2GRVI Phalanx: A 1332-core RISC-V RV64I Processor Cluster Array with an HBM2 High Bandwidth Memory System, and an OpenCL-like Programming Model, In a Xilinx VU37P FPGA **[WIP Report]**

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

2GRVI (and its predecessor, GRVI) are FPGA-efficient 64b (resp. 32b) RISC-V processing element cores. Phalanx is a parallel processor and accelerator array overlay framework. Groups of PEs and accelerator cores form shared memory compute clusters. Clusters, DRAM, NICs and other I/O controllers communicate by message passing on an FPGA-optimal Hoplite torus soft NoC. This extended abstract summarizes work-in-progress to redesign the 2017 GRVI Phalanx to take advantage of new Xilinx FPGAs with 460 GB/s dual stack HBM2 DRAM-in-package, and to provide a familiar parallel programming experience via an OpenCL-like programming model and tools. The new system is the first kilocore RV64I SoC and the first RISC-V multiprocessor with an HBM2 memory system.

I. Introduction: GRVI Phalanx

Complementing server CPUs, FPGA accelerators promise higher throughput, lower latency, and lower energy [1]. But it is challenging to move working software into an accelerator, and to maintain it as the code evolves. RTL, High Level Synthesis, and even OpenCL-to-FPGA tools have serious shortcomings, such as high porting effort and multi-hour builds. Another challenge is system design and timing closure of a complex SOC comprising many cores and fast I/O and DRAM interfaces. Few organizations succeed with FPGA accelerator development.

The present work is *v2* of GRVI Phalanx [2,3], a parallel processor *overlay* to simplify accelerator development. It supports a software-first approach. A C++ or OpenCL workload runs on the soft processors in the overlay. Then custom instructions, accelerator cores, or memories may be introduced to speed up bottlenecks. Most design iterations are quick recompiles; development is more like performance engineering.

GRVI Phalanx, the first kilocore 32b RISC SoC, showed that a frugal NoC and PEs, a replicated cluster tile architecture, and the RISC-V ecosystem, together enable low complexity, high throughput compute accelerators. This *v1* hardware had some shortcomings though: 1) modern big data OpenCL kernels require PEs with 64b pointers; 2) GRVI's 32b architecture squanders half the bandwidth of the cluster's 64b UltraRAMs; and 3) in a very congested FPGA its Fmax is just 300 MHz.

Perhaps the least competitive aspect of GRVI Phalanx performance (and of most FPGA "accelerators") is poor DRAM bandwidth. In accelerator cards and in AWS F1, GRVI Phalanx is limited to four 64b DDR4 DRAM channels – far less bandwidth than that of inexpensive GPUs, limiting the range of interesting workloads.

II. Version Two: Introducing 2GRVI Phalanx

2GRVI Phalanx is tackling these and other issues. It is redesigned for Xilinx UltraScale+ HBM2 devices [6] such as the VU37P FPGA, with two stacks (8 GB) of HBM2 DRAM and 32 256b hardened AXI-HBM controllers, R/W up to 460 GB/s.

Fig. 1 presents an (empty) VU37P floorplan, rotated 90 degrees. It spans three SLR dies, SLR0-1-2, and includes about 1.3M 6-LUTs (not shown), 2000 36 Kb BRAMs (yellow), 960 288Kb UltraRAMs (green), 9000 DSPs (blue), 21,000 SLL inter-SLR interconnect nets per SLR-pair (white), and *most notably* 32 256b @ 450 MHz AXI-HBM bridges [7] (magenta) across the base of SLR die 0. These bridges incorporate a switch between the AXI ports and the HBM memory controllers, optionally enabling any AXI master to access memory behind any of the 16 memory channels (32 HBM2 pseudochannels).

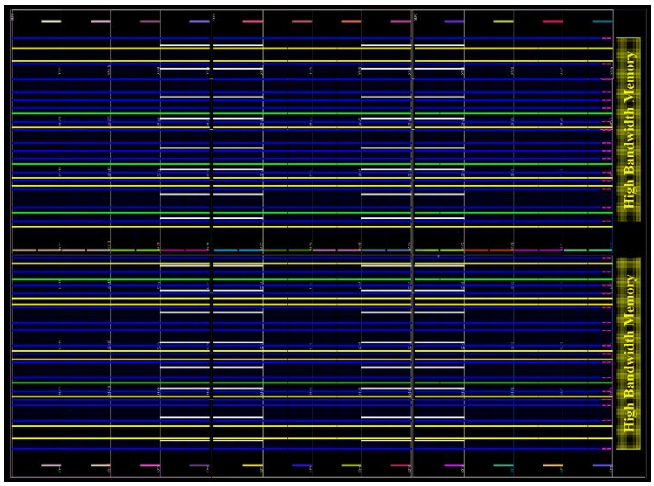

Figure 1: Xilinx VU37P with HBM2 Device Floorplan

But it is challenging to move data at up to 3.7 Tb/s to/from the AXI-HBM controllers at the base of the FPGA, from/to the various cores across the length and breadth of the device.

A very fast, very wide soft NoC is the way forward, although at FPGA SoC frequencies (300-500 MHz) this requires many thousands of northbound and southbound nets. (The faster the NoC clock, the fewer nets required.) Then other clock constraints must be considered. The GRVI PEs are too slow; the NoC and UltraRAMs can run at 600 MHz, but the AXI-HBM controllers' Fmax is 450 MHz. To avoid CDCs we are targeting and tuning the implementation to run each component at 450 MHz. At 450 MHz, a 15x15x256b Hoplite NoC [5] will carry ~200 GB/s of read data and ~200 GB/s of write data between the HBM controllers and any FPGA clusters or I/O controllers. If not yet full peak HBM2 bandwidth, it is nevertheless a leap ahead for RISC-V multiprocessors, and FPGA accelerators.

III. The 2GRVI RISC-V RV64I Core

At just 320 LUTs/PE, GRVI still has leading soft processor throughput÷area, but its limitations include its 32-bit width, its 300-400 MHz Fmax, and its simple scalar RISC in-order loads: *in an 8 PE GRVI cluster setting* a load takes five cycles there and back through the interconnect. This is awful in a function epilog, reloading *n* callee save registers, five cycles each.

2GRVI is a new core design with lower area/bit-width, RV64I/RV32I support, 550 MHz Fmax, and much better latency tolerance. Using a busy-register scoreboard, loads do not stall until/unless subsequent use of a still busy register. In an epilog register reload, or an unrolled block copy loop, 2GRVI issues one load each cycle. The same mechanism enables concurrent execution and out-of-order completion of long latency FUs, using a to-be-proposed open Custom Function Unit interface.

As with GRVI, 2GRVI is carefully technology mapped and floorplanned for Xilinx 6-LUT FPGAs. It embraces Jan's Razor [4]: "In a chip multiprocessor design, strive to leave out all but the minimal kernel set of features from each processing element, so as to maximize processing elements per die." This leads to a deconstructed PE architecture where functions such as shifts, multiplies, even byte-aligning load/store memory ports, are factored out of the PE core such that multiple PEs share those occasional-use resources. This gets the 64-bit 2GRVI PE core down to just 400 LUTs; the total area overhead of the PE and its

share of a six PE cluster, function units, cluster interconnect, and 300b Hoplite router, is about 700 LUTs.

For highest Fmax of 550 MHz, 2GRVI implements a 4-stage pipeline which still issues up to one instruction per cycle, but which has a minimum ALU operation latency of two cycles. This configuration incurs result-use stalls if an instruction consumes the result of the prior instruction.

To mitigate result-use stalls, and 4 cycle taken branches, 2GRVI may incorporate optional two-way HW multithreading. This would cost ~100 LUTs, including 80 LUTs to double the physical register file to 64x64b. It remains to be seen if this actually improves PE throughput ÷ area.

Table 1 compares and contrasts the two PE cores.

	GRVI	2GRVI
Year	2015 Q4	2019 Q2
Target	20 nm UltraScale	16nm UltraScale+
RTL	Verilog	System Verilog
ISA	RV32I +mul/lr/sc	RV64I +mul/lr/src
Area	320 LUTs	400 LUTs (sans bshift)
Fmax/congested	400/300 MHz	550/TBD MHz
Pipeline stages	2/3	2/3/4 (super-pipe'd)
Out-of-order retire	-	typical but optional
2 hardware threads	-	optional
Cluster: load II	5 cycles	1 cycle
Cluster: load to use	5 cycles	4/5/6 cycles
Cluster, peak BW	4.8 GB/s @ 300 MHz	12.8 GB/s @ 400 MHz

Table 1: 32-bit GRVI vs. new 64-bit 2GRVI

In all, the extra bit-width, load latency tolerance, and higher Fmax affords 2GRVI clusters over twice the peak bandwidth to the cluster RAMs vs. GRVI PEs in GRVI Phalanx, using the same resources (LUT RAMs, block RAMs, and UltraRAMs).

The 2GRVI cluster retains the same general design as the prior GRVI cluster: 0-8 PEs, 4-8 KB kernel instruction RAMs, 128 KB of shared banked address-interleaved cluster shared memory, pipelined crossbar interconnect, network interface, and 32B/cycle Hoplite router. See Fig. 2. Here all interconnect datapaths must grow from 32- to 64-bits wide.

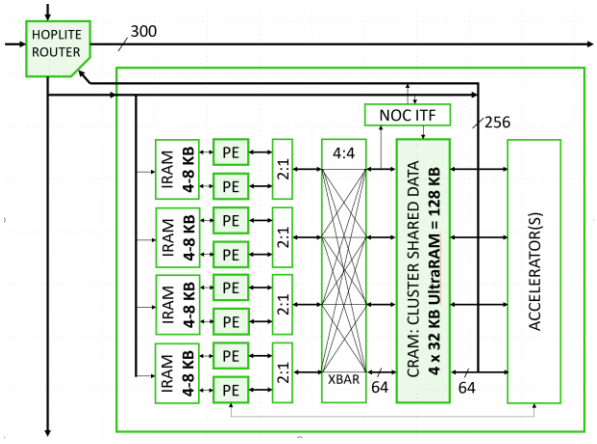

Figure 2: GRVI/2GRVI Cluster Architecture

The device pitch of the UltraRAMs sets the resource budget for the cluster and hence the number of processors per cluster. In the VU37P, there are 960 URAMS in five columns of 192 URAMs. The present design assigns four URAMs per cluster, up to 240 clusters in all, and five approximately equally sized clusters span the die horizontally. The cluster LUT budget is about 1.3M / 240 = 5400 6-LUTs, enough to implement PE clusters with eight 32-bit GRVI PEs or six 64-bit 2GRVI PEs.

IV. Phalanx Redesign for HBM2 Memory

The redesign for VU37P entails: 1) modifying the NoC's X rings x Y rings topology to include at least twice as many die-spanning vertical Y rings; 2) designing a wide, deeply pipelined NoC-AXI RDMA bridge that can sustain write requests and burst read requests on back to back clock cycles, 256 bits per bridge per cycle, every cycle; and 3) increasing the Fmax of every element of the SoC from 300 MHz to 450 MHz. At present, the first two have been achieved, and timing optimization work continues in the new RTL codebase.

The updated SoC interconnect is a chip-spanning 16 row by 15 column Hoplite NoC. It interconnects a 15x15 (-3) array of GRVI/2GRVI clusters, and a row of 15 NoC-AXI RDMA bridges. See Fig. 3.

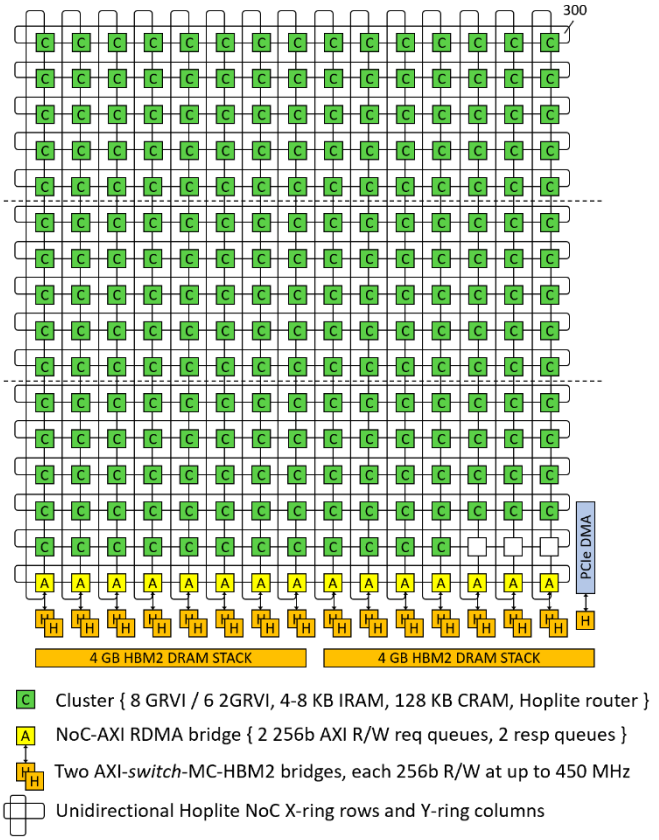

Figure 3: 2GRVI Phalanx SoC Architecture

The NoC carries 312-bit one-flit messages, each message comprising: 12 bit Hoplite message header, 4 bit transaction ID, 40 bit address, and 256 bits of data. Messages may carry data from an agent (i.e. PE or accelerator) in one cluster to an agent in another cluster, or may carry 32B DRAM write requests, 32*n*B burst read requests, or a stream of 32B read responses, from an agent to/from a NoC-AXI RDMA bridge located alongside its AXI-HBM bridge(s) at the bottom of SLR0.

A horizontal bisection of the 15 column NoC spans 9,360 nets (about 45% of the possible 21,000 SLLs that cross each pair of Super-Logic Region dies) and the NoC's bisection bandwidth is 15x2x256x300 MHz = 2.3 Tb/s and at 450 MHz *will be* about 3.5 Tb/s. The NoC uses ~79,000 LUTs, or 6% of the VU37P LUTs.

The SoC also incorporates a PCI-Express gen3x1 XDMA controller. This uses ~15,000 LUTs; three PE clusters from the first row of PE clusters are depopulated to make room.

V. PE↔Cluster RAM↔NoC↔AXI↔HBM Design

HBM2 DRAM transactions are a minimum of 32B in size, and so for performance, this is reflected in the Phalanx C++ and OpenCL-like programming models, which currently exclusively expose the shared memory as a 32B-block device accessed by through a block copy API. This function performs a block write as a stream of 32B block write request messages; and a block read as stream of 32B-to-512B burst read request messages. A PE writes the write data to be written, or the burst read RDMA command, as a 32B message in its cluster RAM, then sends that message over a NoC Y ring down to that cluster's column's NoC-AXI RDMA bridge.

The bridge accepts one write or burst-read request message per cycle; these are queued in a 300b-wide FIFO. The request at the head of the FIFO is issued to the hard AXI-HBM bridge at a rate of up to one 32B (256b) AXI transaction per cycle (AXI AW+W channels or AR channel). Pending read request response metadata move to a read response FIFO. Later (80-100 cycles later) as the AXI interface responds with a read response (AXI R channel), the 1-16 beats of 32B of data are sent, as 32B read response messages, back on the NoC Y ring to the requester (or to wherever the requester specified the RDMA read response to be sent, including multicast over the whole Phalanx).

By sending write and burst-read requests over a Y ring column of the NoC, message ordering is preserved, as is per-PE and per-cluster views of read and write memory access ordering. When necessary, simple backpressure flow control, from a NoC-AXI bridge request FIFO 'nearing capacity high water mark' signal to the clusters on its Y-ring, suspends memory access request message sends at the source(s).

VI. Status

A 1776 PE 32-bit GRVI Phalanx now runs in a VU37P-ES1 in a Xilinx Alveo U280-ES1 card. It has the topology of Fig. 3, with 222 clusters of eight GRVI PEs. See Fig. 4.

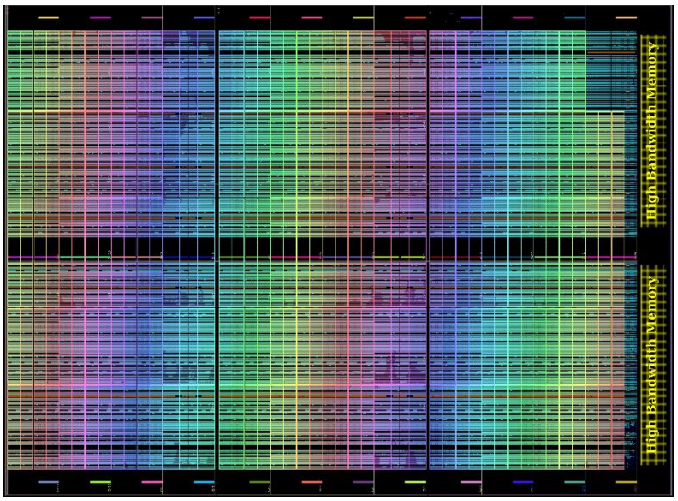

Figure 4: 1776 PE 32-bit GRVI Phalanx with HBM

A 1332 PE 64-bit 2GRVI Phalanx also runs in a VU37P-ES1. It too also has the topology of Fig. 3, with 222 clusters of PEs, but here clusters are six 2GRVI PEs, not eight, due to the larger PEs, and the substantially larger 64-bit local cluster interconnect. See Fig. 5.

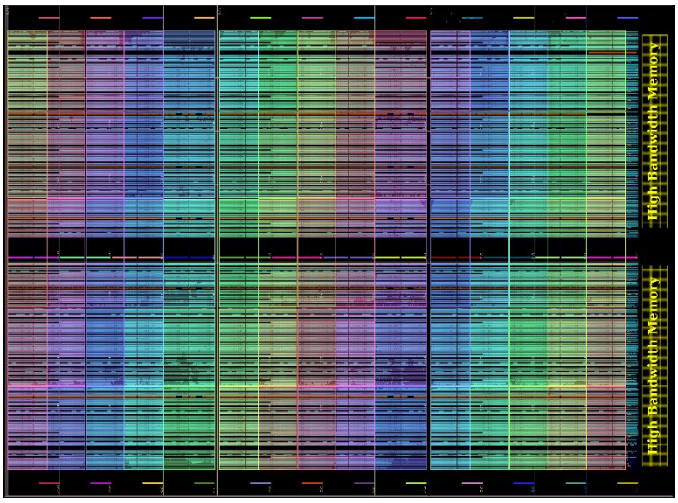

Figure 5: 1332 PE 64-bit 2GRVI Phalanx with HBM

Both SoCs currently run at 300 MHz, however the nascent 2GRVI system is designed for 400-450 MHz operation, and its planned timing improvement work is expected by H2RC 2019.

At 300 MHz, a row of 15 clusters with 3 PEs/s, issuing a series of back to back 32B writes and 512B burst reads, to 15 NoC-AXI RDMA bridges, each connected to two hard AXI-HBM bridges, writes at 130 GB/s, while simultaneously burst reading data at *(precise figure TBD but over 100)* GB/s.

At present it is extremely difficult to measure the power consumption of these SoCs in situ on the Alveo U280-ES1 accelerator card. But for the 1776 PE design, at 300 MHz, Vivado *estimates* power at 109 W, including: clocks: 8 W; LUTs: 26 W; interconnect: 39 W; BRAM: 5 W; URAM: 6 W; HBM: 16 W; static power: 9 W. However, comparable GRVI Phalanx designs, measured on a VU9P, at 250 MHz, had a power/PE (including PE's share of interconnect, NoC, RAMs, etc.) of 18-24 mW/PE, so this estimate is probably conservative.

Once the system RTL is integrated with the SDAccel-for-RTL system and its shell (following section) it will be possible to measure system power using the SDAccel software. This too is expected by H2RC 2019.

VII. Towards an OpenCL-Like Model and Tools

To date, GRVI Phalanx and 2GRVI Phalanx have been programmed in explicitly placed, explicitly scheduled, bare metal C++ with a simple message passing library. This is too basic, too low level, and too unfamiliar to most programmers.

Now with availability of U280 and U50 accelerator cards, 2GRVI Phalanx for HBM-FPGAs can build upon the Xilinx SDAccel (OpenCL) for RTL tools. Here the host-side code is standard (Xilinx supported) OpenCL, copying buffer data to/from the FPGA, and scheduling and invoking OpenCL kernels in the FPGA. (SDAccel-for-RTL is also available on AWS F1, simplifying the back-port of 2GRVI Phalanx to F1.)

Unlike SDAccel's conventional single-work-item OpenCL-to-FPGA-kernel flow, which incurs several hours of place and route time per design iteration to build a new kernel bitstream, here a 2GRVI Phalanx overlay will directly execute OpenCL kernels in software in the many RISC-V PEs across the Phalanx overlay. (It is an open design question how OpenCL kernel developers might access any FPGA custom instructions and standalone accelerator cores.) In this model, an OpenCL *work item* runs on and corresponds to a 2GRVI PE, and an OpenCL *work group*, which provides shared memory across a set of work

items, corresponds to a PE cluster. A parallel kernel invocation will execute across the many clusters of PEs in the Phalanx.

As with most GPU OpenCL NDRange memory-based kernels' execution, the typical workgroup will block copy its input data from DRAM (here an HBM2 channel) to group-shared memory (here cluster RAM); compute on it locally (across work items / RISC-V PEs); then block copy back the result data to DRAM.

Since there does not yet exist an OpenCL kernel compiler for RISC-V, instead an OpenCL-like runtime library and some macros will implement the OpenCL memory hierarchy types and kernel runtime functions including get_work_item, barrier, and (new) burst write/burst read block copy routines. The resulting OpenCL-like integer vector add kernel might read:

```
kernel void vector_add(
  global int* g_a,
  global int* g_b,
  global int* g_sum,
  const unsigned n)
{
  local align int a[N], b[N], sum[N];
  int iloc = get_local_id(0) * n;
  int iglb = (get_group_id(0) * get_local_size(0)
              + get_local_id(0)) * n;
  int size = n * 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; ++i)
    sum[i] += a[i] + b[i];
  barrier(CLK_LOCAL_MEM_FENCE);

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

Of course, even with an HBM2 memory system, external DRAM is slow and energy-inefficient. When aiming data at memory it is better to miss it entirely. Besides support for the classic memory copying kernel pattern (above), we aspire to also provide OpenCL pipes support, so that many concurrent OpenCL kernels may be composed together in a dataflow pipeline. Here again the kernel workgroup invocations will map to clusters, and with clusters composed by (on-die) message passing over the NoC, rather than by exchanging buffers in in HBM DRAM.

VIII. Xilinx HBM FPGAs: Research Directions

To obtain the full bandwidth of the Xilinx VU3xP/VU4xP HBM2 memory system requires the SoC master all 32 AXI-HBM bridges, at full frequency. In addition, for best channel efficiency, memory channels need long burst transactions – at least two beats (64 B), and ideally longer.

Also, as noted, the array of hard AXI-HBM bridges incorporates a switch so that any AXI master may access any memory bank. However, the switch is not a full-bandwidth crossbar, so there may be at most two eastbound transactions and two westbound transactions through any vertical section of the switch at any instant. That's only about 50 GB/s of bisection bandwidth across the switch. Perhaps it is better to use the 2D torus Hoplite NoC itself as the horizontal interconnect to ensure accesses to a given bank originate via an AXI bridge adjacent to that bank's memory controller.

It is also unclear how to best expose the HBM memory system as an abstraction to software. For example, if you provide byte granularity loads and stores to HBM (which hardware supports), when software developers use that facility, you have lured them into a low performance design pattern.

To date Phalanx has eschewed caches, but the 2GRVI Phalanx HBM design might adopt small distributed last level caches at the AXI-HBM bridges to reduce access latency of hot blocks of data. However the utility of such caches as bandwidth filters is reduced in a device with such abundant bandwidth!

Another promising area of research will explore enhancing the NoC-AXI RDMA bridges to perform a variety of "remote computing at the memory interface" functionality such as: scatter/gather accesses, block zero, block copy, add to memory, checksum, select, reduce, regexp, sort, and decompress. Such "compute at the memory" controllers should reduce the need for ultimate FPGA-spanning full HBM bandwidth R/W.

IX. Xilinx HBM FPGAs: Impressions, and the Democratization of HBM Memory Systems

We have had access to an Alveo U280-ES1 with a VU37P-ES1 for approximately ten weeks. We have found that the hard AXI-HBM bridges are easy to design to. The bridges' switch simplifies SoC interconnect implementation, and compared with other UltraScale+ FPGAs such as VU9P, here the provision of 32 hard AXI-HBM bridges saves 100,000s of LUTs previously spent on soft DDR4 DRAM controllers (15-20 KLUTs each) and on many 10K LUT soft AXI interconnects. These hard HBM interfaces also make easy work of DRAM timing closure.

Now with the availability of HBM FPGAs, scientists and engineers have access to high bandwidth memory systems in excess of 400 GB/s, leapfrogging x86 PC bandwidth, and approximately matching GPU bandwidth. This heralds a golden age for custom HPC and datacenter accelerators.

X. Acknowledgements

We thank Paul Hartke of Xilinx Research Labs for access to an Alveo U280-ES1 card with VU37P-ES1 used in the development and testing of the two kilocore HBM Phalanxes.

References

[1] A. Putnam, et al, A reconfigurable fabric for accelerating large-scale datacenter services, in 41st ISCA, June 2014.

[2] J. Gray. GRVI-Phalanx: A Massively Parallel RISC-V FPGA Accelerator Accelerator. In Proc. 24th IEEE FCCM, May 2016.

[3] J. Gray, GRVI Phalanx: …: A 1680-core, 26 MB Parallel Processor Overlay for Xilinx UltraScale+ VU9P, poster, Hot Chips 2017 http://fpga.org/wp-content/uploads/2017/08/HotChips29-GRVI-Phalanx-extd-abs.pdf

[4] J. Gray, FPGA CPU News, Multiprocessors, resource sharing, and all that, Blog post, March 5, 2002 http://www.fpgacpu.org/log/mar02.html#020305

[5] N. Kapre and J. Gray. 2017. Hoplite: A Deflection-Routed Directional Torus NoC for FPGAs. ACM Trans. Reconfigurable Technol. Syst. 10, 2, Article 14 (March 2017), 24 pages. DOI: https://doi.org/10.1145/3027486

[6] Xilinx, Virtex UltraScale+ HBM FPGA Product Brief https://www.xilinx.com/publications/product-briefs/virtex-ultrascale-plus-hbm-product-brief.pdf

[7] Xilinx, AXI High Bandwidth Memory Controller v1.0 https://www.xilinx.com/support/documentation/ip_documentation/hbm/v1_0/pg276-axi-hbm.pdf