2GRVI Phalanx: *W.I.P.* Towards Kilocore RISC-V® FPGA Accelerators with HBM2 DRAM

⇨ 1776 RV32I / 1332 RV64I cores, 28 MB SRAM, 30 HBM2 DRAM Channels, PCIe, on Xilinx UltraScale+ VU37P / Alveo U280

Jan Gray | Gray Research LLC | Bellevue, WA | jan@fpga.org | http://fpga.org

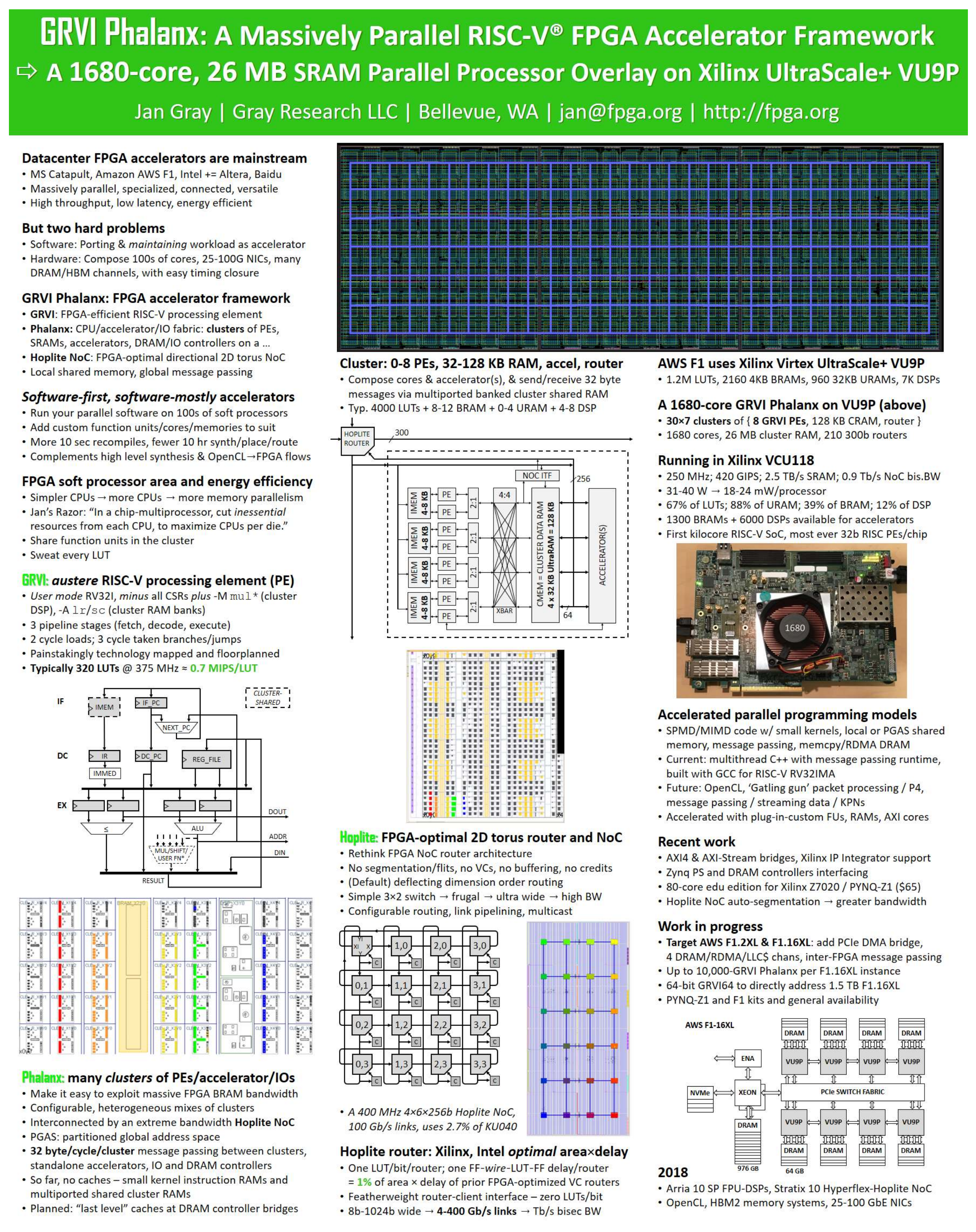
Datacenter FPGA accelerators await our apps
- MS Catapult, Amazon AWS F1, Alibaba, Baidu, Nimbix
- Massively parallel, specialized, connected, versatile
- High throughput, low latency, energy efficient

But two hard problems
- Software: Porting & *maintaining* workload as accelerator
- Hardware: Compose 100s of cores, 100G NICs, many DRAM/HBM channels, with easy timing closure

Mission: GRVI Phalanx FPGA accelerator kit
- **GRVI**: FPGA-efficient RISC-V processing element cores
- **Phalanx**: array of clusters of PEs, SRAMs, accelerators
- **Hoplite NoC**: FPGA-optimal directional 2D torus soft NoC
- Local shared memory, global message passing, PGAS

Software-first, software-mostly accelerators
- Run your C++/OpenCL† kernels on 100s of soft processors
- Add custom function units/cores/memories to suit
- More 10 sec recompiles, fewer 5 hour synth/place/route
- Complements high level synthesis & OpenCL→FPGA flows

2017: V1: 1680 core GRVI Phalanx in a VU9P



GRVI Phalanx: A Massively Parallel RISC-V® FPGA Accelerator Framework
⇨ A 1680-core, 26 MB SRAM Parallel Processor Overlay on Xilinx UltraScale+ VU9P
Jan Gray | Gray Research LLC | Bellevue, WA | jan@fpga.org | http://fpga.org

Datacenter FPGA accelerators are mainstream
- MS Catapult, Amazon AWS F1, Intel += Altera, Baidu
- Massively parallel, specialized, connected, versatile
- High throughput, low latency, energy efficient

But two hard problems
- Software: Porting & *maintaining* workload as accelerator
- Hardware: Compose 100s of cores, 25-100G NICs, many DRAM/HBM channels, with easy timing closure

GRVI Phalanx: FPGA accelerator framework
- **GRVI**: FPGA-efficient RISC-V processing element
- **Phalanx:** CPU/accelerator/IO fabric: **clusters** of PEs, SRAMs, accelerators, DRAM/IO controllers on a ...
- **Hoplite NoC**: FPGA-optimal directional 2D torus NoC
- Local shared memory, global message passing

Software-first, software-mostly accelerators
- Run your parallel software on 100s of soft processors
- Add custom function units/cores/memories to suit
- More 10 sec recompiles, fewer 10 hr synth/place/route
- Complements high level synthesis & OpenCL→FPGA flows

FPGA soft processor area and energy efficiency
- Simpler CPUs → more CPUs → more memory parallelism
- Jan's Razor: "In a chip-multiprocessor, cut *inessential* resources from each CPU, to maximize CPUs per die."
- Share function units in the cluster
- Sweat every LUT

GRVI: *austere* RISC-V processing element (PE)
- *User mode* RV32I, *minus* all CSRs *plus* -M `mul*` (cluster DSP), -A `lr/sc` (cluster RAM banks)
- 3 pipeline stages (fetch, decode, execute)
- 2 cycle loads; 3 cycle taken branches/jumps
- Painstakingly technology mapped and floorplanned
- **Typically 320 LUTs** @ 375 MHz ≈ **0.7 MIPS/LUT**

Phalanx: many *clusters* of PEs/accelerator/IOs
- Make it easy to exploit massive FPGA BRAM bandwidth
- Configurable, heterogeneous mixes of clusters
- Interconnected by an extreme bandwidth **Hoplite NoC**
- PGAS: partitioned global address space
- **32 byte/cycle/cluster** message passing between clusters, standalone accelerators, IO and DRAM controllers
- So far, no caches – small kernel instruction RAMs and multiported shared cluster RAMs
- Planned: "last level" caches at DRAM controller bridges

Cluster: 0-8 PEs, 32-128 KB RAM, accel, router
- Compose cores & accelerator(s), & send/receive 32 byte messages via multiported banked cluster shared RAM
- Typ. 4000 LUTs + 8-12 BRAM + 0-4 URAM + 4-8 DSP

Hoplite: FPGA-optimal 2D torus router and NoC
- Rethink FPGA NoC router architecture
- No segmentation/flits, no VCs, no buffering, no credits
- (Default) deflecting dimension order routing
- Simple 3×2 switch → frugal → ultra wide → high BW
- Configurable routing, link pipelining, multicast
- *A 400 MHz 4×6×256b Hoplite NoC, 100 Gb/s links, uses 2.7% of KU040*

Hoplite router: Xilinx, Intel *optimal* area×delay
- One LUT/bit/router; one FF-*wire*-LUT-FF delay/router = **1%** of area × delay of prior FPGA-optimized VC routers
- Featherweight router-client interface – zero LUTs/bit
- 8b-1024b wide → **4-400 Gb/s links** → Tb/s bisec BW

AWS F1 uses Xilinx Virtex UltraScale+ VU9P
- 1.2M LUTs, 2160 4KB BRAMs, 960 32KB URAMs, 7K DSPs

A 1680-core GRVI Phalanx on VU9P (above)
- **30×7 clusters** of { **8 GRVI PEs**, 128 KB CRAM, router }
- 1680 cores, 26 MB cluster RAM, 210 300b routers

Running in Xilinx VCU118
- 250 MHz; 420 GIPS; 2.5 TB/s SRAM; 0.9 Tb/s NoC bis.BW
- 31-40 W → 18-24 mW/processor
- 67% of LUTs; 88% of URAM; 39% of BRAM; 12% of DSP
- 1300 BRAMs + 6000 DSPs available for accelerators
- First kilocore RISC-V SoC, most ever 32b RISC PEs/chip

Accelerated parallel programming models
- SPMD/MIMD code w/ small kernels, local or PGAS shared memory, message passing, memcpy/RDMA DRAM
- Current: multithread C++ with message passing runtime, built with GCC for RISC-V RV32IMA
- Future: OpenCL, 'Gatling gun' packet processing / P4, message passing / streaming data / KPNs
- Accelerated with plug-in-custom FUs, RAMs, AXI cores

Recent work
- AXI4 & AXI-Stream bridges, Xilinx IP Integrator support
- Zynq PS and DRAM controllers interfacing
- 80-core edu edition for Xilinx Z7020 / PYNQ-Z1 ($65)
- Hoplite NoC auto-segmentation → greater bandwidth

Work in progress
- **Target AWS F1.2XL & F1.16XL**: add PCIe DMA bridge, 4 DRAM/RDMA/LLC$ chans, inter-FPGA message passing
- Up to 10,000-GRVI Phalanx per F1.16XL instance
- 64-bit GRVI64 to directly address 1.5 TB F1.16XL
- PYNQ-Z1 and F1 kits and general availability

2018
- Arria 10 SP FPU-DSPs, Stratix 10 Hyperflex-Hoplite NoC
- OpenCL, HBM2 memory systems, 25-100 GbE NICs

V1 shortcomings
- 32b pointers poor for AWS F1 / big data / OpenCL kernels
- 32b accesses waste half 64b UltraRAM cluster bandwidth
- In-order μarch stalls on loads = 5 cycles in an 8-PE cluster
- Packed, congested, insuff. pipelining = low freq (300 MHz)
- ***DRAM (10s GB/s) uncompetitive vs. GPUs (100s GB/s)***

2019: game changer: FPGAs += HBM2 DRAM
- Xilinx UltraScale+ VU3xP and Intel Stratix 10 MX families
- Xilinx: two HBM2 stacks; 32 AXI-HBM bridge/controllers
- AXIs↔MCs switch: any AXI port can access any controller
- Reads/writes up 32 x 256b x 450 MHz = **460 GBs**

Let's make it easy to use this bandwidth
- TB/s to BRAMs/UltraRAMs, 100s GB/s to HBM2 DRAM

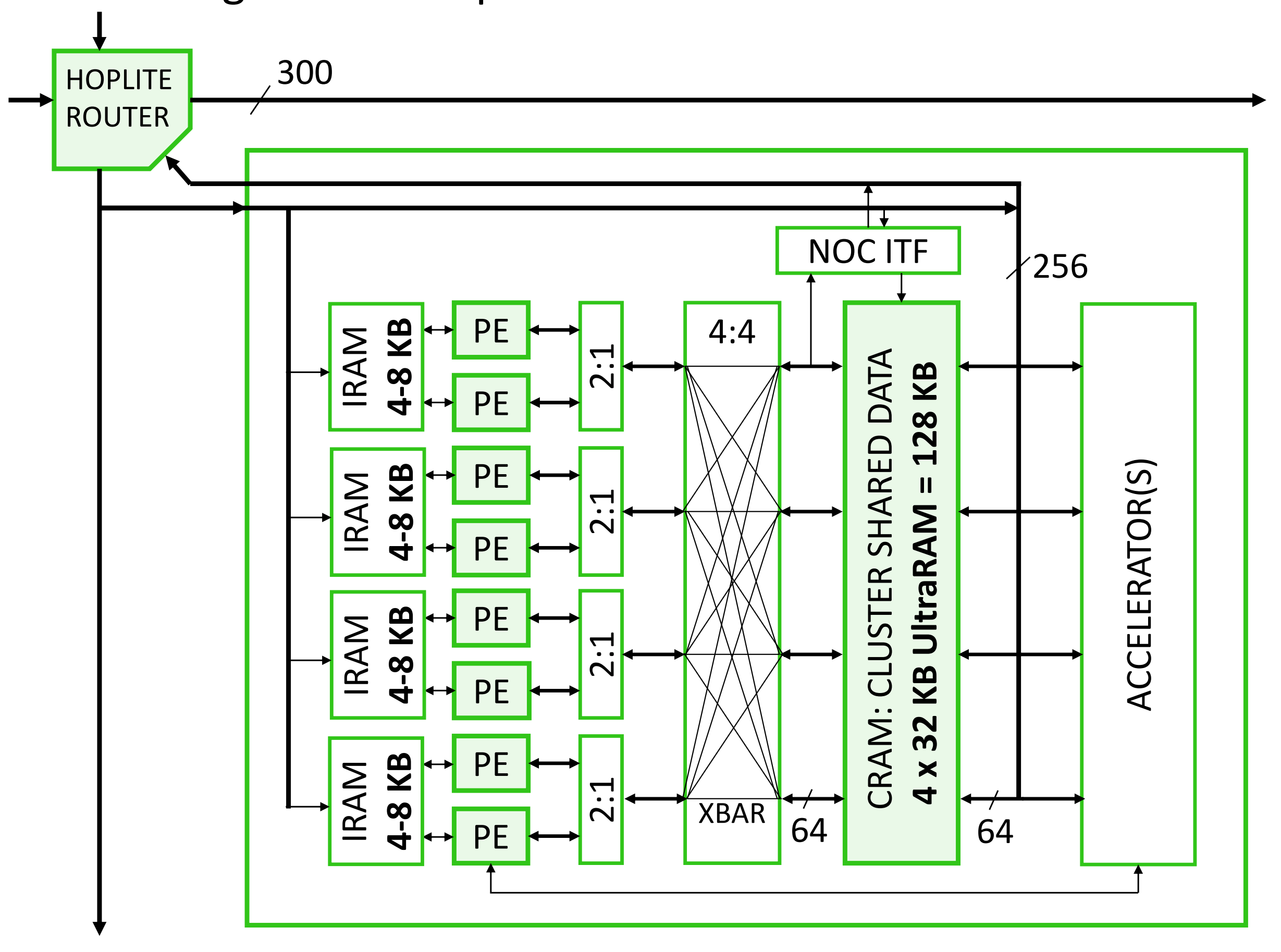
V2: redesign GRVI Phalanx for HBM FPGAs
- New latency tolerant RV64I PEs ✓
- New 64b cluster interconnect, 64b UltraRAM banks ✓
- New 32B/cycle deep pipeline NoC-AXI RDMA bridges ✓
- Add PCIe XDMA mastering (1 AXI-HBM channel) ✓
- Add many more NoC ring columns ✓
- Add Fmax: floorplan, pipelining, FIFOs: target 400+ MHz †
- Add SDAccel-for-RTL shell itf, OpenCL kernel runtime †
- *15 cols x 256b x 400 MHz – peak 192+192 GB/s R+W?* †
- †: *work-in-progress / pending / current plan*

FPGA soft processor area and energy efficiency
- Simpler CPUs → more CPUs → more memory parallelism
- Deconstruct PEs into minimal core plus cluster-shared concurrent FUs: shifts, mul, custom FUs, memory ports (!)

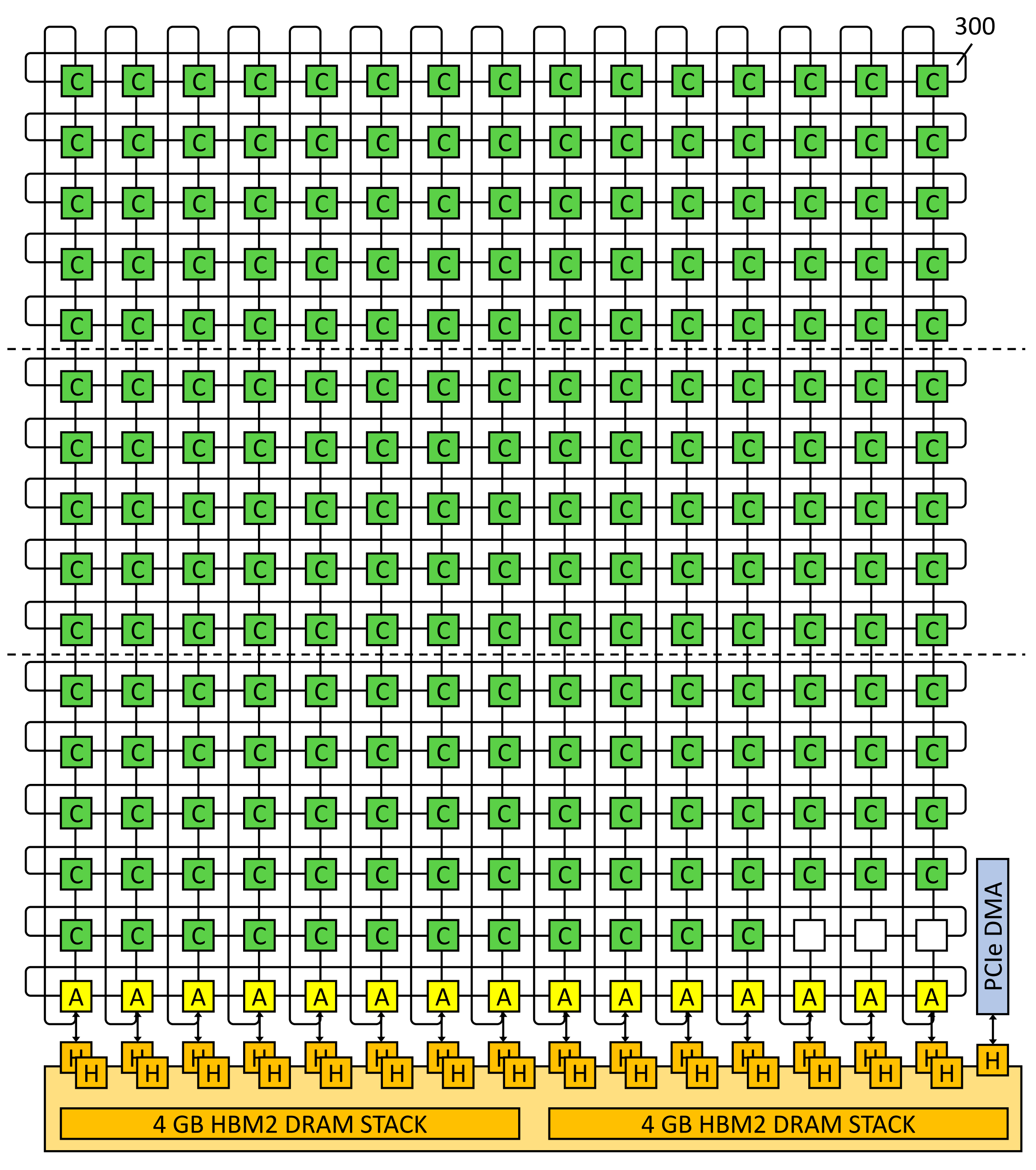
2GRVI – a simple, latency tolerant RV64I PE
- 400 LUTs (sans shared barrel shifter), up to 550 MHz
- Register scoreboard: only stall on use of a busy register
- Out of order retirement; concurrent execution
  - Callee save reg reloads, block copies: now 1 load/cycle
- 2 stg: DC|EX – 3 stg: IF|DC|EX – 4 stg: IF|DC|EX|WB
- 4 stage (superpipelining) has L=2 ALU – CPI ↑25%
- Plan: further tolerate latency with two hardware threads

	GRVI PE	2GRVI PE
Year	2015 Q4	2019 Q2
FPGA Target	20 nm UltraScale	16 nm UltraScale+
RTL	Verilog	System Verilog
ISA	RV32I + mul/lr/sc	RV64I + mul†/lr/sc
Area	320 6-LUTs	400 6-LUTs (- cluster <<)
Fmax / congested	400 / 300 MHz	550 / TBD† MHz
Pipeline stages	2 / 3	2 / 3 / 4 (superpipelined)
Out-of-order retire	-	typical but optional
Two HW threads	-	optional† (+100 LUTs)
Cluster, load interval	5 cycles	1 / cycle
Cluster, load-to-use	5 cycles	6 cycles / 3 thread-cycles†
Cluster, Σ RAM BW	4.8 GB/s (300 MHz)	12.8 GB/s (400† MHz)

Cluster: 0-8 PEs, 128 KB RAM, accel'rs, router
- Compose cores & accelerator(s), & send/receive 32 byte messages via multiported banked cluster shared RAM



SoC: 15x15-3 array of clusters + HBMs + PCIe
- 15 columns x NoC-AXI RDMA bridge + 2xAXI-HBM bridge

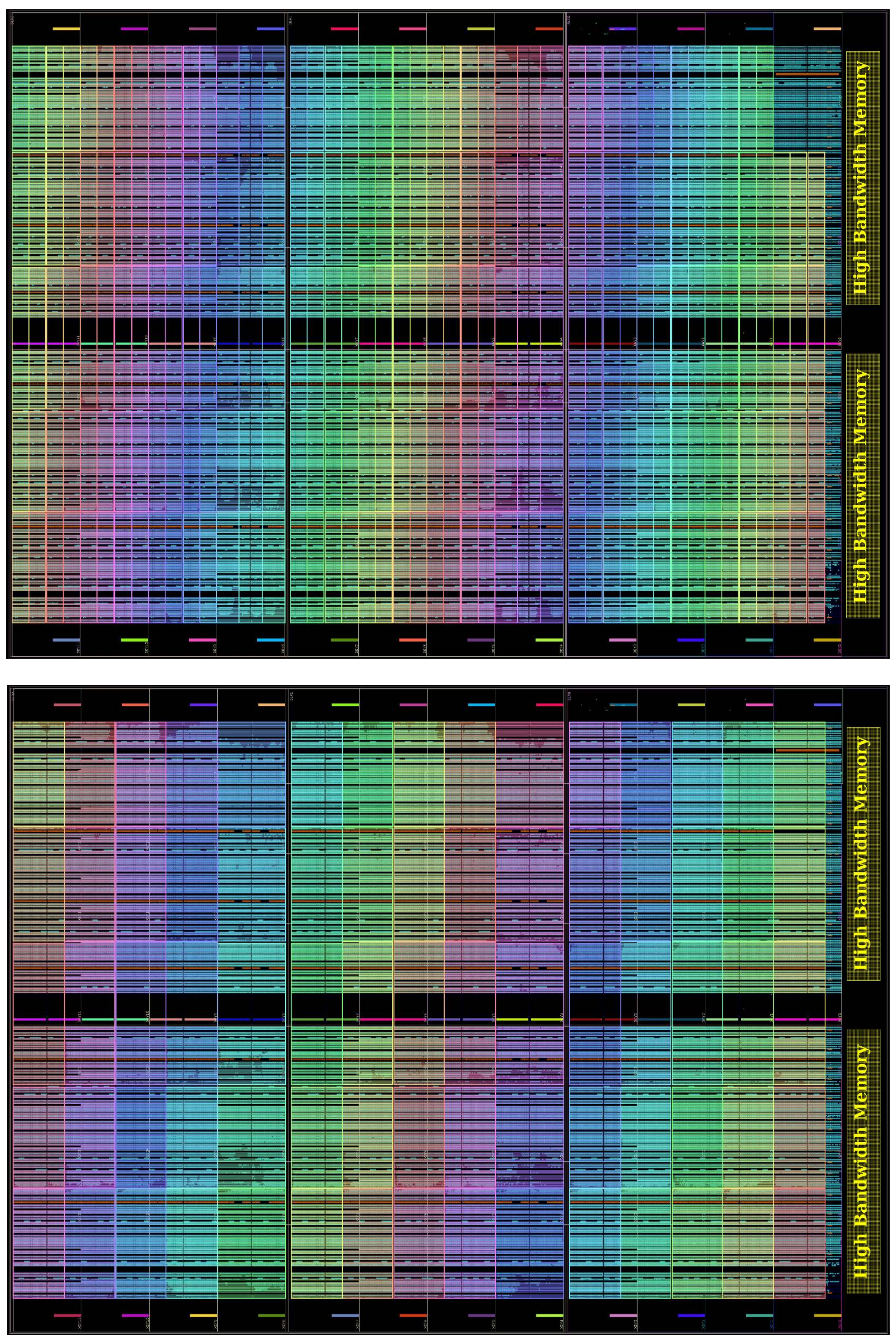


C Cluster { 8 GRVI / 6 2GRVI, 4-8 KB IRAM, 128 KB CRAM, Hoplite router }

A NoC-AXI RDMA bridge { 2 256b AXI R/W req queues, 2 resp queues }

H Two AXI-*switch*-MC-HBM2 bridges, each 256b R/W at up to 450 MHz

Unidirectional Hoplite NoC X-ring rows and Y-ring columns



Kilocore GRVI and 2GRVI HBM2 Phalanxes,
Now Running in a Xilinx VU37P-ES1 in an Alveo U280-ES1

Images: Xilinx.com

- 1776 GRVI @ 300 MHz: **first kilocore RV32I with HBM2**
  - *~60°C. Vivado estimates power of 109W { clks:8 LUTs:26 nets:39 BRAM:5 URAM:6 HBM:16 static:9 } but comparable VU9P Phalanxes measured ~25 mW/PE – stay tuned*
- 1332 2GRVI @ ___ MHz: **first kilocore RV64I with HBM2**

PE↔cluster RAM↔NoC↔AXI↔HBM design
- PEs send **write / read-burst requests** to a NoC-AXI bridge
  - 32B writes/32*n*B reads, split trans, deeply pipelined
- Bridge queues, issues R/W to an AXI-HBM
- Bridge queues, sends **read response** messages over NoC
- Per-PE and per-cluster R/W order preserved
  - Requests/responses on NoC Y-rings → in-order delivery
  - Never queue in Y-rings → request ingress flow control†

NoC-AXI RDMA bridge future research
- Bandwidth studies
- Small LLC$s at NoC-AXI RDMA bridges?
- Software defined access reordering→ SW sets AXI txn IDs
- "Computational HBM" – compute offload at AXI bridges
  - Scatter/gather, add-to-memory, block zero, copy, checksum, reduce, select, regexp, sort, decompress, …

Xilinx VU3xP HBM2 first impressions
- The 32 hard 256b AXI-HBM bridges are easy to design to
- AXI-HBM bridges' switch simplifies SoC interconnect
- Save 100,000s of LUTs vs. soft intercons/DRAM controllers
- Easy DRAM interface timing closure – *wow!*
- Max bandwidth: longer bursts & avoid switch; use NoC?
- Nontrivial to access and transport all that bandwidth

Democratizing HBM memory systems
- HBM technology now accessible to any engineer
- HW: VU3xP in Alveo U280 / U50; *cloud instances?*
- SW: OpenCL via Xilinx SDAccel; 1H20 2GRVI-Phalanx

RISC-V®

GRAY