Combining real time evaluation with simulation or emulation

Maya Gokhale
Scott Lloyd
Chris Macaraeg

Lawrence Livermore National Laboratory

LLNL-PRES-875935

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

We study application performance on future memory architectures

- Obtain memory traces through simulation or emulation
- Derive realistic latencies, bandwidths for emerging and future memories
  - Combination of simulation/emulation and measurement

- Iterative process to tune parameters
  - Set latency, best guess based on measurement of test patterns on experimental memory
  - Run simulation, capture memory traces
  - Measure latency and bandwidth when traces drive the memory
  - Revise latency estimate

Obtaining traces

- Speed of software simulation
  - CPU simulation multiplier is O(10000)
- Effort and time
  - can take years of effort to develop accurate processor models
  - Have to handle interactions of memory system with CPU
  - FPGA-based emulator or hybrid techniques take lots of time and high level of expertise to build

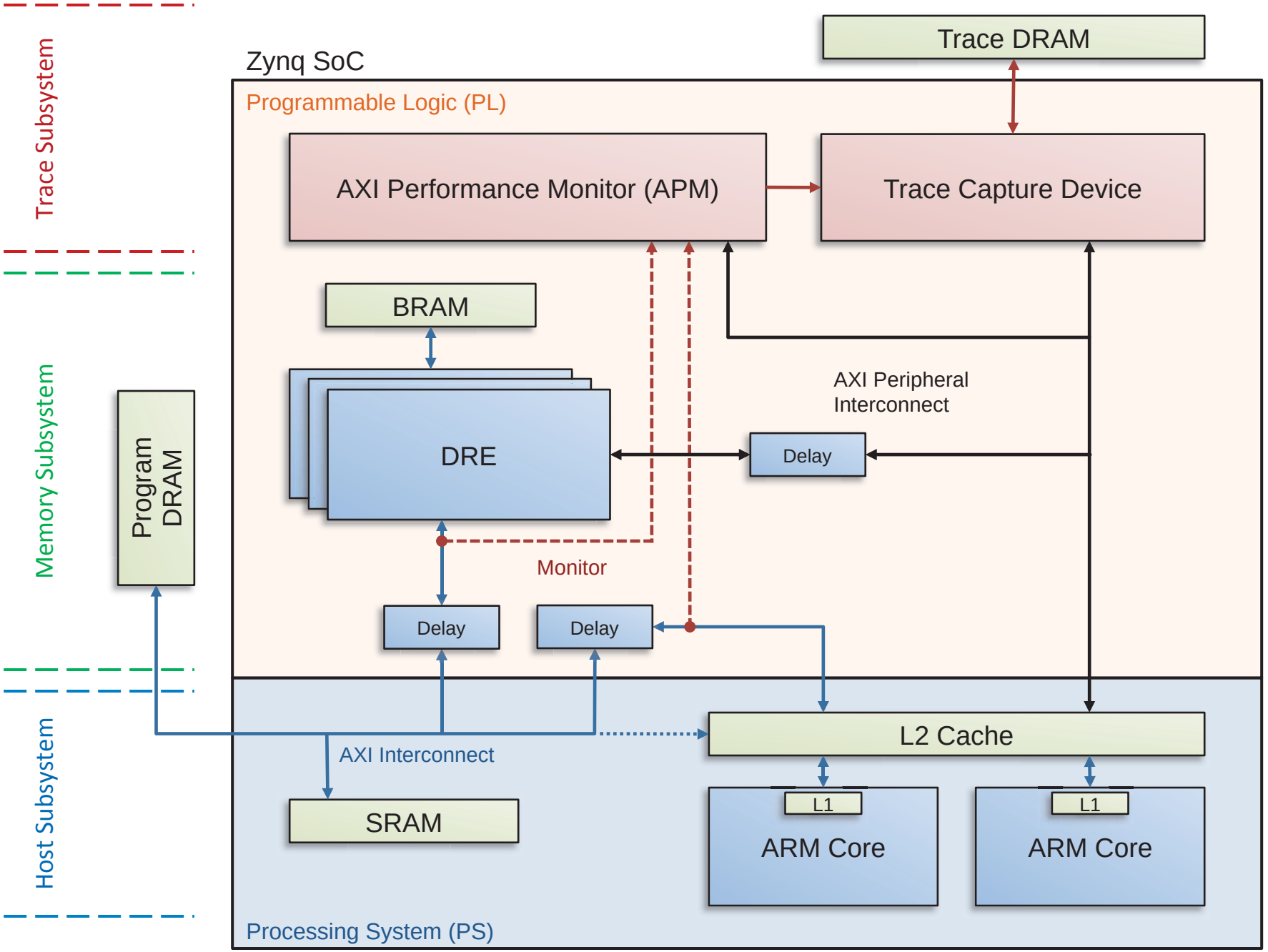
Evaluating accuracy

SpMV (2M x 2M)

Machine	Runtime (sec)
gem5, arm_64 arm_detailed @ 2.00 GHz (prefetch)	2.29
gem5, arm_64 detailed @ 2.00 GHz	1.99
fyi75-pub, AArch64 @ 1.60 GHz	1.71
gem5, x86_64 detailed @ 2.00 GHz	1.96
cab, Xeon E5-2670 0 @ 2.60 GHz	0.508

STREAM (Rate MB/s)

Machine	Copy	Scale	Add	Triad
gem5, arm_64 arm_detailed @ 2.00 GHz (prefetch)	5265.5	4616.9	5545.8	5589.9
gem5, arm_64 detailed @ 2.00 GHz	3001.4	1628.3	2025.5	2008.7
fyi75-pub, AArch64 @ 1.60 GHz	13349.7	11869.3	11692.8	11037.0
gem5, arm_64 detailed @ 2.00 GHz	1546.9	1543.6	2367.7	2245.0
cab, Xeon E5-2670 0 @ 2.60 GHz	13027.8	13102.8	13020.9	13247.5


Trace Subsystem
Memory Subsystem
Host Subsystem
Trace DRAM
Zynq SoC
Programmable Logic (PL)
AXI Performance Monitor (APM)
Trace Capture Device
BRAM
AXI Peripheral
Interconnect
DRE
Delay
Program
DRAM
Monitor
Delay
Delay
L2 Cache
AXI Interconnect
SRAM
L1
ARM Core
L1
ARM Core
Processing System (PS)

Emulator Scaling by 20

Actual

- Memory 1.6 GB/s, 180 ns
- CPU 1-800 MB/s, 1-800 MHz
- DRE 2-1600 MB/s, 1-200 MHz

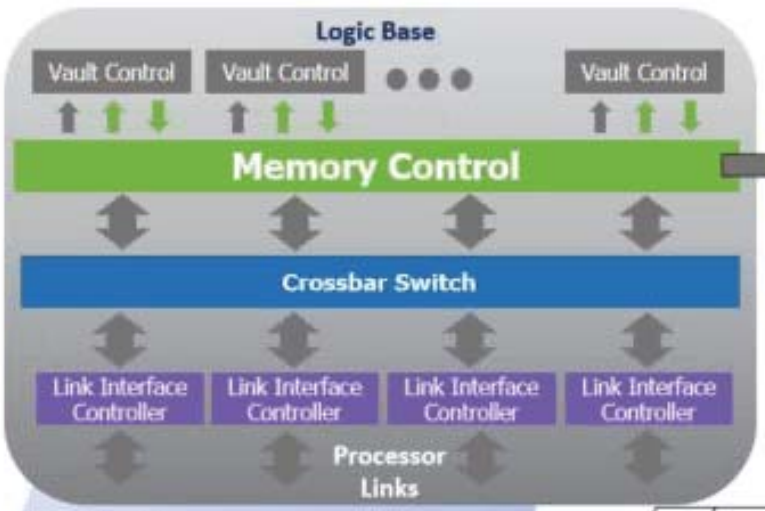
Target

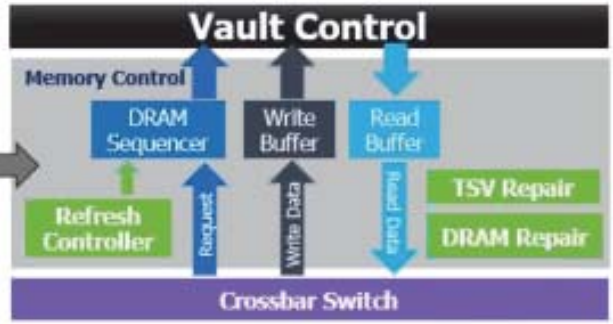
- Memory 40 GB/s, 100ns
- CPU 5 GB/s, 2.5 GHz
- DRE 10 GB/s, 1.25 GHz

Delay is programmable over a wide range: 0 - 262 us in 0.25 ns increments

Component	Actual	Emulated
Memory Bandwidth	1.6 GB/s	32 GB/s (3 Vaults)
Memory Latency	180 ns	9 ns (too low)
Memory Latency w/ delay	180 ns	9+91 = 100 ns
CPU Frequency	128.6 MHz	2.57 GHz
CPU Bandwidth	257 MB/s	5.1 GB/s
DRE Frequency	62.5 MHz	1.25 GHz (base logic)
DRE Bandwidth	500 GB/s	10 GB/s

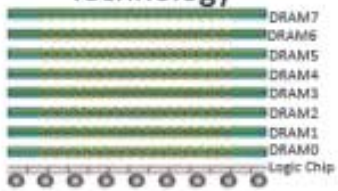
HMC Architecture





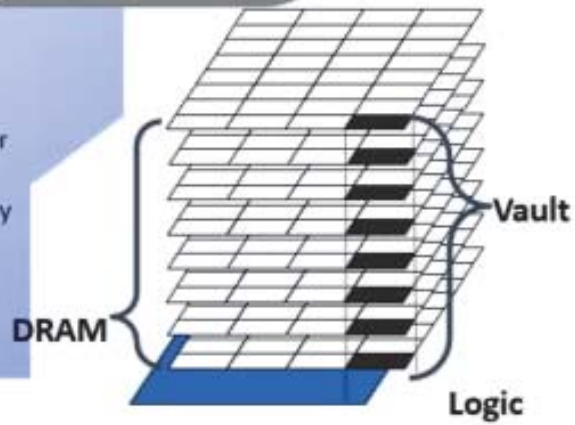
Detail of Memory Interface

3DI & TSV Technology



Logic Base

- Multiple high-speed local buses for data movement
- Advanced memory controller functions
- DRAM control at the memory rather than at distant host controller
- Reduced memory controller complexity and increased efficiency



Vaults are managed to maximize overall device availability -

- Optimized management of energy and refresh
- Self test, error detection, correction, and repair in the logic base layer

Using Arira Design FPGA board

4 Altera Stratix V A3 FPGAs. Each FPGA drives on link.
FPGA contains custom VLIW to generate packets. One packet per clock cycle
Board instrumented to measure BW, latency, power.

HMC characterization using test patterns

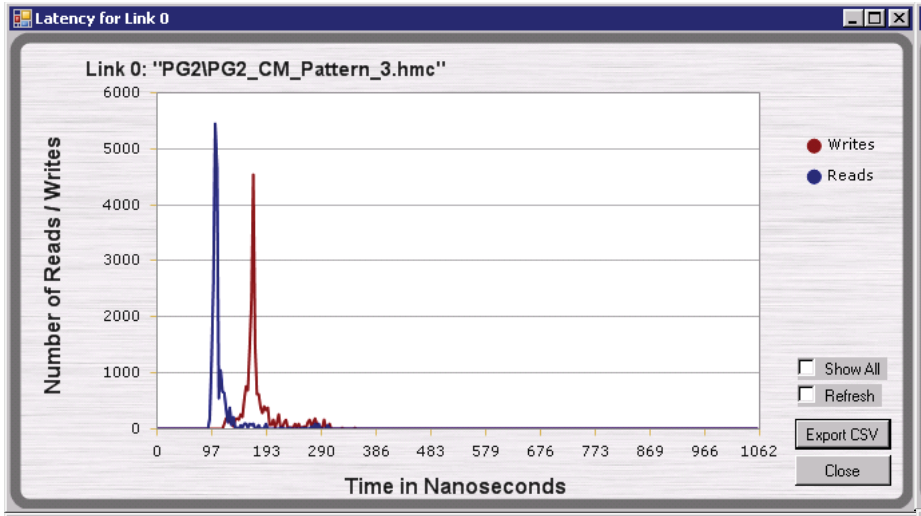
Pattern characteristics: All Writes and Reads are 128 bytes
50% reads, 50% writes

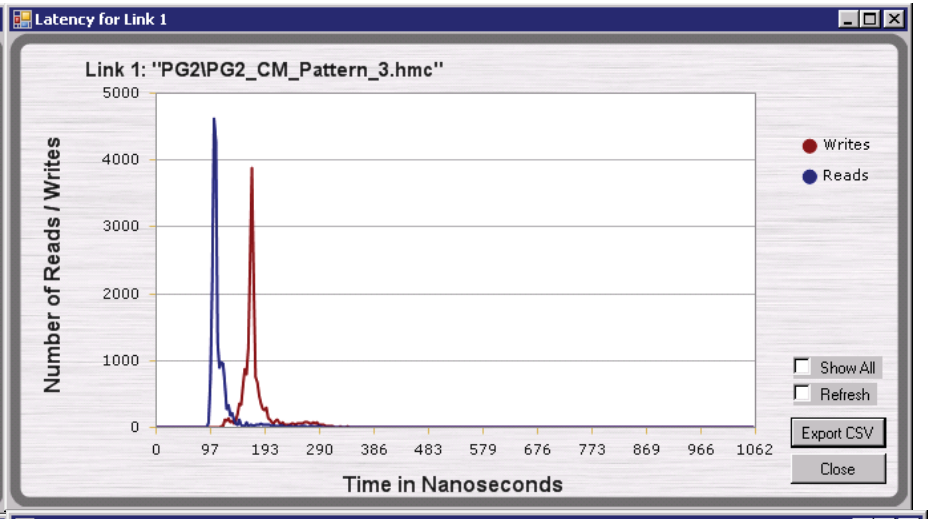
Test1: All links enabled, accessing local vaults per test parameters
Test2: All links enabled, accessing vaults 0, 1, 2, 3

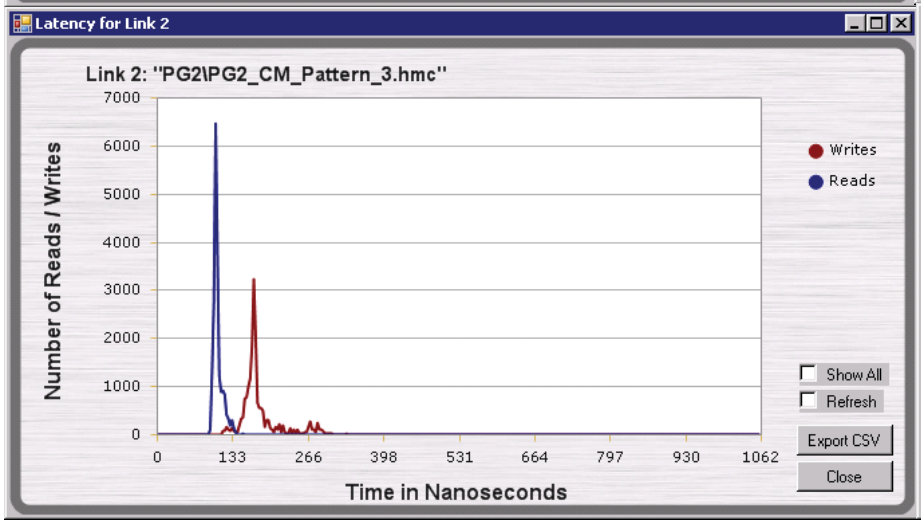
Test Algorithm in pseudo code (test is loaded into each enabled FPGA and drives its associated link):

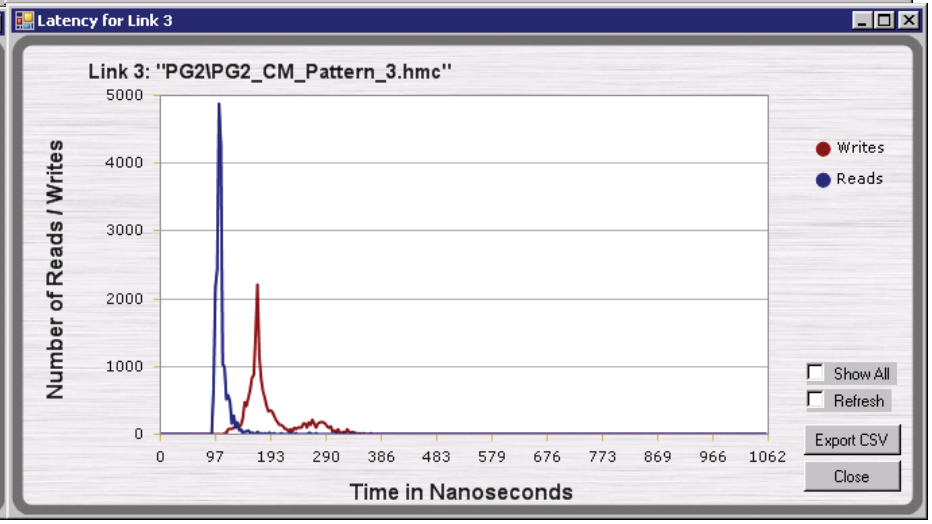
```
Loop_row:
   LoopColumn:
      Write all 4 vaults in each bank of target quadrant
      Read all 4 vaults in each bank of target quadrant
      Increment column by 8 (Writes and Reads are 128 bytes)
   Loop_end
   Increment row
Loop_end
```

Test 1 Latency

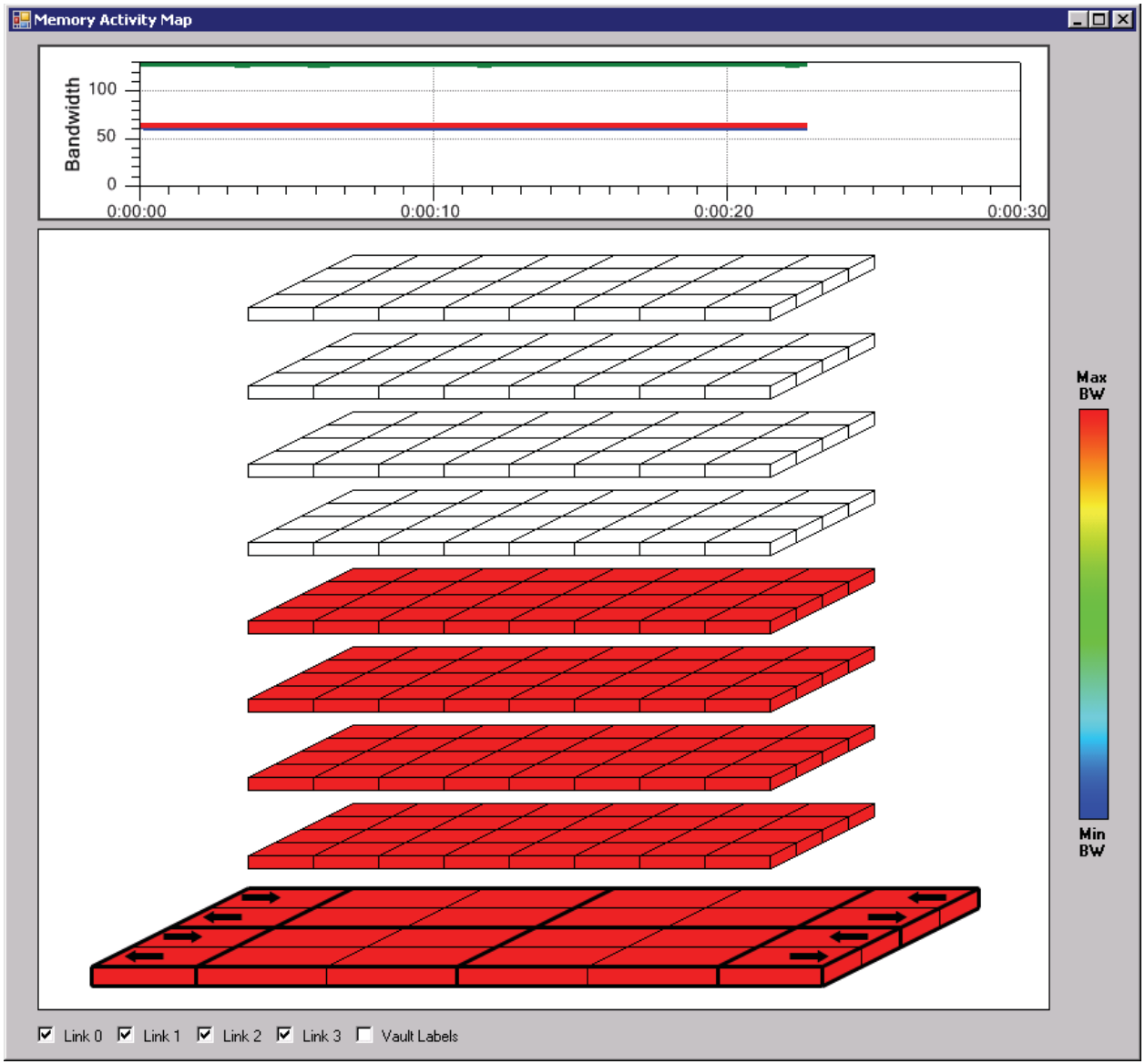




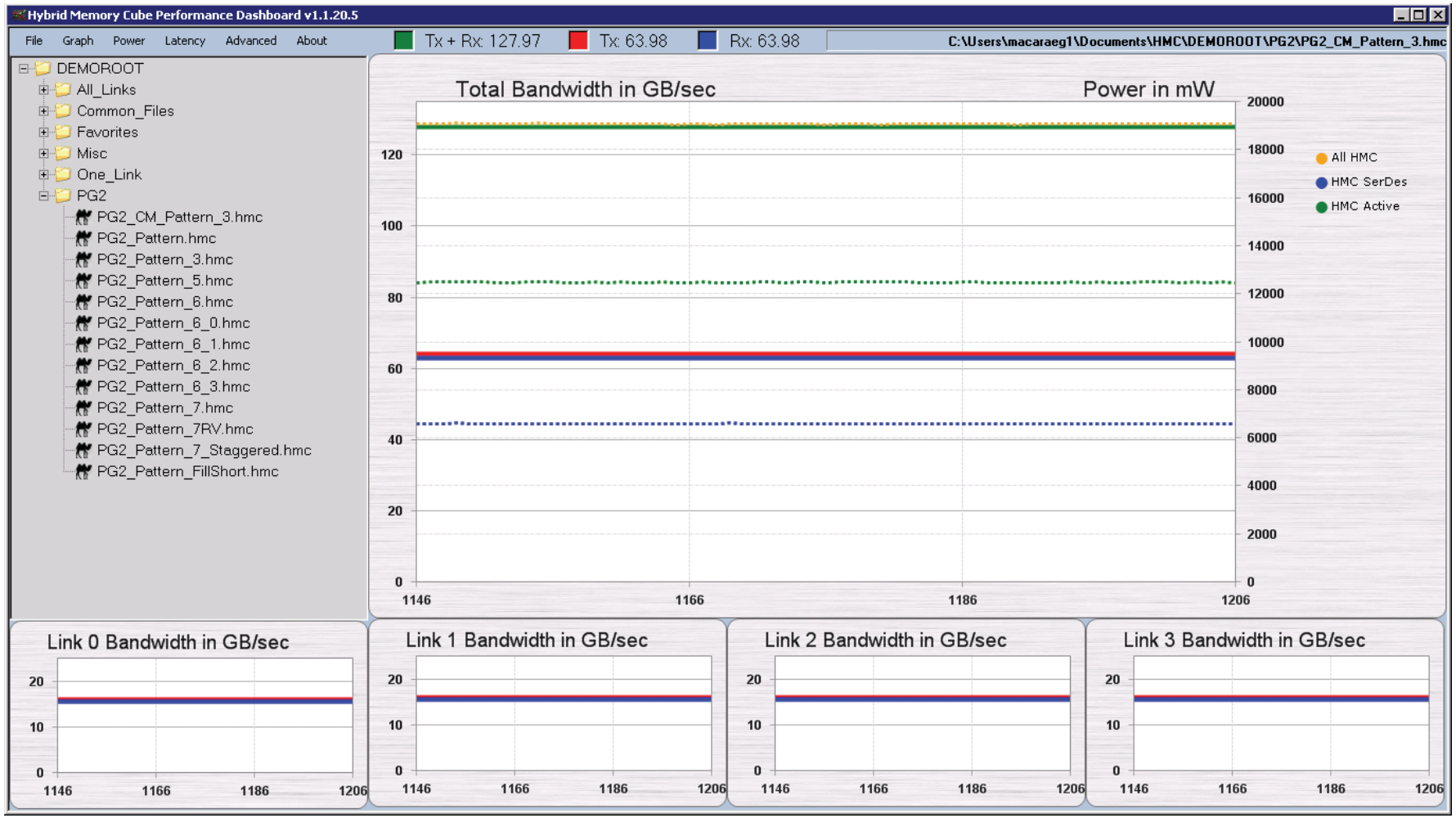




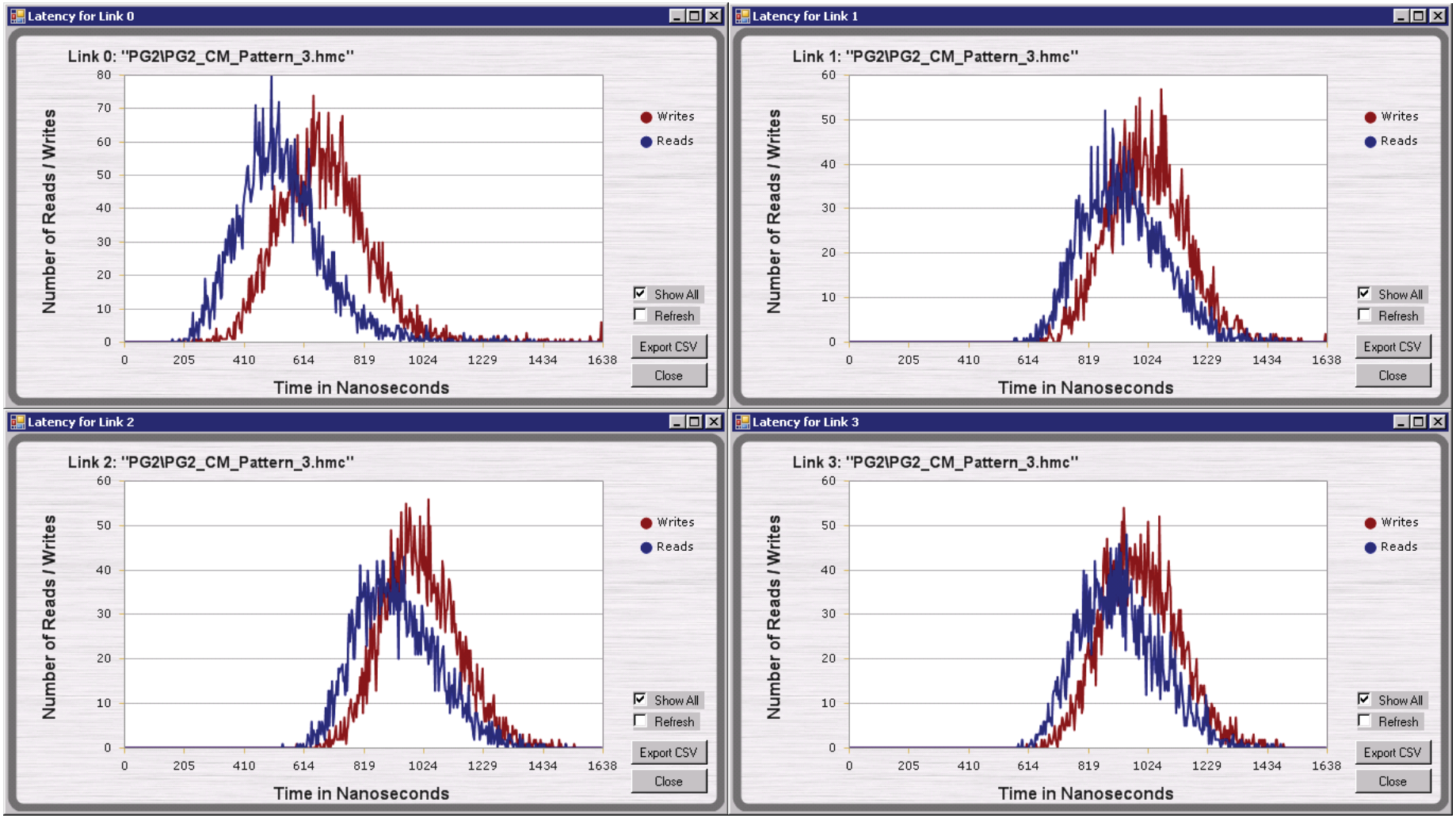
Test 1 Memory Activity Map



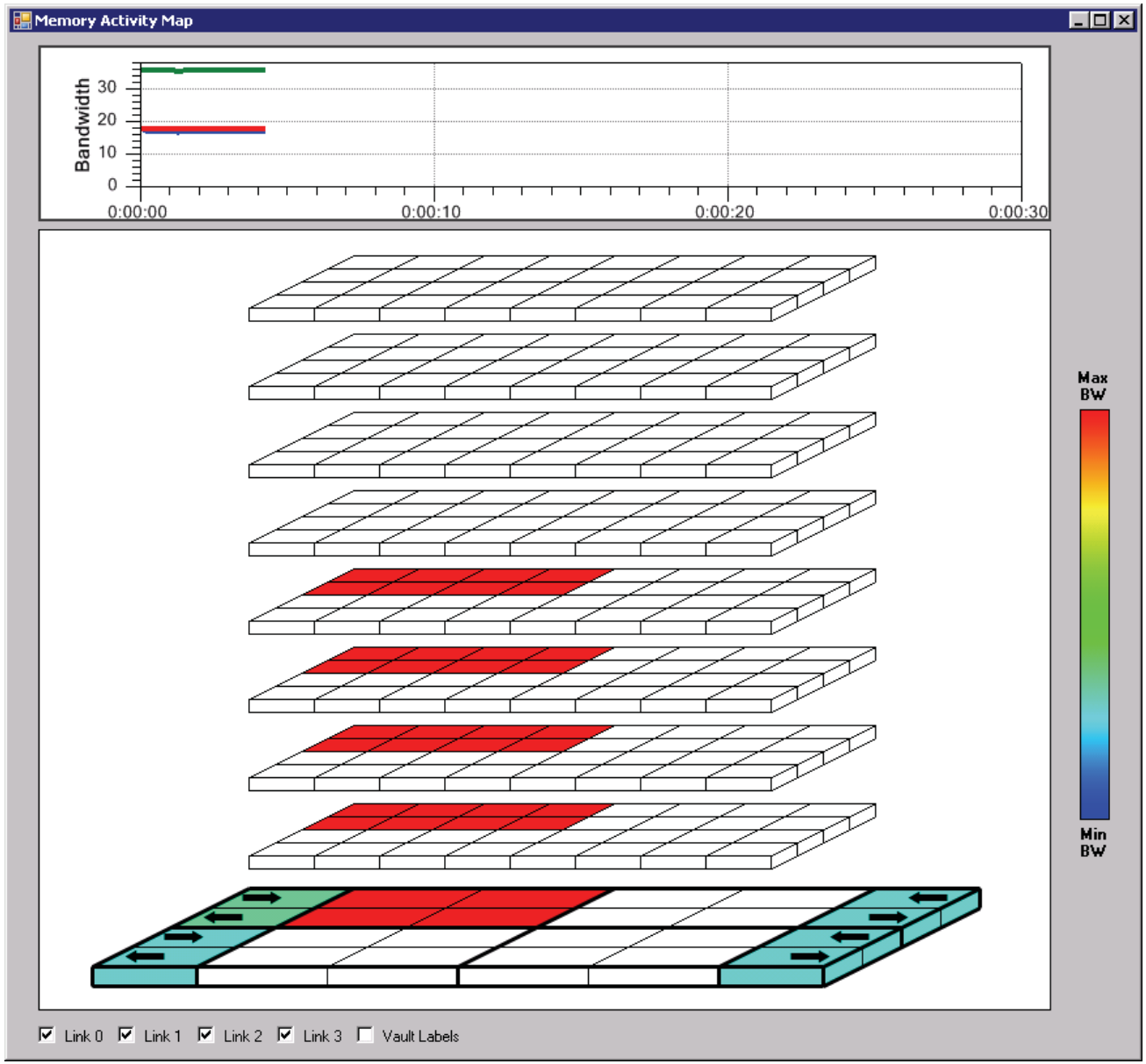
Test 1 Bandwidth



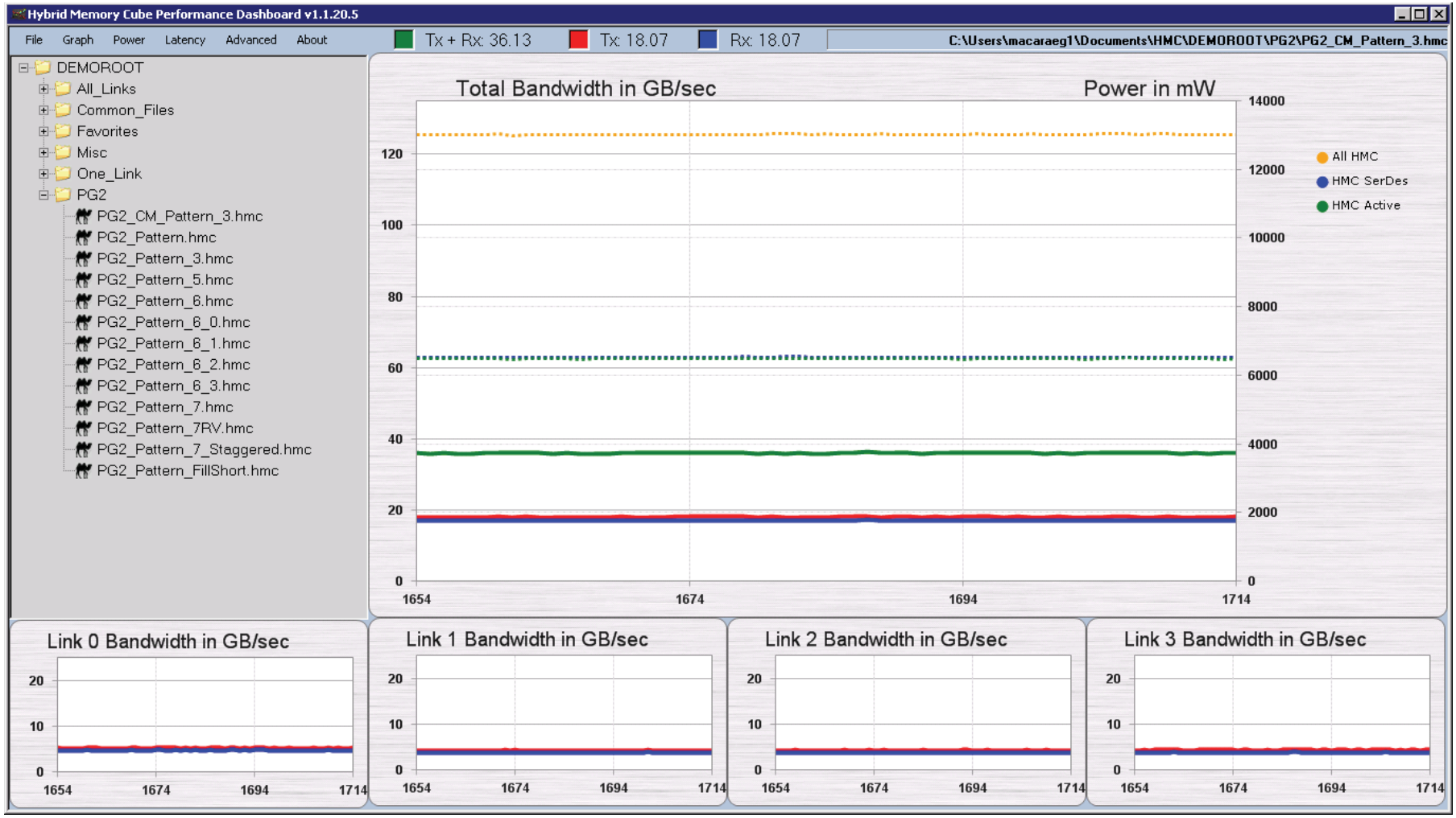
Test 2 – All links active, All links accessing vaults 0, 1, 2, 3 (i.e. Link 0 local vaults)
Latency



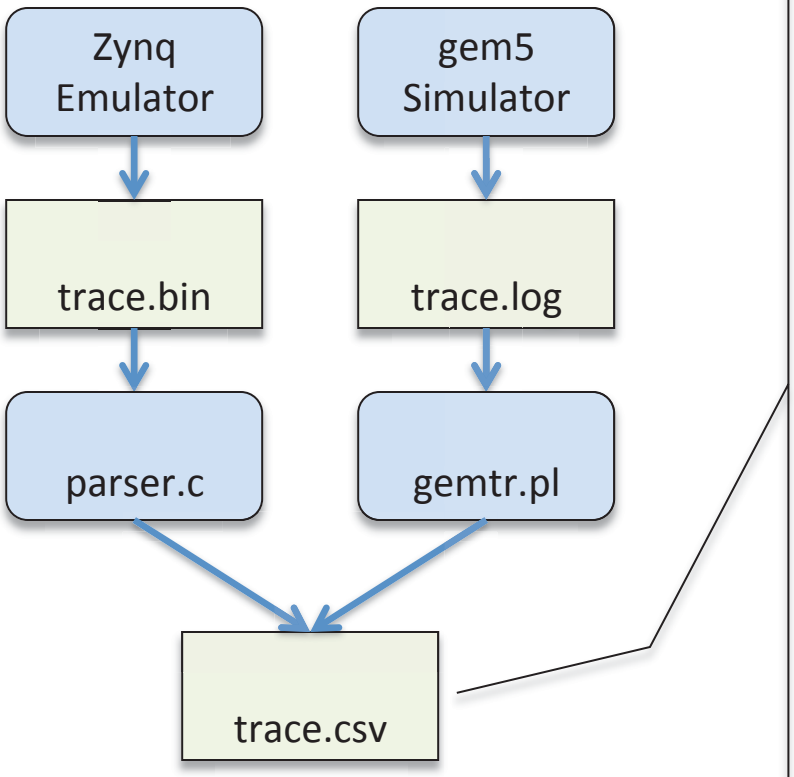
Test 2 – All links active, All links accessing Link 0 local vaults
Memory Activity Map



Test 2 – All links active, All links accessing Link 0 local vaults Bandwidth



Trace Capture

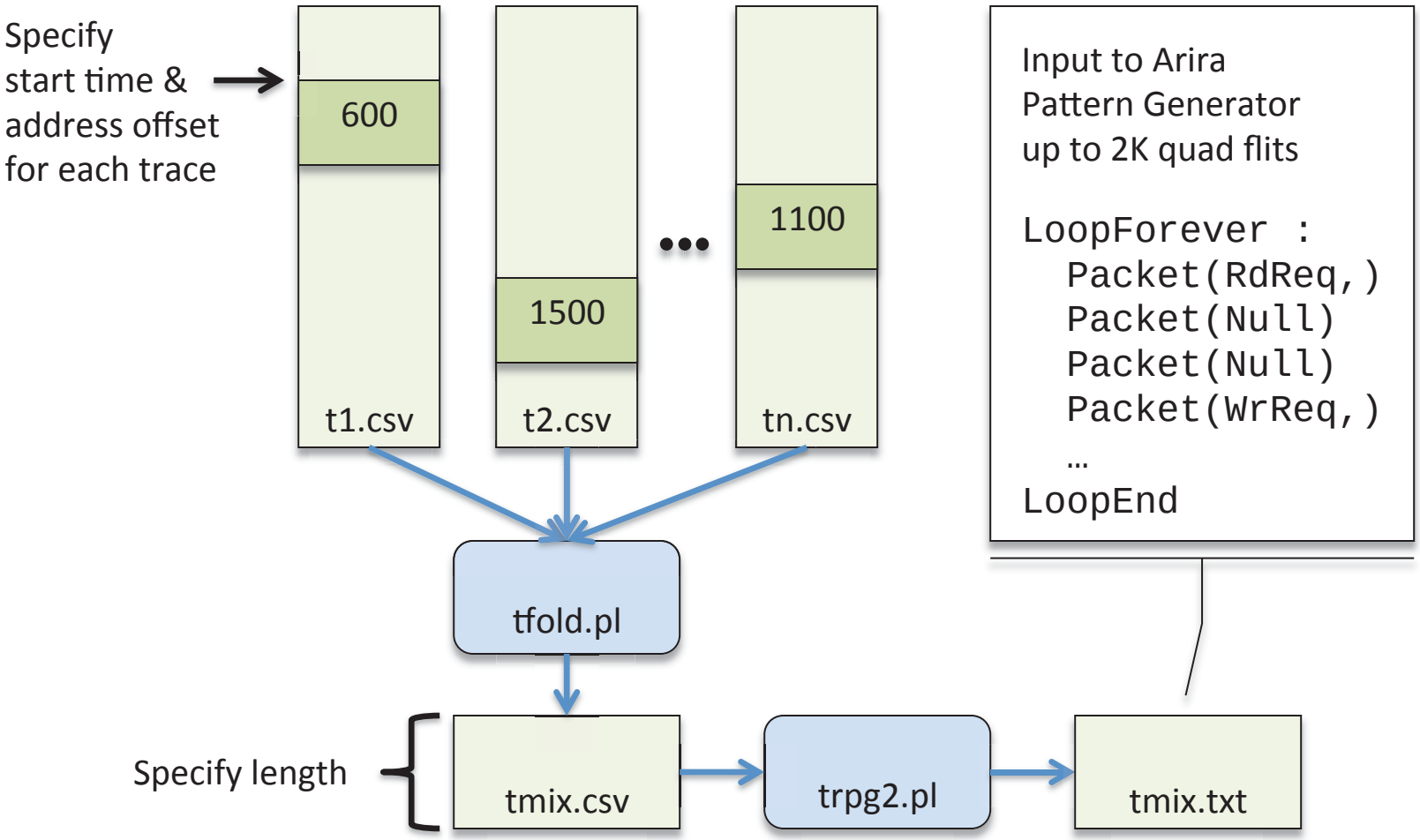



```
0,R,0x40101520,32,29,2131065
0,R,0x7f186b20,32,30,2131091
1,R,0x40185060,8,0,2132263
1,R,0x40185080,8,0,2132270
…
1,W,0x40000030,8,2,2132548
1,W,0x40000034,8,2,2132554
1,R,0x40185260,8,0,2132571
1,R,0x40185280,8,0,2132575
1,W,0x40000038,8,2,2132577
1,W,0x4000003c,8,2,2132581
```

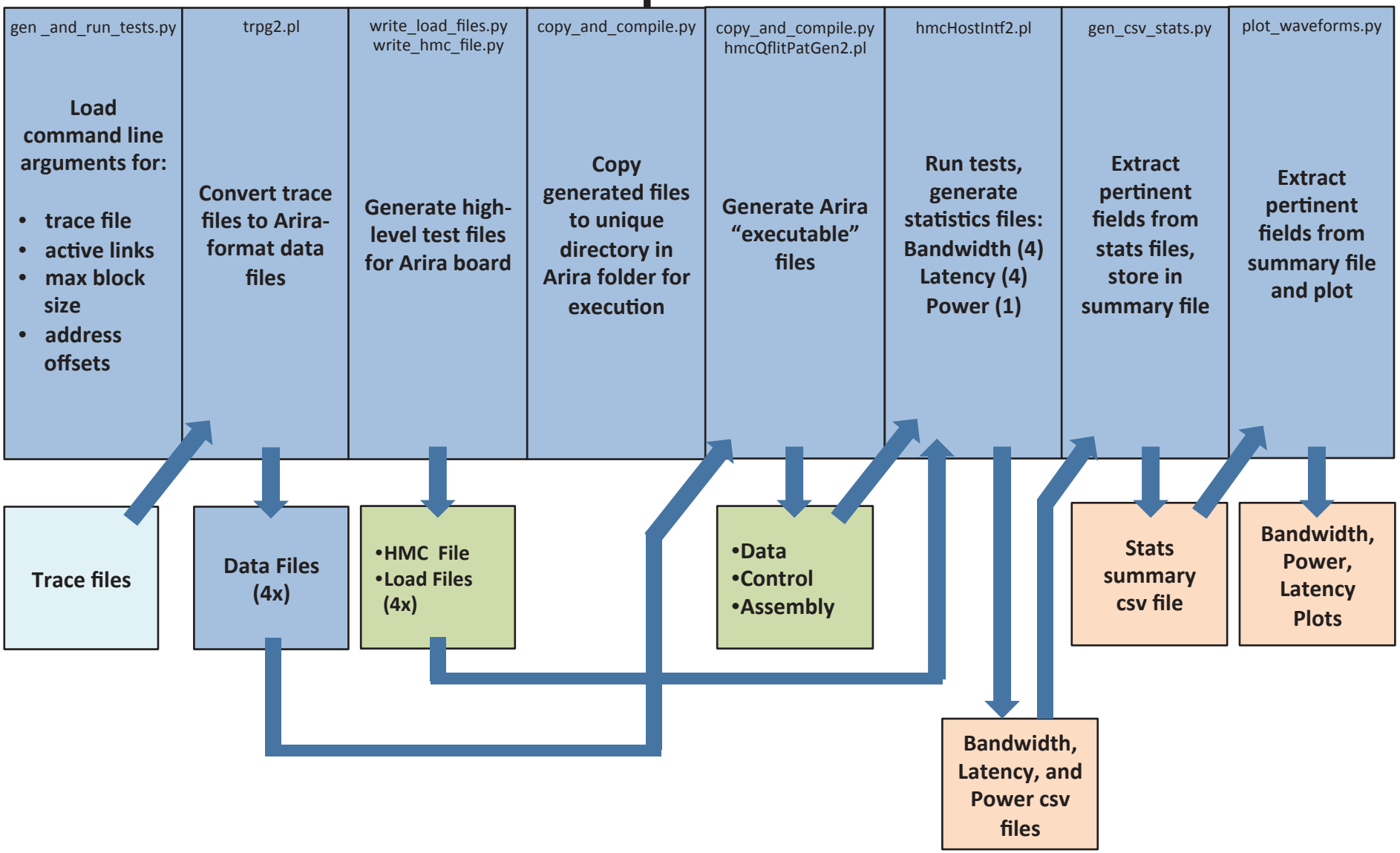
Trace format by column

1) CPU=0 / Accelerator=1
2) Read/Write
3) Address
4) Transaction size in bytes
5) AXI ID
6) Time stamp

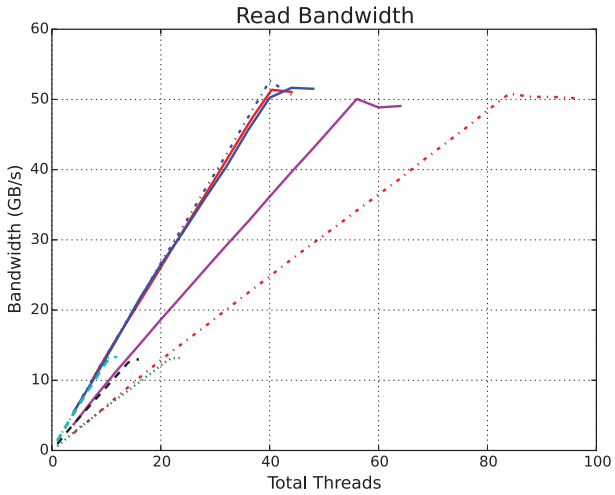
Trace Folding

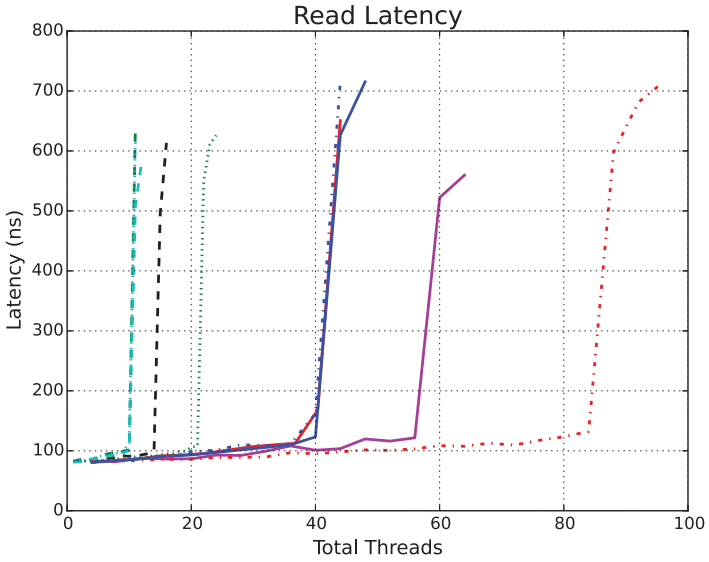


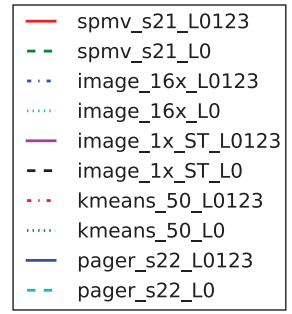
Script Flow

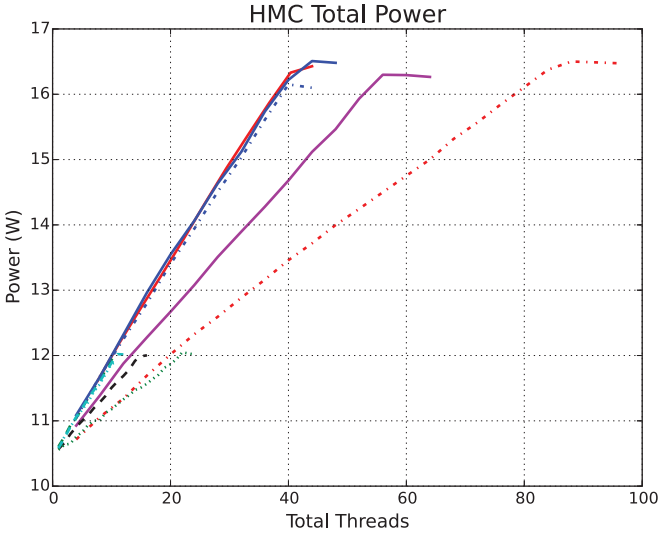

gen _and_run_tests.py
Load command line arguments for:
• trace file
• active links
• max block size
• address offsets
trpg2.pl
Convert trace files to Arira-format data files
write_load_files.py
write_hmc_file.py
Generate high-level test files for Arira board
copy_and_compile.py
Copy generated files to unique directory in Arira folder for execution
copy_and_compile.py
hmcQflitPatGen2.pl
Generate Arira "executable" files
hmcHostIntf2.pl
Run tests, generate statistics files:
Bandwidth (4)
Latency (4)
Power (1)
gen_csv_stats.py
Extract pertinent fields from stats files, store in summary file
plot_waveforms.py
Extract pertinent fields from summary file and plot
Trace files
Data Files (4x)
•HMC File
•Load Files (4x)
•Data
•Control
•Assembly
Bandwidth, Latency, and Power csv files
Stats summary csv file
Bandwidth, Power, Latency Plots

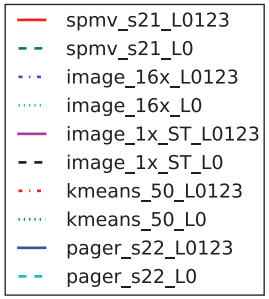
Measured HMC Performance for Various Workloads











- Bandwidth peaks at about 50 GB/s for 32 byte packets
- Latency typically ranges between 93 ns and 133 ns
  - Spikes exponentially at saturation
- Power ranges from 10 W (idle) up to about 16.5 W

Methodology can be used on most application (segments)

- Run application in simulator of your choice and capture memory traces
  - Latency initialized to first guess from pattern generator tests
- Select portions of trace to run individually
- Run on board to measure latency, bandwidth, power
- Re-run application through simulator to find revised run time, latency, bw, etc.