"If you were plowing a field, which would you rather use: two strong oxen or 1024 chickens?" – Seymour Cray

GRVI Phalanx Update:

Plowing the Cloud with Thousands of RISC-V Chickens

Jan Gray | jan@fpga.org | http://fpga.org

FPGA Datacenter Accelerators Are Here!

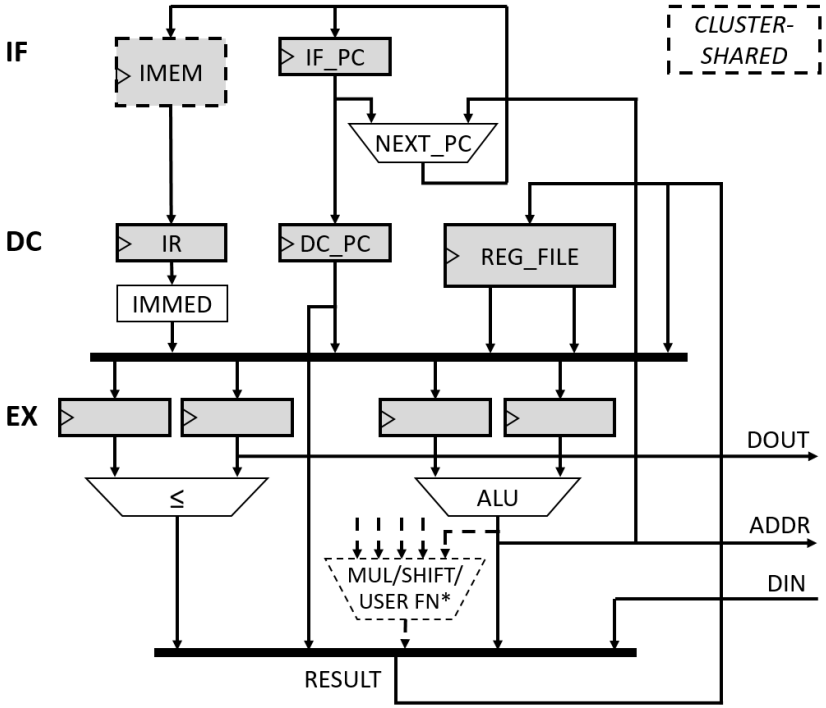
- FPGAs as cloud accelerators
  - Massively parallel, customized, 25/100G networking
  - Genomics, video, analytics, machine learning
  - Catapult v2, Amazon F1, Baidu, Alibaba, Huawei
- Design productivity challenge
  - Software: multithread C++ app $\rightarrow^{?}$ spatial accelerator
  - Hardware: tape-out a complex SoC daily?

GRVI Phalanx Accelerator Kit

- A parallel processor overlay for *software-first* accelerators:
  - Recompile and run on 100s of RISC-V cores
  - = More 5 second recompiles, fewer 5 hour PARs
  - + Add custom FUs and accelerators, as needed
- **GRVI:** FPGA-efficient RISC-V processing element
- **Phalanx:** fabric of clusters of PEs, memories, IOs
- **Hoplite:** 2D torus network on chip

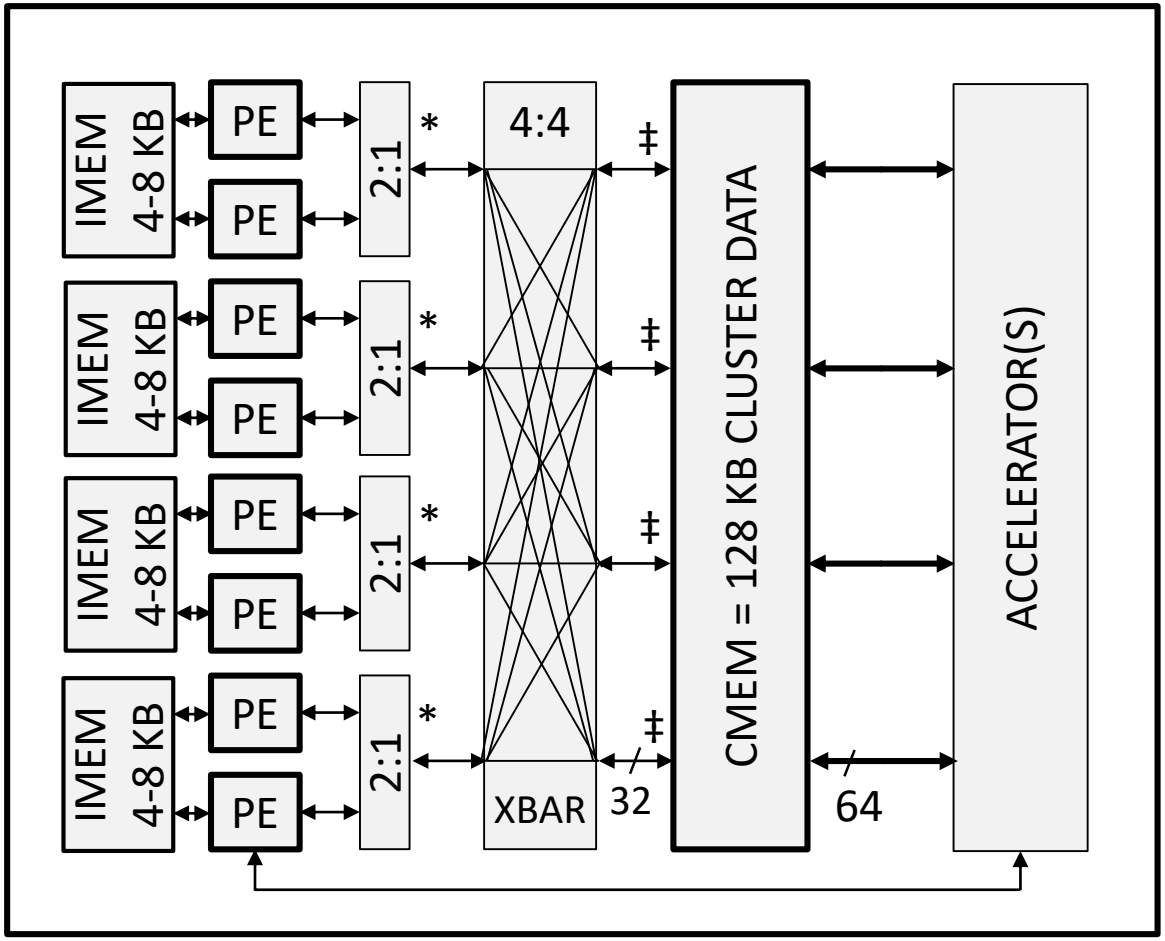
GRVI: Austere RISC-V Processing Element

- Simpler PEs → more PEs → more parallelism!
- GRVI: RV32I - CSRs - exceptions + `mul*` + `lr/sc`



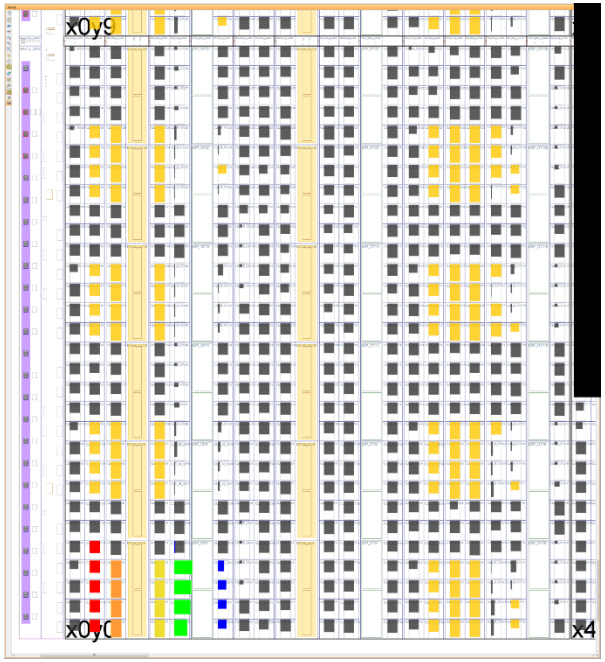
~320 LUTs @ 375 MHz

GRVI Cluster: 8 PEs, Shared RAM, More



* per pair: `lb sb lh sh sll sr* mul*`

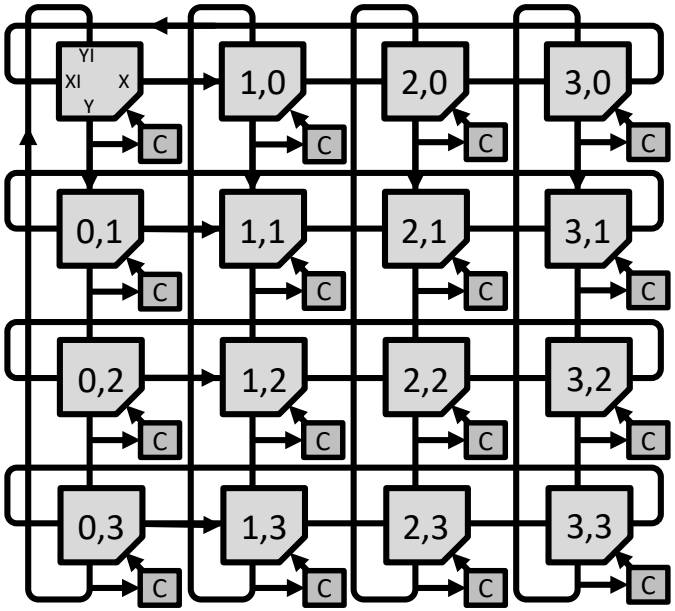
‡ per bank, per PE: `lr sc`

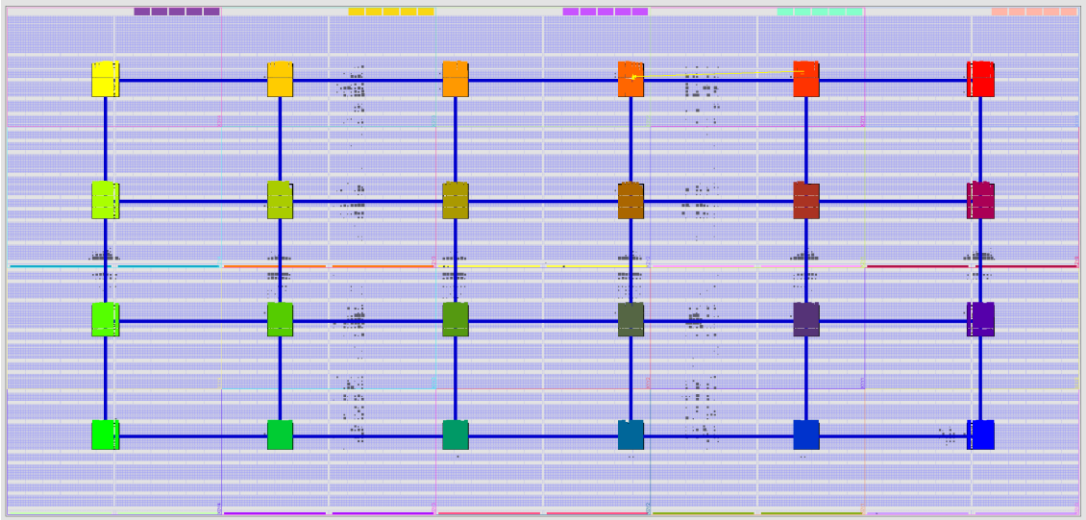


~3500 LUTs

Composition: Message Passing On a NoC

- Hoplite: FPGA-optimal 2D torus NoC router
  - Single flit, unidir rings, deflection routing, multicast

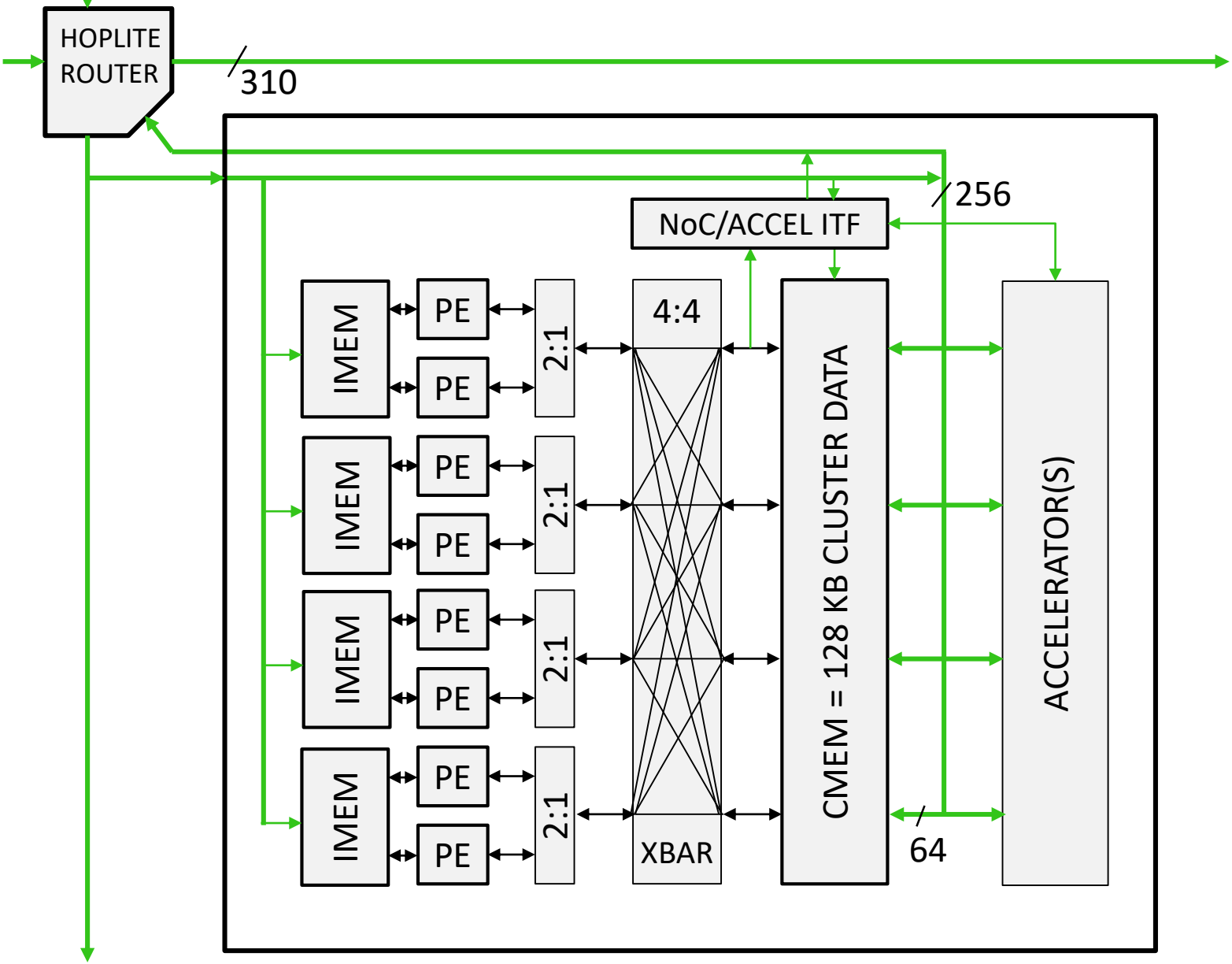




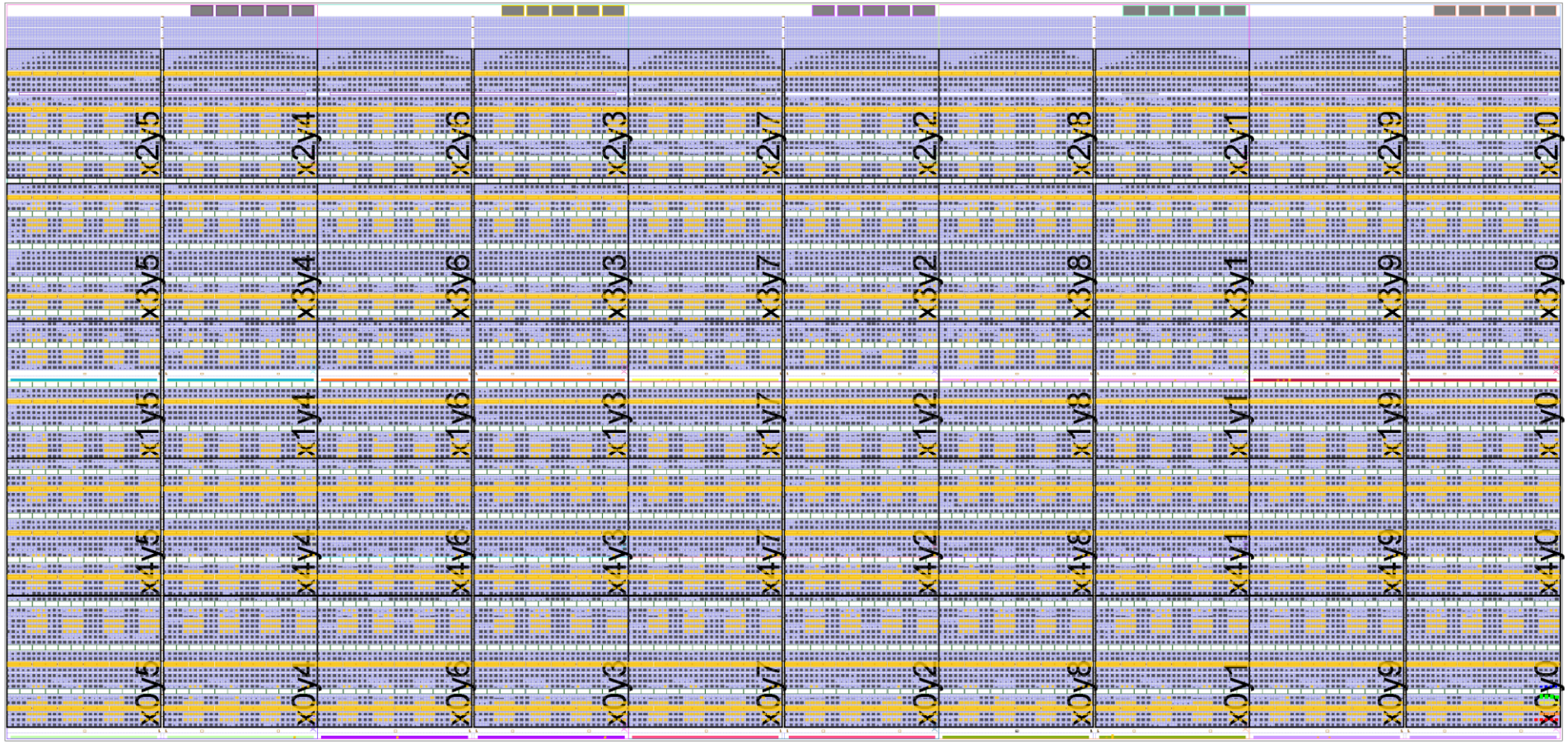
256b @ 400 MHz = 100 Gb/s links

- PGAS: dram:40 -*or*- { dest:20; local: 20 }

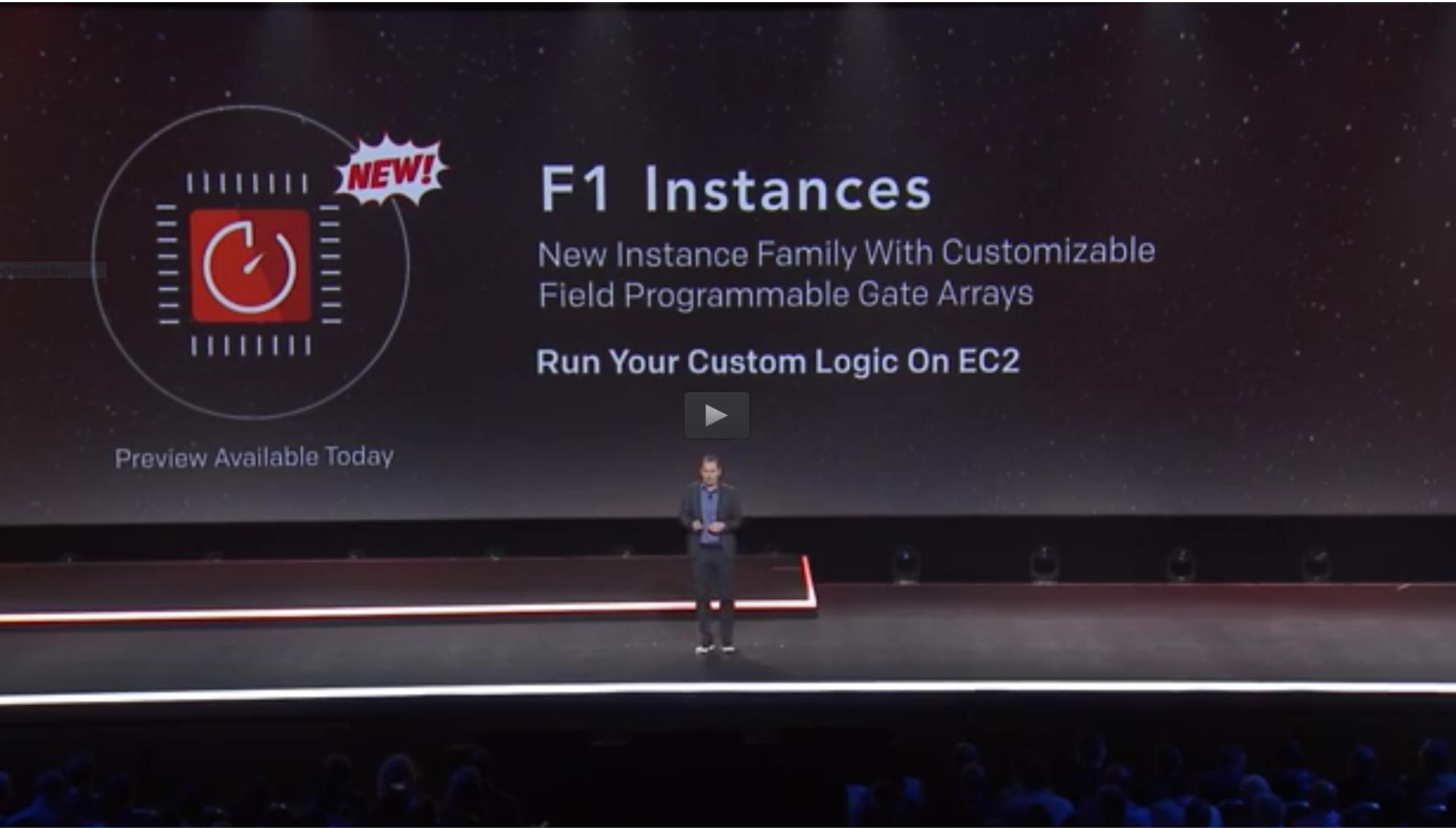
GRVI Cluster + NoC Interconnect



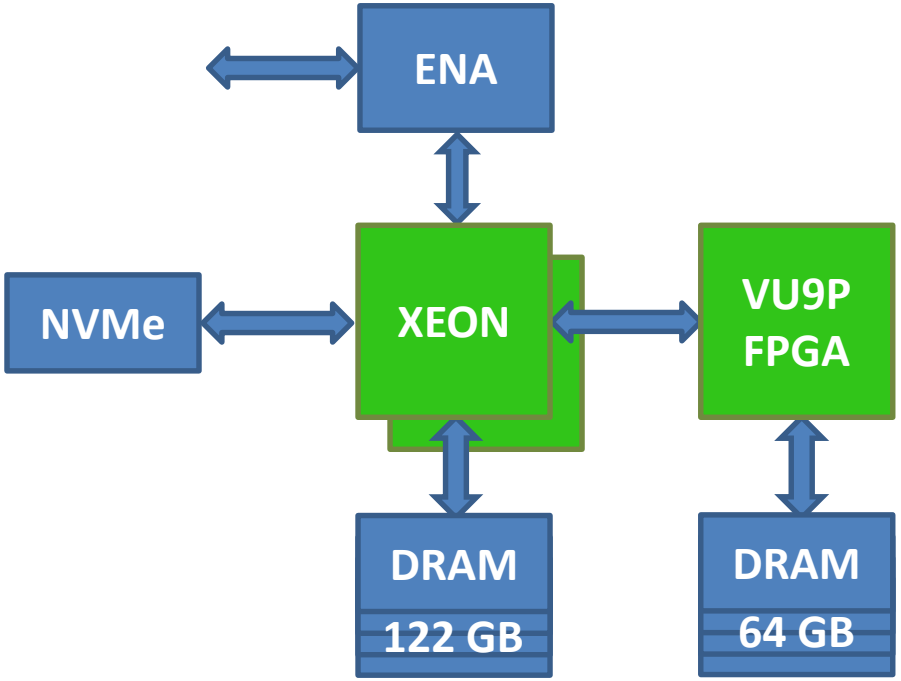
10×5 Clusters × 8 GRVI PEs = 400 GRVI Phalanx (KU040)



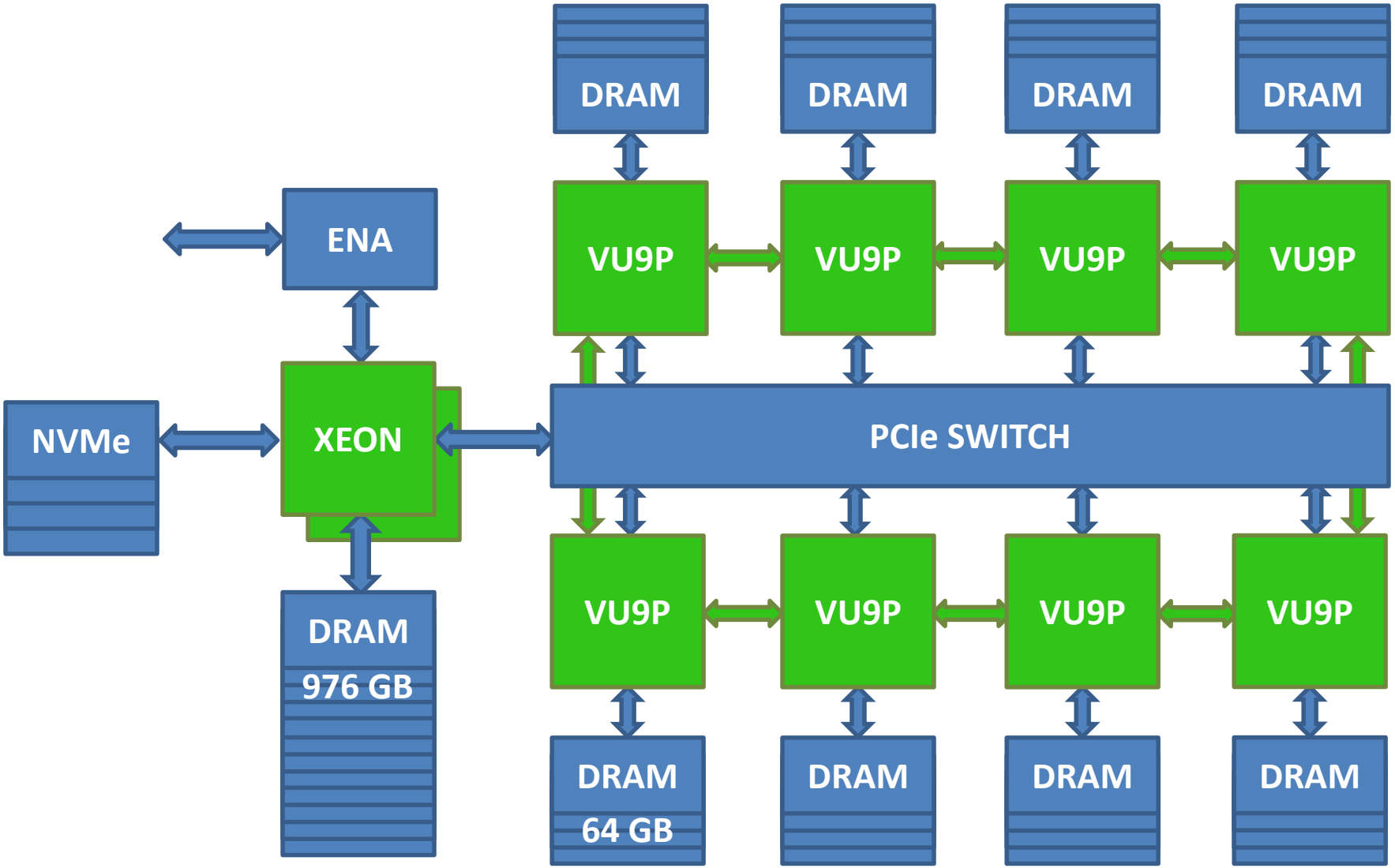
11/30/16: Amazon AWS EC2 F1!



Amazon F1.2xlarge Instance



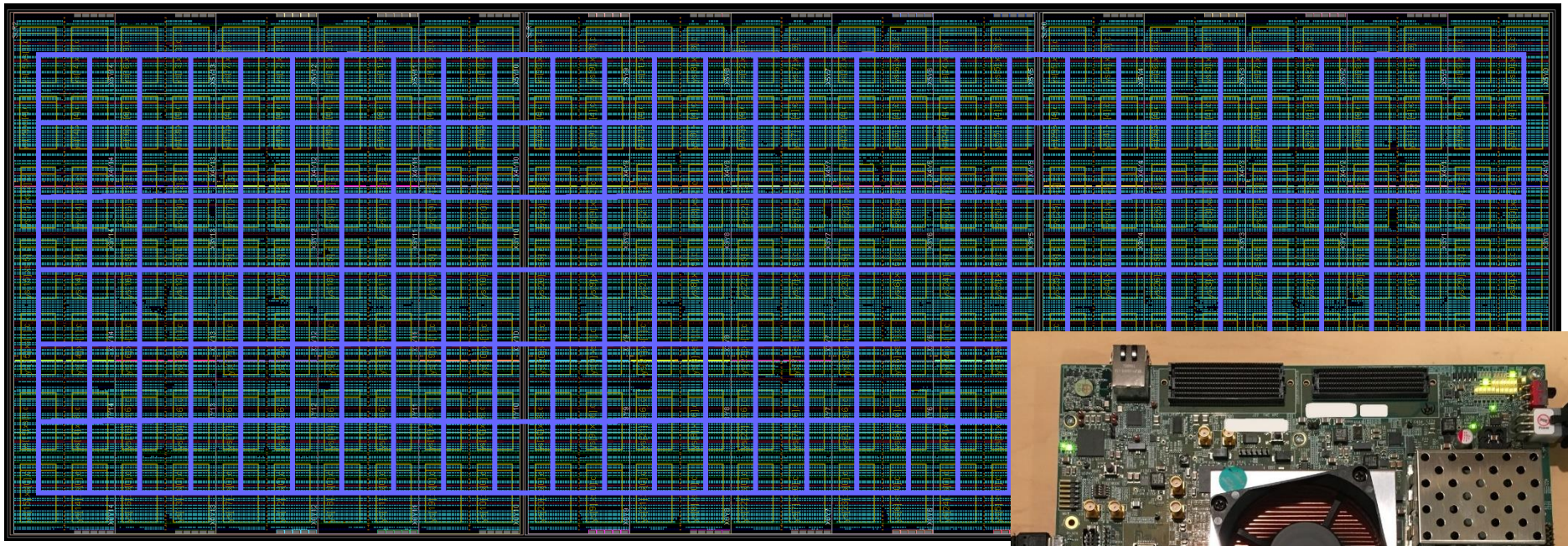
Amazon F1.16xlarge Instance



F1's UltraScale+ VU9P FPGAs

- **1.2M** 6-LUTs
- **2K** 4 KB BRAMs = 8 MB
- **1K** 32 KB URAMs = 30 MB
- **7K** DSPs

1680 RISC-Vs, 26 MB CMEM (VU9P, 12/2016)



- 30×7 × { 8 PEs, 128 KB, router }
- First kilocore RISC-V
- (Then) most 32b RISCs/chip

1680 Core GRVI Phalanx Statistics

Resource	Use	Util. %
Logical nets	3.2 M	-
Routable nets	1.8 M	-
CLB LUTs	**795 K**	**67.2%**
CLB registers	744 K	31.5%
BRAM	840	38.9%
URAM	**840**	**87.5%**
DSP	840	12.3%

```
     VCCINT: (0x40) =  32.38 W, 0.72 V, 44.82 A,  1.26 A, 52.30 A
VCCINTIO_BRAM: (0x48) =   0.30 W, 0.85 V,  0.35 A,  0.25 A,  0.37 A
     VCC1V8: (0x41) =   1.58 W, 1.80 V,  0.88 A,  0.88 A,  0.94 A
    VADJ_FMC: (0x42) =   0.00 W, 1.80 V,  0.00 A,  0.00 A,  0.01 A
     VCC1V2: (0x43) =   0.37 W, 1.20 V,  0.31 A,  0.30 A,  0.31 A
     MGTAVCC: (0x44) =   0.07 W, 0.90 V,  0.08 A,  0.06 A,  0.08 A
     MGTAVTT: (0x45) =   0.07 W, 1.20 V,  0.06 A,  0.01 A,  0.07 A
------------------------------------------------------------------
                    Power Summary
------------------------------------------------------------------
------------------------- Total ---------------------------------
      Logic: 39.82 W
        GTY:  0.15 W
      Total: 39.96 W
```

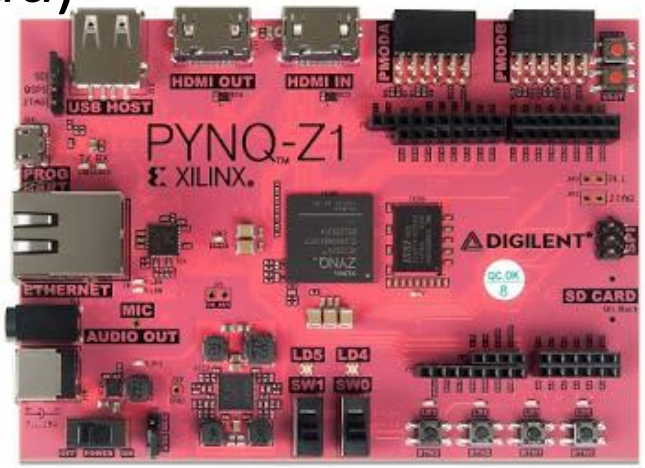
Frequency	250 MHz
Peak MIPS	420 GIPS
CMEM Bandwidth	2.5 TB/s
NoC Bisection BW	900 Gb/s
Power (INA226)	31-40 W
Power/Core	18-24 mW/core
MAX VCU118 Temp	44C

Vivado	2016.4 / ES1
Max RAM use	~32 GB
Flat build time	11 hours
Tools bugs	0

>1000 BRAMs + 6000 DSPs available for accelerators

Towards a GRVI Phalanx SDK

- Bridge Phalanx and AXI4 system interfaces
  - Message passing bridge to host CPUs (x86 or ARM)
  - R/W req/resp messages + RDMA bridge to DRAM
- Two SDK hardware targets
  - 8-80-core PYNQ (Z7020) ($65 edu)
  - 800-10K-core AWS F1 ($-$$/hr)



PYNQ-Z1 Demo Video

GRVI Phalanx on Pynq-Z1

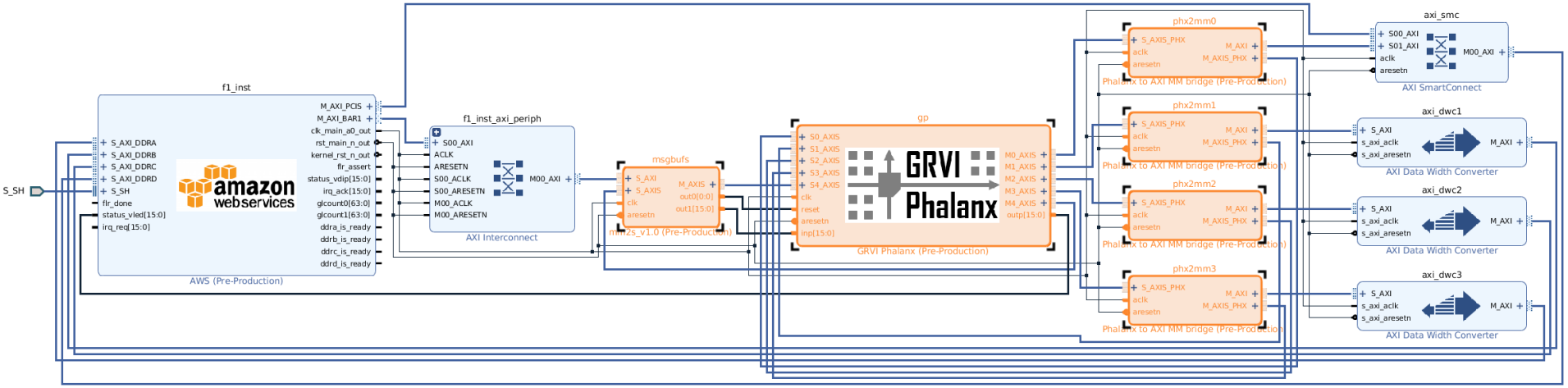
88 GRVI RISC-V cores + 352 KB SRAM
+ 32 KB Frame Buffer

(work in progress)

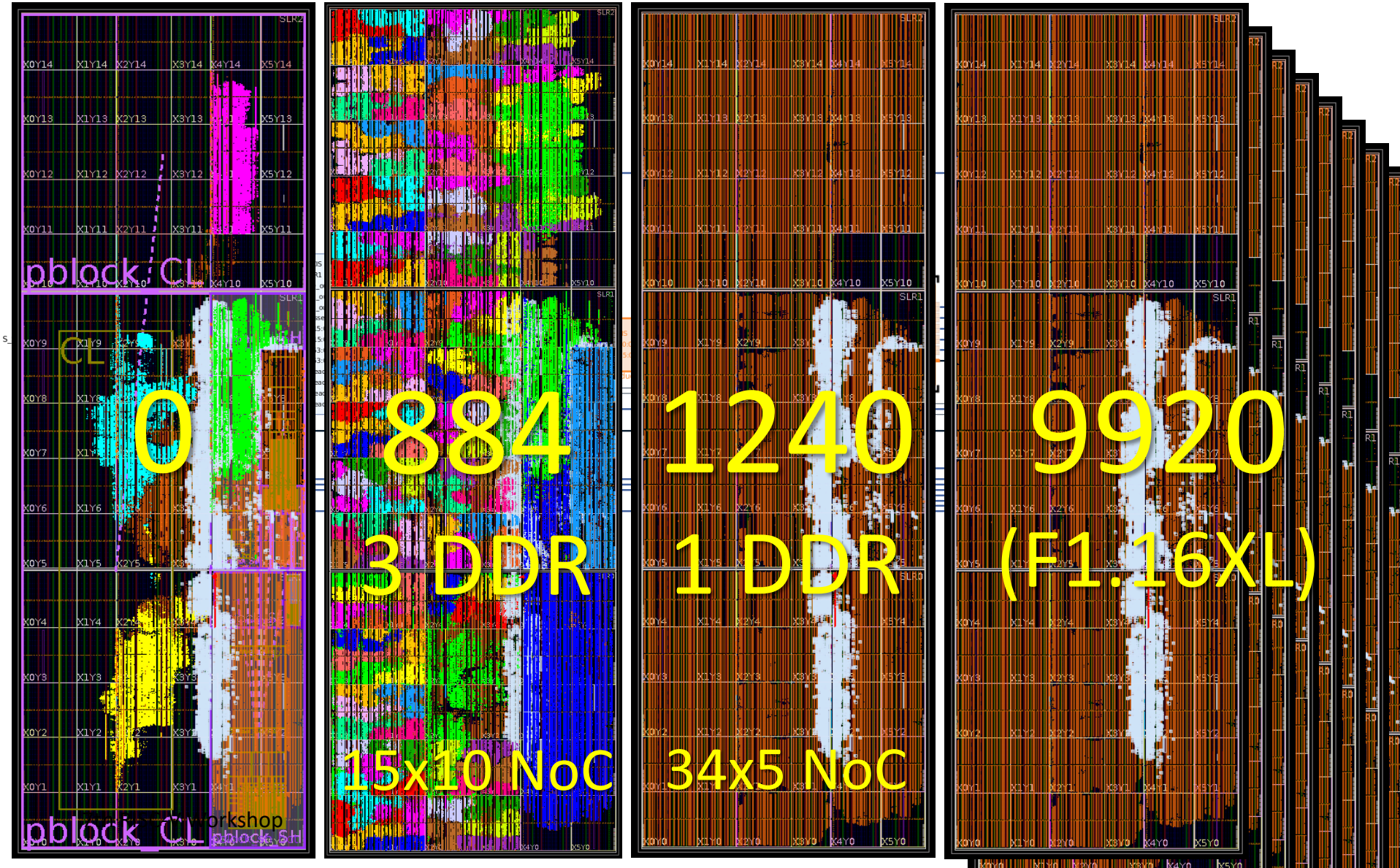
Jan Gray
jan@fpga.org

http://fpga.org/grvi-phalanx

GRVI Phalanx on AWS F1



GRVI Phalanx on AWS F1



SDK Programming Model #1: OpenCL

- Current: bare metal multithread C++, messages
- W.I.P: OpenCL, using new 'SDAccel for RTL' for F1
  - Flat data parallel kernels, NDRange of work items
  - Read ‖ local compute ‖ write back to DRAM
- Host: SDA OpenCL to stage buffers, invoke kernels
- Phalanx: SDAccel shell & DRAMs ⇔ GRVI Phalanx
  - Workgroup = cluster; workitem = PE
  - RISC-V kernel .text/.data + input buffers passed in DRAM

SDK Programming Model #1: OpenCL

- Current: bare metal r
- W.I.P: OpenCL, using
  - Flat data parallel kerr
  - Read ‖ local comput
- Host: SDA OpenCL to
- Phalanx: SDAccel she
  - Workgroup = cluster;
  - RISC-V kernel .text/.d

```
kernel
void vector_add(
    global_ptr<int> g_a,
    global_ptr<int> g_b,
    global_ptr<int> g_sum,
    const unsigned n_elts)
{
    local align int a[MAX_ELTS];
    local align int b[MAX_ELTS];
    local align int sum[MAX_ELTS];

    int iloc = get_local_id(0) * n_elts;
    int iglb = (get_group_id(0)*get_local_size(0)
                + get_local_id(0)) * n_elts;
    int size = n_elts * sizeof(int);

    copy(a + iloc, g_a + iglb, size);
    copy(b + iloc, g_b + iglb, size);
    barrier(CLK_LOCAL_MEM_FENCE);

    for (int i = 0; i < n_elts; ++i)
        sum[i] = a[i] + b[i];
    barrier(CLK_LOCAL_MEM_FENCE);

    copy(g_sum + iglb, sum + iloc, size);
}
```

Plan

- Add SDAccel shell integration
- Complete F1.2XL and F1.16XL ports
- GRVI Phalanx SDK for F1 and PYNQ
  - F1: AMI+AFI in AWS Marketplace
  - PYNQ: Jupyter Python notebooks, bitstreams
  - Tools, specs, libs, tests, examples
- Ideation
  - GRVI64
  - Latency tolerant GRVI with OoO retirement
  - Wider, faster cluster RAM + vector<fp8> dot product accel = 250 GFLOPS/cluster, 120 TFLOPS/F1.16XL

In Summary

- Make it easier to access FPGA spatial parallelism
  - Value proposition unproven, awaits workloads
- Competitive perf demands a frugal design
- SDK coming: 8-80-800-10,000 core designs
- Enabled by the excellent RISC-V ecosystem

Thank you!