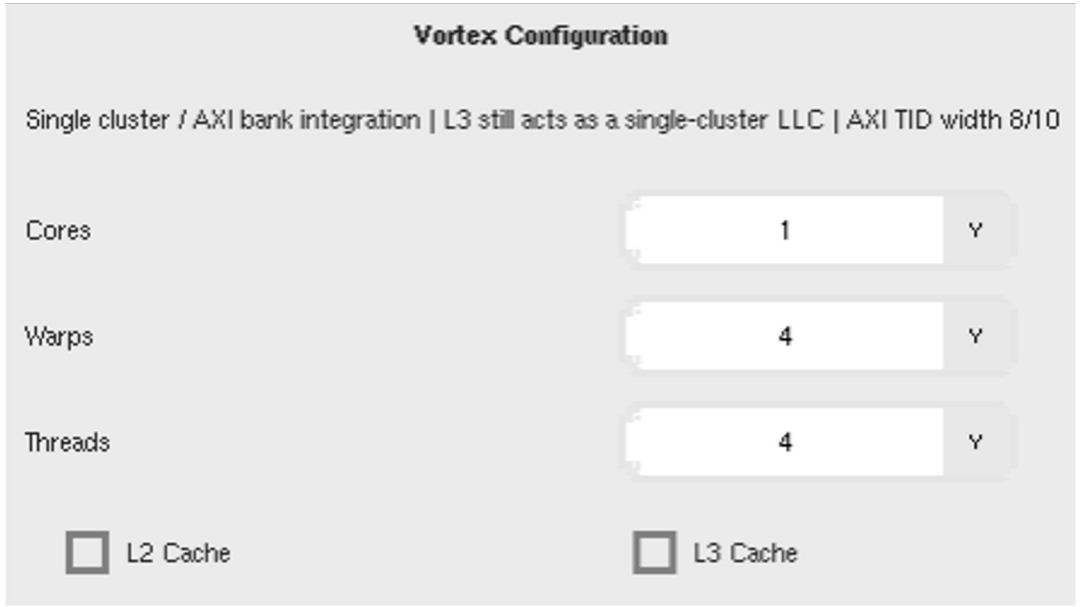


Source: https://www.esp.cs.columbia.edu/

Vortex in ESP: Hardware

- Integrated 64-bit version of Vortex v2.2
  - 8-byte memory block size to align with third-party socket's AXI data width
  - Vortex's 64-bit memory map truncated to fit within ESP's 32-bit memory space
  - Custom ESP wrapper to control Vortex and its DCR registers

- ESP GUI exposes Vortex configuration options
  - Configure number of cores, warps, and threads, and toggle L2 and L3 caches
  - Support configurations that keep Vortex's *tag-id* at or below ESP's maximum 10-bit AXI *tag-id* limit.
    - Up to 64 cores, 64 warps, 64 threads, and L2/L3 caches, though not all at once

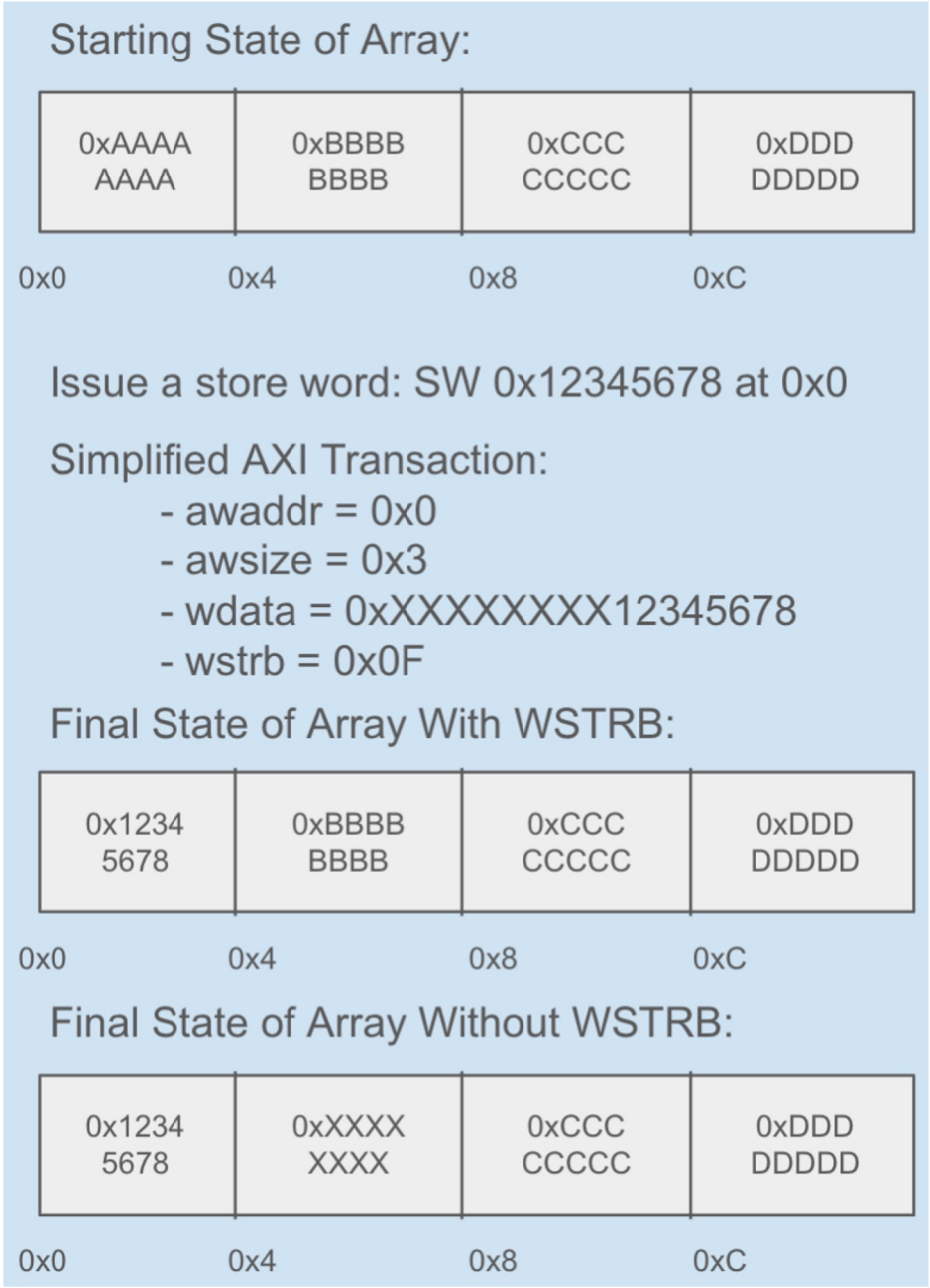


[1] D. Giri et al. CARRV '20

Vortex in ESP: WSTRB

- Vortex integration utilizes ESP's third-party accelerator socket
  - Socket was initially developed for integrating NVDLA [1]
  - Memory access via 64-bit AXI4 primary interface
  - CSR control from 32-bit APB secondary interface

- Added AXI *wstrb* support to the socket for Vortex
  - Goal to avoid significant rework of NoC and memory tile
  - Modified NoC AXI proxies to split 64-bit transactions into subtransactions



[1] D. Giri et al. CARRV '20

Vortex in ESP: Software

- Unlike NVDLA, Vortex does not come with its own Linux drivers
  - Modified ESP's existing accelerator drivers for Vortex-specific register addresses and functions
  - The modified ESP kernel drivers connect to a custom version of Vortex's runtime drivers
  - Lacking virtual memory support, drivers reserve a section of physical memory

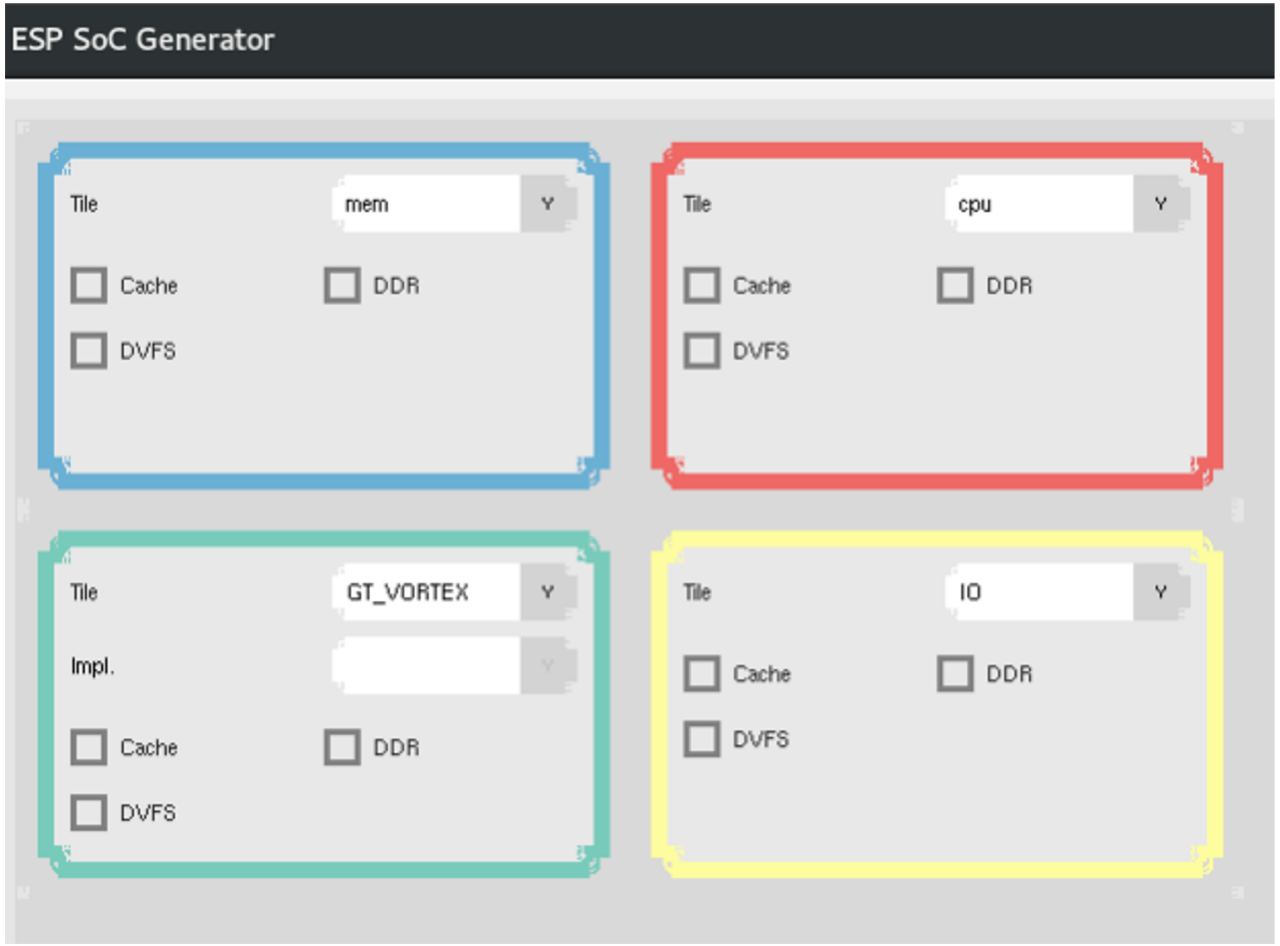
- To confirm functionality, we ran Vortex's provided regression tests [1]
  - Tests were run from the Linux environment running on an ESP SoC containing Vortex
  - OpenCL not currently supported in ESP

```
# /applications/test/gt_vortex_rtl_vortex_stencil3d.exe -k /ap
rtex_kernels/stencil3d.vxbin
open device connection
data type: float
matrix size: 64x64x64
block size: 2x2x2
allocate device memory
A_addr=0xff10000
B_addr=0x10010000
allocate host buffers
upload source buffer0
upload program
upload kernel argument
start device
wait for completion
download destination buffer
verify result
cleanup
PERF: core0: instrs=19865870, cycles=105965001, IPC=0.187476
PERF: core1: instrs=19865870, cycles=105976489, IPC=0.187455
PERF: core2: instrs=19865870, cycles=105986376, IPC=0.187438
PERF: core3: instrs=19865871, cycles=105770970, IPC=0.187820
PERF: instrs=79463481, cycles=105986376, IPC=0.749752
PASSED!
```

[1] Vortex Github Repository

Experimental Evaluation: Setup

- To evaluate performance we developed a floating-point transformer test
  - Test is highly parallelizable and reflective of programs one would run on Vortex
  - All results are for the pythia-70m model with 64 prefill and 32 decode tokens
  - All tests run without CPU side verification
    - Purpose was performance comparison
    - One run done with CPU verification enabled for 2-4-4-L2 to get CPU runtime
- Used ESP SoC with 4 tiles (Ariane CPU, Vortex GPU, memory tile, and I/O tile), no ESP side caches, and 64-bit NoC width



Experimental Evaluation: Vortex Configurations

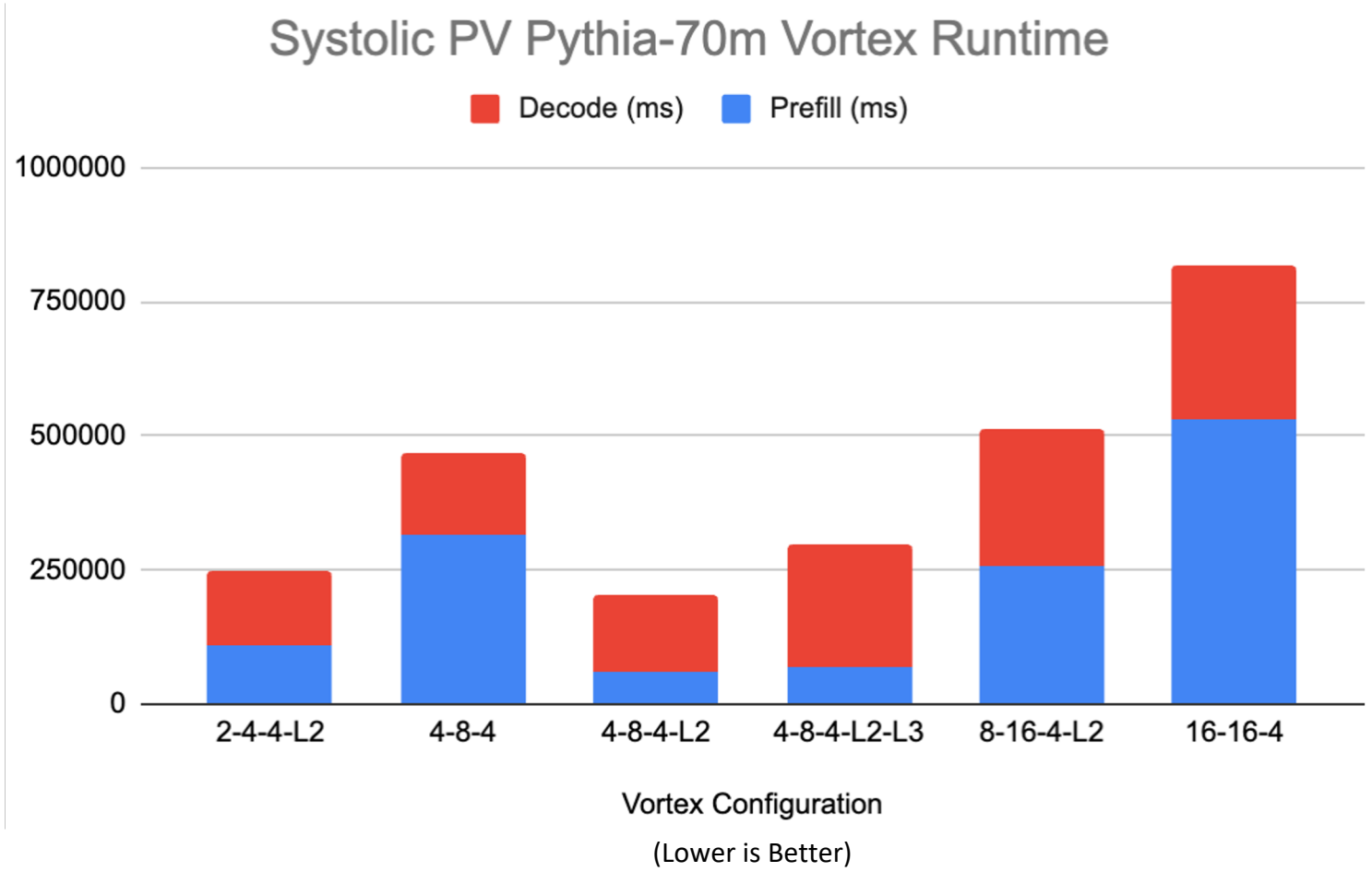
- Ran experiments across various Vortex configurations
  - VCU128 refers to a Virtex UltraScale+ VCU128 board running ESP at 75 MHz
  - Cortana refers to a proFPGA board with four xcvu19p FPGAs running ESP at 100 MHz
  - Cortana used for the larger Vortex configurations that would not fit on VCU128

Run	FPGA	ESP Clock	Cores	Warps	Threads	L2	L3
2-4-4-L2	VCU128	75 MHz	2	4	4	Enabled	Disabled
4-8-4	VCU128	75 MHz	4	8	4	Disabled	Disabled
4-8-4-L2	VCU128	75 MHz	4	8	4	Enabled	Disabled
4-8-4-L2-L3	Cortana	100 MHz	4	8	4	Enabled	Enabled
8-16-4-L2	Cortana	100 MHz	8	16	4	Enabled	Disabled
16-16-4	Cortana	100 MHz	16	16	4	Disabled	Disabled

Experimental Evaluation: Runtime

Run Key: Core-Warps-Threads-Caches

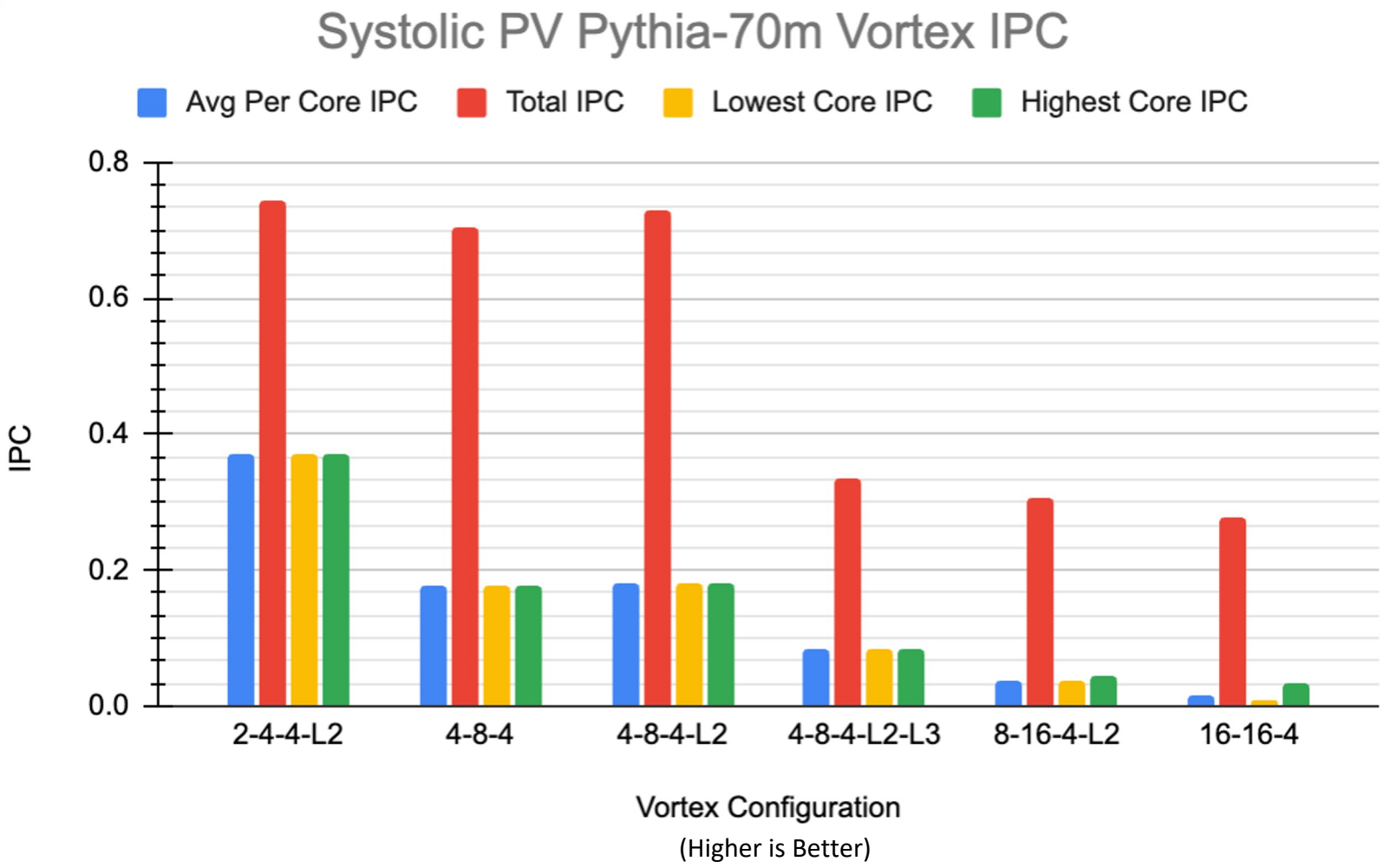
- **Runs:**
  - 4-8-4-L2 took the least time while 16-16-4 took most
- **Core Scaling:**
  - Comparing 4-8-4 and 16-16-4, 16-16-4 performs worse
  - Similar trend between 2-4-4-L2, 4-8-4-L2, and 8-16-4-L2
- **Cache:**
  - Vortex's inbuilt cache improves performance
  - Improvement primarily in compute-bound Prefill
  - Memory-bound decode stays roughly the same



Experimental Evaluation: IPC

Run Key: Core-Warps-Threads-Caches

- **Results Details:**
  - Due to Vortex's performance reporting, IPC numbers only reflect last kernel in workload
  - Results still useful for comparative analysis
- **Test Parallelization:**
  - Difference in IPC across cores is negligible for 2, 4, and 8 core runs
  - Difference becomes significant for 16 core run
  - Results indicate test scales well up to 8 cores but fails parallelize across more, at least for last kernel
- **Per-Core IPC Scaling:**
  - Results do not explain 8-16-4-L2 regression in runtime and total IPC versus 2-4-4-L2 and 4-8-4-L2
  - Also do not explain why total IPC for 2-4-4-L2 and 4-8-4-L2 is similar



Experimental Evaluation: Analysis

- **Performance Bottleneck:**
  - Results indicate bottleneck within ESP causing performance plateauing and eventual regression as more cores are added to Vortex
  - Likely running into a memory bandwidth limitation, particularly ESP's lack of multiple outstanding AXI transactions support

- **Lack of Multiple Outstanding Transactions**
  - Consequently, when a Vortex core issues a request to memory, it must wait for all prior requests and then its request to finish before it can do anything else
  - As we add cores, this memory-request serialization forces cores to spend an increasing portion of time waiting for memory transactions to finish and less on compute
  - Vortex's inbuilt caches can help alleviate this bottleneck but not fully, especially as more cores are added

Experimental Evaluation: Comparison

- Despite bottleneck, Vortex significantly outperforms Ariane CPU for our test workload
- Ariane tested on VCU128 at 75 MHz
- Slowest Vortex configuration, 16-16-4, is almost three times as fast
- Fastest Vortex configuration, 4-8-4-L2, is over 11 times as fast

Conclusion

- Successfully integrated Vortex into ESP as a third-party accelerator
- Performed performance analysis of various Vortex configurations within an ESP SoC
- **Future Improvement:** Results show significant performance to be gained by improving ESP's memory bus, with most immediate improvement being support for multiple outstanding AXI transactions
- Open Source available here. Will be included in next release of ESP

Thank you from the ESP team!

sld.cs.columbia.edu


ColumbiaSld

esp.cs.columbia.edu


sld-columbia/esp


c/ESP-platform

Integration of Open-Source Vortex GPU into ESP

Michael Lippe, Biruk Seyoum, Gabriele Tombesi, Joseph Zuckerman, and Luca P. Carloni

OSCAR 2026