2GRVI Phalanx: A Kilocore RISC-V RV64I Processor Cluster Array with HBM2 In a Xilinx VU37P FPGA

Work in Progress Report

Jan Gray | Gray Research LLC | Bellevue, WA | http://fpga.org
Software-First FPGA Accelerator Design

• *Make it easier* for programmers to exploit spatial fabrics
• Manycore accelerator overlays
  • Run C++ or OpenCL kernels on 100s of soft processors
  • Add custom functions/accelerators/memories to suit
  • More 5 second recompiles, fewer 5 hour place and routes
• Software + overlays = familiar programming experiences, easier ports, rapid iteration, design agility
GRVI Phalanx Accelerator Framework

- A processor cluster array overlay
  - GRVI/2GRVI: RISC-V processing elements
  - Phalanx: fabric of clusters of PEs, memories, accelerators, bridges, IOs
  - Hoplite: 2D torus network on chip
GRVI Processing Element

• Simpler PEs $\rightarrow$ more PEs $\rightarrow$ greater memory parallelism
• GRVI: austere RISC-V RV32I + mul*/lr/sc

~320 LUTs @ 400 MHz
GRVI Cluster: PEs, Shared Memory, Accelerators

- IMEM 4-8 KB
- IMEM 4-8 KB
- IMEM 4-8 KB
- IMEM 4-8 KB
- IMEM 4-8 KB
- IMEM 4-8 KB
- IMEM 4-8 KB

- PE
- PE
- PE
- PE
- PE
- PE
- PE

- CMEM = 128 KB CLUSTER DATA

- XBAR

- 2:1
- 2:1
- 2:1

- 32

- 64

- ACCELERATOR(S)

- ~3500 LUTs
Cluster Composition: Message Passing On a NoC

- **Hoplite**: FPGA-optimal 2D torus NoC router
  - Single flits, unidirectional rings, deflection routing, multicast; configurable
  - 300b-wide router uses only ~330 LUTs

![Diagram of Hoplite router](image)

256b @ 400 MHz = 100 Gb/s links
GRVI Cluster: PEs, Memory, Router, Message Passing

PGAS: { mx:1; my:1; x:4; y:6; addr:20 } or { dram_addr:40 }
$10 \times 5 \text{ Clusters} \times 8 \text{ PEs} = 400 \text{ PEs}$

(KU040, 12/2015)
30×7 Clusters x 8 PE = 1680 PEs, 26 MB SRAM
(VU9P, 12/2016)

- 400,000 MIPS @ 250 MHz @ 40 W
GRVI Phalanx V1 Shortcomings

- 32b pointers: awkward for big data on AWS F1, OpenCL
- 32b accesses: wastes half of 64b UltraRAMs bandwidth?
- In-order μarch: stall on loads = ~5 cycles
- DDR4 bandwidth << GPU GDDRx/HBM2 bandwidth
UltraScale+ HBM2 FPGAs!

- VU37P w/ two 4 GB HBM2 stacks
- 32 AXI-HBM bridge/controllers
- 32 x 256b x 450 MHz = 460 GB/s
V2 Redesign for HBM FPGAs

- Latency tolerant 2GRVI RV64I PEs
- 64b cluster datapaths
- 32B/cycle deep pipeline NoC-AXI RDMA bridges
- Double NoC column rings
2GRVI – A Simple, Latency Tolerant RV64I PE

- Register file scoreboard: only stall issue on use of a busy register
- Concurrent execution and out of order retirement
- Example: unrolled block copy – no issue stalls even with 7 cycle memory

```c
for (int i = 0; i < N; i += 8) {
    to[i] = from[i];
    to[i+1] = from[i+1];
    to[i+2] = from[i+2];
    to[i+3] = from[i+3];
    to[i+4] = from[i+4];
    to[i+5] = from[i+5];
    to[i+6] = from[i+6];
    to[i+7] = from[i+7];
}
```

<table>
<thead>
<tr>
<th>time</th>
<th># ex:pc</th>
<th>insn</th>
<th>rd</th>
<th>res</th>
</tr>
</thead>
<tbody>
<tr>
<td>3818</td>
<td>0010c</td>
<td>bne a4,t4,00c4</td>
<td></td>
<td>0[720]=000000000000000e4</td>
</tr>
<tr>
<td>3820</td>
<td>0~0110</td>
<td>(annul)</td>
<td></td>
<td>0[728]=000000000000000e5</td>
</tr>
<tr>
<td>3822</td>
<td>00c4</td>
<td>1d t3,0(a4)</td>
<td></td>
<td>0[730]=000000000000000e6</td>
</tr>
<tr>
<td>3824</td>
<td>00c8</td>
<td>1d t1,8(a4)</td>
<td></td>
<td>0[738]=000000000000000e7 t3?</td>
</tr>
<tr>
<td>3826</td>
<td>00cc</td>
<td>1d a7,16(a4)</td>
<td></td>
<td>t1? t3?</td>
</tr>
<tr>
<td>3828</td>
<td>00d0</td>
<td>1d a6,24(a4)</td>
<td></td>
<td>t1? a7? t3?</td>
</tr>
<tr>
<td>3830</td>
<td>00d4</td>
<td>1d a8,32(a4)</td>
<td></td>
<td>t1? a6? a7? t3?</td>
</tr>
<tr>
<td>3832</td>
<td>00d8</td>
<td>1d a1,40(a4)</td>
<td></td>
<td>t1? a0? a6? a7? t3?</td>
</tr>
<tr>
<td>3834</td>
<td>00dc</td>
<td>1d a2,48(a4)</td>
<td></td>
<td>0[f40] t1? a0? a1? a6? a7? t3?</td>
</tr>
<tr>
<td>3836</td>
<td>00e0</td>
<td>1d a3,56(a4)</td>
<td></td>
<td>0:t3=00000000000000008 0[f48] t1? a0? a1? a2? a6? a7? t3? t3!</td>
</tr>
<tr>
<td>3838</td>
<td>00e4</td>
<td>sd t3,0(a5)</td>
<td>0:t1=000000000000000e9 0[f50] t1? a0? a1? a2? a3? a6? a7? t1!</td>
<td></td>
</tr>
</tbody>
</table>

- 400 6-LUTs (sans <<)!
### GRVI vs. 2GRVI

<table>
<thead>
<tr>
<th></th>
<th>32b GRVI PE</th>
<th>64b 2GRVI PE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Year</strong></td>
<td>2015 Q4</td>
<td>2019 Q2</td>
</tr>
<tr>
<td><strong>ISA</strong></td>
<td>RV32I + mul/lr/sc</td>
<td>RV64I + lr/sc</td>
</tr>
<tr>
<td><strong>Area</strong></td>
<td>320 6-LUTs</td>
<td>400 6-LUTs (sans shared &lt;&lt;)</td>
</tr>
<tr>
<td><strong>Fmax / congested</strong></td>
<td>400 / 300 MHz</td>
<td>500+ / TBD MHz</td>
</tr>
<tr>
<td><strong>Pipeline stages</strong></td>
<td>2 / 3</td>
<td>2 / 3 / 4 (superpipelined)</td>
</tr>
<tr>
<td><strong>Out-of-order retire</strong></td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td><strong>Cluster, load interval</strong></td>
<td>5 cycles</td>
<td>1 / cycle</td>
</tr>
<tr>
<td><strong>Cluster, load-to-use</strong></td>
<td>5 cycles</td>
<td>3-6 cycles</td>
</tr>
<tr>
<td><strong>Cluster, Σ RAM BW</strong></td>
<td>4.8 GB/s (300 MHz)</td>
<td>12.8 GB/s (400 MHz)</td>
</tr>
</tbody>
</table>
Phalanx SoC: 15x15-3 Array of Clusters + HBM + PCIe

PE ↔ Cluster RAM ↔ NoC ↔ AXI ↔ HBM

- 32 B write request message; 32×n B burst-read request → n×32 B read responses
- PE sends R/W request message to its NoC-AXI bridge; bridge issues request to its AXI-HBM channel(s); bridge sends read response messages to dest. address
- 32 B write + 32 B read response per cycle per bridge
- Measured ~130 GB/s write + ~130 GB/s read at 300 MHz

Cluster { 8 GRVI / 6 2GRVI, 4-8 KB IRAM, 128 KB CRAM, Hoplite router }
NoC-AXI RDMA bridge { 2 256b AXI R/W req queues, 2 resp queues }
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

2019/11/17
2GRVI Phalanx: Kilocore RV64I + HBM Accelerator Framework
NoC-AXI-HBM Transactions in Flight

15 32B Writes

8 512B Burst Reads
222 x 8 GRVI PEs = 1776 RV32I PEs
222 x 6 2GRVI PEs = 1332 RV64I PEs
Phalanx-HBM2 Next Steps

• Tune up to 400+ MHz = ~200 GB/s writes + ~200 GB/s reads
• Computational HBM2 – compute at the bridges
  • Scatter/gather, add-to-memory, block zero, copy, hash, reduce, select, regexp, sort, ...
Phalanx Parallel Programming Models

• Architecture: array of clusters of PEs, no caches, message passing

• Today: bare metal C/C++ + message passing runtime

• Future
  • Flat data parallel NDRange OpenCL kernels
  • Streaming kernels composed with OpenCL pipes
  • ‘Gatling gun’ parallel packet processing
An OpenCL-like Model and Tools

• Familiar to GPU developers(?)

• Host side: Xilinx SDAccel OpenCL runtime
  • Setup, copy buffers, queue parallel kernel calls, wait, copy results

• FPGA side: GRVI Phalanx ⇔ SDAccel-for-RTL shell
  • Map work groups to PE clusters, work items to PEs
  • Memory: global = HBM; local = cluster RAM (static); private = thread (auto)
  • Scheduler (PEs at cluster 0): distribute kernels, map work groups to idle clusters

• Plan. Not yet implemented
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); // from HBM
  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); // to HBM
}
Take Aways

- (Prior work)
  - Software-first, software-mostly manycore accelerators
  - Die filling, FPGA frugal, clustered, tiled, NoC-interconnected overlays

- Democratizing HBM
  - Xilinx AXI-HBM bridges are easy to use, simplify interconnects, save 100Ks LUTs
  - HBM bandwidth is now accessible to all

- Towards an OpenCL-like SDK, on AWS F1, Azure NP10, Alveo
References

- The Past and Future of FPGA Soft Processors (2014)

- GRVI Phalanx and Hoplite NoC (2016)
  http://fpga.org/grvi-phalanx/
  http://fpga.org/hoplite/

- 2GRVI Phalanx at Hot Chips 31 (2019)
  http://fpga.org/2019/08/19/2grvi-phalanx-at-hot-chips-31-2019/

- Xilinx AXI High Bandwidth Memory Controller v1.0