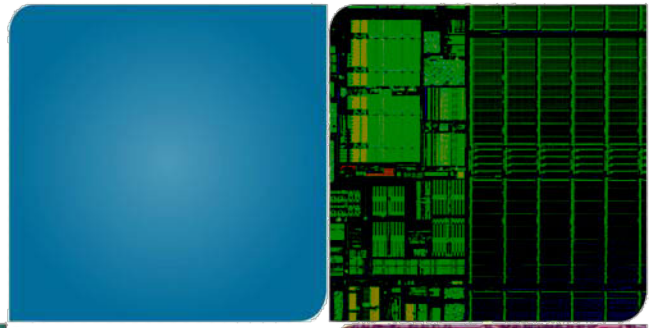
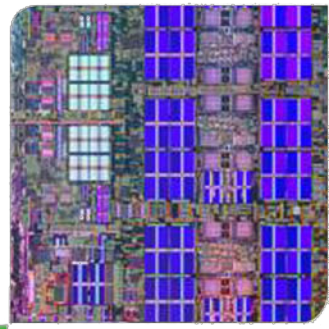
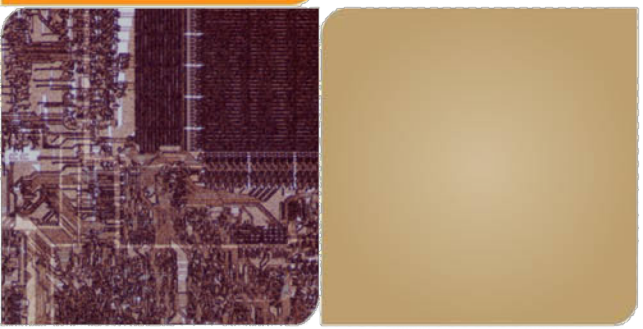




Performance Characterisation and Simulation of Intel's Integrated GPU Architecture

Prasun Gera, Hyojong Kim, Hyesoon Kim (Georgia Tech)

Sunpyo Hong , Vinod George, Chi-Keung Luk (Intel)



Georgia Tech

comparch

intel

Background

- GPUs are widely used in HPC and machine learning
- Studied extensively in the context of discrete GPUs

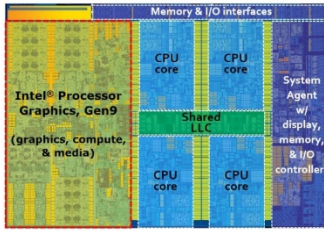
NVIDIA GPUs

- Key characteristics:
  - Connected to the CPU externally (for eg., over PCI-E)
  - Separate host and device memory
  - SIMT execution model

- New low power solutions for client systems: Integrated GPUs (iGPU)

Intel HD 530

- Key characteristics:
  - On the same chip as CPU
  - Cache coherent memory hierarchy. Shared DRAM
  - SIMD + SMT
  - Predicated execution, 2D register addressing



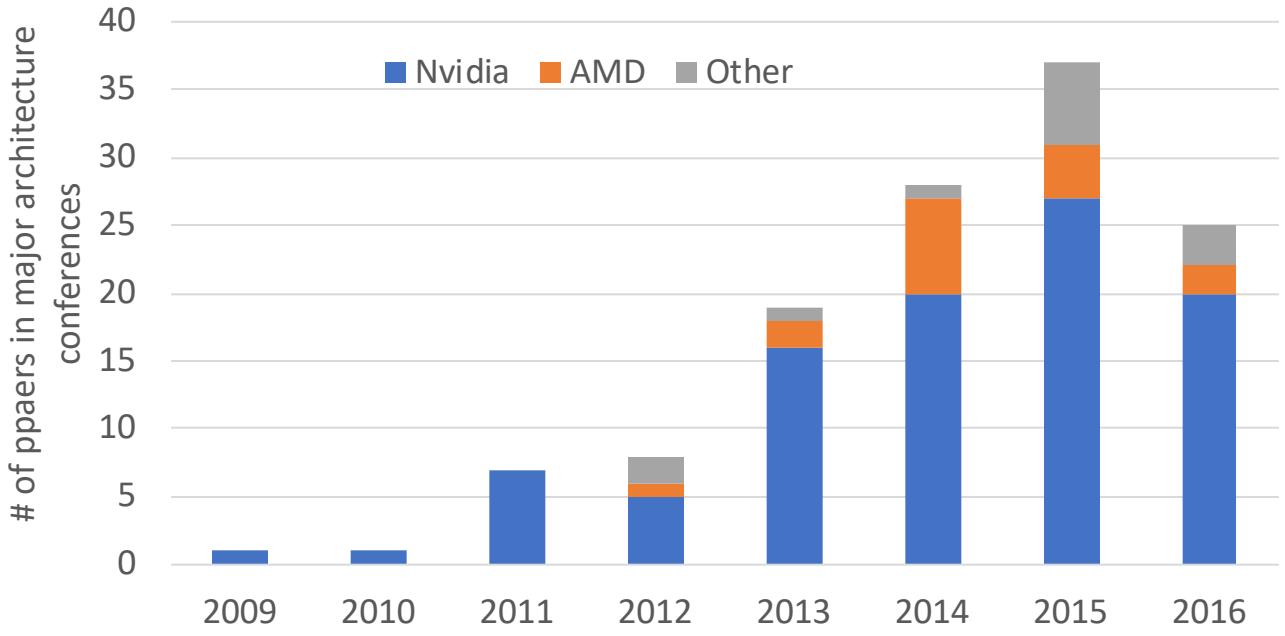
Motivation

iGPU application/research domains:

- Media and gaming
- Mobile and IoT devices
- CPU+GPU heterogeneous workloads.
  - OpenCL unified memory. Zero copy memory objects
- GPU virtualization
- Many more

Motivation

External simulators for NVIDIA and AMD GPUs are available



Need tools to **model** and **simulate** iGPUs



Contributions

- Characterise the performance of Intel's Skylake and Kabylake iGPUs
- Develop an instruction level trace generator for Intel iGPUs built upon GT-Pin, a binary instrumentation framework
- Develop the iGPU module in MacSim, an open source heterogeneous architecture simulator.



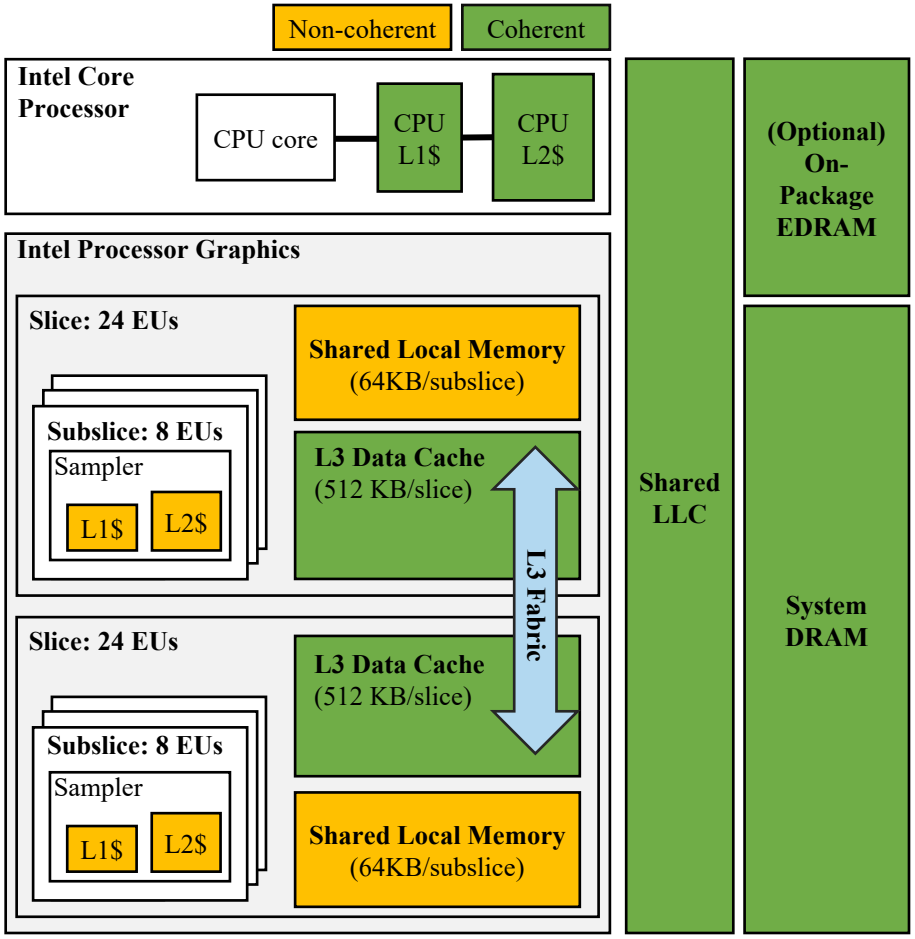
Outline

- Intel iGPU architecture
- Microbenchmark analysis:
  - Thread scheduling and floating point performance
  - Memory hierarchy latency and CPU-GPU cache sharing effects
  - Memory bandwidth and coalescing
  - Memory level parallelism (MLP)

Measure real hardware

- GT-Pin and Trace generation
- Simulation evaluation
- Conclusion

Intel GEN9 IGPU Architecture



- Execution units (EUs) that perform SIMD computation.
- Collection of 8 EUs form a subslice.
- The subslice also contains a common I-cache, L1 and L2 sampler caches, and a memory load/store unit called the data port.
- 3 subslices aggregated into 1 slice. The slice additionally consists of the L3 data cache, and a highly banked shared local memory (SLM).
- L3 cache coherent with CPU.
- LLC , optionally eDRAM cache, and system DRAM shared with CPU

Hardware Specifications

TABLE I: CPU Parameters

Parameter	i7-6700k (SKL)	i7-7567U (KBL)
Cores	4	2
Threads	8	4
Base Freq	4.00 GHz	3.50 GHz
Max Turbo Freq	4.20 GHz	4.00 GHz
L1 I-cache (per core)	32 KB	32 KB
L1 D-cache (per core)	32 KB	32 KB
L2 cache (per core)	256 KB	256 KB

TABLE II: iGPU Parameters

Parameter	HD 530 (SKL)	Iris Plus 650 (KBL)
EUs	24	48
H/W Threads per EU	7	7
SIMD FP Units per EU	2	2
SIMD FP Width per unit	32 bits	32 bits
Base Freq	350 MHz	300 MHz
Max Dynamic Freq	1.15 GHz	1.15 GHz
Number of Slices	1	2
Number of Sub-Slices	3	6
L3 data cache	512 KB	1 MB
SLM (per sub-slice)	64 KB	64 KB

TABLE III: Shared Resources

Parameter	i7-6700k + HD 530 (SKL)	i7-7567U + Iris Plus 650 (KBL)
LLC	8 MB	4 MB
eDRAM	N/A	64 MB
DRAM	DDR4 2133 MHz	DDR4 2133 MHz

- One Skylake (SKL) and one Kabylake (KBL) system used in our experiments.
- The SKL configuration is a desktop processor whereas the KBL configuration is an NUC processor.
- KBL configuration relative to SKL:
  - Half the CPU cores
  - Lower CPU frequency
  - Twice the number of execution units (EUs) in the iGPU
  - Additional eDRAM cache

OpenCL

- Collection of microbenchmarks written in OpenCL to characterise the hardware.
- OpenCL:
  - Popular parallel programming framework with wide device support.
  - OpenCL ⇔ CUDA equivalence:
    - Work Groups (WG) ⇔ Thread Blocks
    - Work Item (WI) ⇔ Thread
- Thread scheduling differences:
  - NVIDIA GPU:
    - A warp is a collection of scalar *CUDA threads*. The scheduler can schedule a ready warp from any of the thread blocks assigned to a Streaming Multiprocessor (SM).
  - Intel iGPU:
    - Each execution unit (EU) can run multiple *hardware threads* in Simultaneous Multithreading (SMT) fashion.
    - The compiler creates a stream of instructions from OpenCL work groups and work items, and the scheduler maps it to hardware threads.

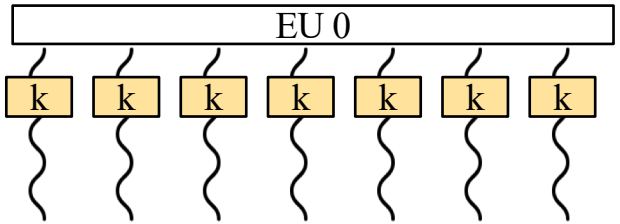
OpenCL Thread Mapping


```
__kernel void
compute_gflops_single(...){
 ...
 for(unsigned i =0 ; i < 512; i++){
            FUSE(x, y);
            FUSE(x, y);
            ...
 }
...
}
```

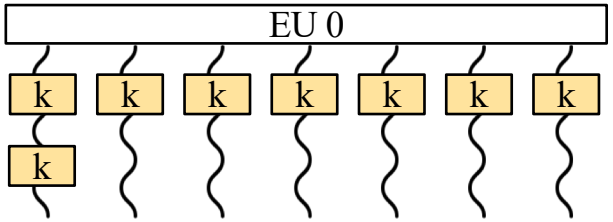
Compile

```
k
mov
mad
mad
mad
...
...
cmp
jmpi
```

Each ins is variable width SIMD



(a) WGs <= hardware threads



(b) WGs > hardware threads

OpenCL Work Groups (WGs):

- Map to a new hardware thread, up to max hardware threads
- Can be split and mapped to multiple hardware threads (subject to barrier and shared memory constraints)
- If WGs > max hardware threads, the kernel invocations for the remaining WGs are stacked at the end of existing hardware threads

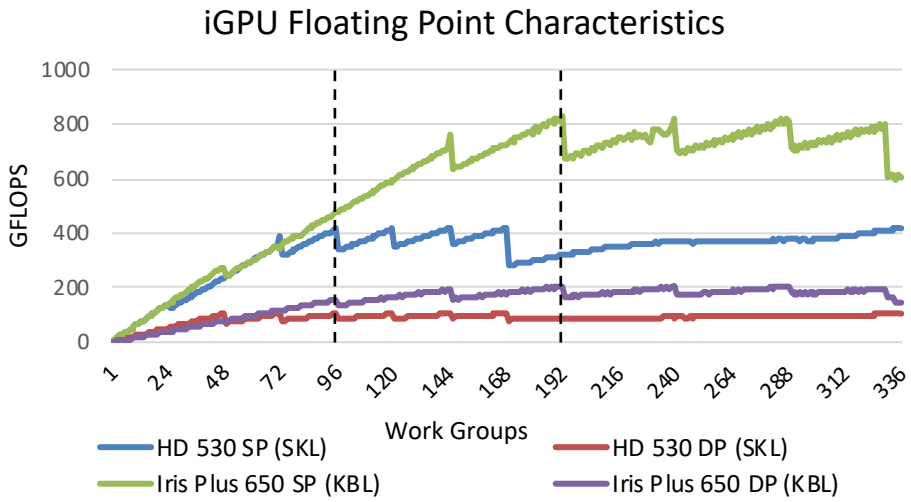
Floating Point (FP) Characteristics

- Use Fused Multiply and ADD (MAD) instructions for peak floating point throughput.
- Each SIMD unit is 4-wide and pipelined with a peak issue rate of 1 MAD instruction per cycle.

$$Peak\ GFLOPS = EUs\ * \frac{SIMD\ units}{EU} * \frac{Flops\ per\ cycle}{SIMD\ unit} * Freq\ GHz$$
$$= 24\ * 2\ * 8 * 1.15$$
$$= 441.6\ Single\ Prec\ GFLOPS \text{ (SKL)}$$

- The KBL throughput is $2x$ due to twice the number of EUs
- The double precision throughput is $\frac{1}{4}x$ single precision throughput.

Floating Point (FP) Characteristics



- Each WG comprised of 32 WIs
- SP = Single Precision
- DP = Double Precision

- 4 hardware threads per EU sufficient to hide the FP latency of 4 cycles.
- At 96 and 192 WGs, for SKL and KBL resp., the EU occupancy is 4 threads, which is sufficient for peak throughput.
- Cyclical dips caused by workload imbalance when certain hardware threads are on the critical path

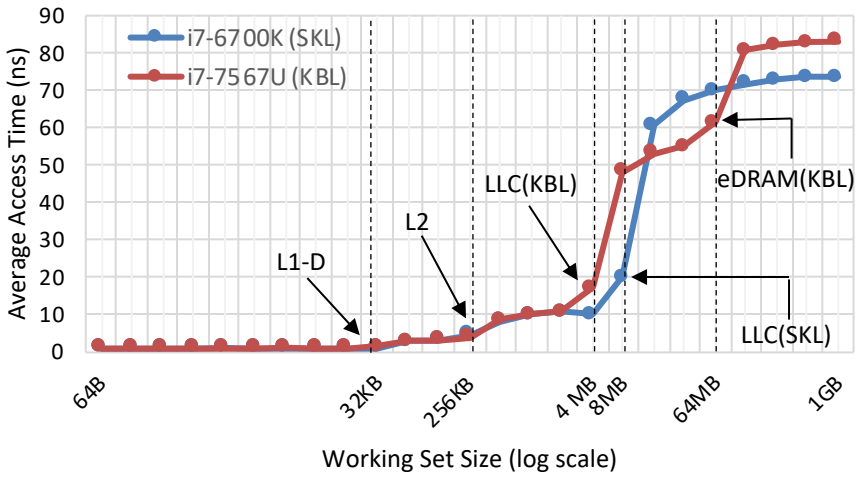


Memory Hierarchy Latency

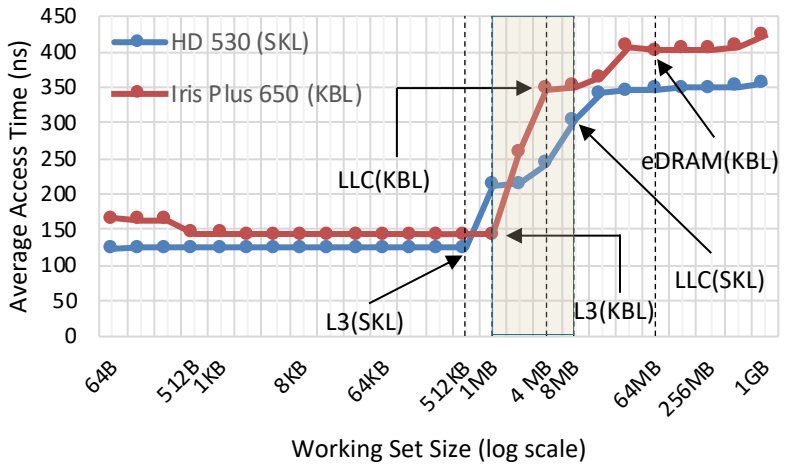
- CPU Mem Hierarchy:
  - L1D -> L2 -> LLC -> eDRAM (KBL only) -> DRAM
- GPU Mem Hierarchy:
  - L3 -> LLC -> eDRAM (KBL only) -> DRAM
  - L1 and L2 are special sampler caches which we don't use in this experiment
- Use a single threaded pointer chasing kernel to measure latency

Memory Hierarchy Latency

CPU Latency Characteristics



iGPU Latency Characteristics

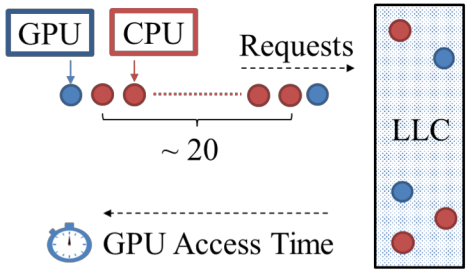


- GPU's latencies are generally higher than CPU's, even for the same (shared) levels.
  - GPU's clocks are lower (peak 1.1 GHz for GPU v/s 4 GHz for CPU). The difference in raw cycles is lower compared to ns
  - High Latency could be an indication of interconnection network's delays
- GPU's access times increases even in the LLC regions
  - Theory: The GPU is not able to make full use of the LLC capacity

Cache (LLC) Sharing Effects

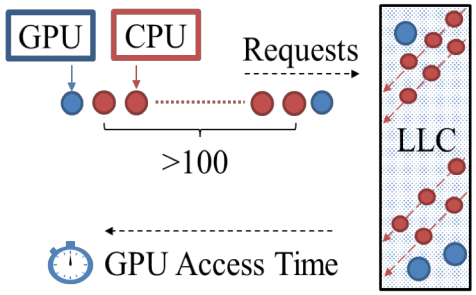
- Shared LLC accesses show different characteristics for the CPU and the GPU.
- Understand the effects of potential interference b/w CPU and GPU
- Measure the latency from one, while varying the interference from the other. Four cases:
  - Single Threaded (ST) interference from CPU; measure GPU's latency
  - Streaming Interference (STRM) from CPU; measure GPU's latency
  - Single Threaded (ST) interference from GPU; measure CPU's latency
  - Streaming Interference (STRM) from GPU; measure CPU's latency

CPU Interference Measure GPU



(a) 20 intervening CPU requests b/w GPU requests

GPU Working Set	(Pointer Chasing Interference) CPU Working Set									
	0 MB	1 MB	2 MB	3 MB	4 MB	5 MB	6 MB	7 MB	8 MB	9 MB
1 MB	213.54	206.77	207.03	208.34	213.66	216.56	228.67	251.68	217.65	210.73
2 MB	215.01	209.33	210.31	215.04	219.57	236.35	249.27	269.91	278.53	275.69
3 MB	231.21	224.12	226.19	234.33	252.32	267.49	280.07	292.09	293.06	293.18
4 MB	246.6	240.19	247.03	262.01	275.24	290.32	298.37	302.39	302.1	302.45
5 MB	258.73	258.04	271.67	281.84	294.99	301.75	307.4	308.22	308.21	308.26
6 MB	273.68	270.74	290.09	296.31	305.99	309.7	312.04	312.57	311.94	312.21
7 MB	286.89	287.13	301.84	305.76	311.56	315	315.53	315.36	314.85	315.09
8 MB	304.29	303.37	310.58	314.81	315.68	317.36	317.55	317.49	316.79	317.07
9 MB	318.28	312.89	314.9	318.84	318.48	319.44	319.3	319.21	318.53	318.65

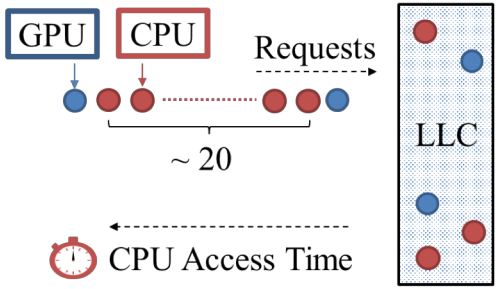


(b) More than 100 intervening CPU requests b/w GPU requests

GPU Working Set	(Streaming Interference) CPU Working Set									
	0 MB	1 MB	2 MB	3 MB	4 MB	5 MB	6 MB	7 MB	8 MB	9 MB
1 MB	213.54	207.64	209.24	209.84	215.48	226.19	234.33	286.03	320.9	339.46
2 MB	215.01	208.24	210.82	217.44	227.23	238.55	257.8	305.11	342.82	355.22
3 MB	231.21	224.43	238.79	244.45	254.61	273.66	288.92	343.72	378.46	384.57
4 MB	246.6	236.95	256.38	263.73	284.87	291.62	304.93	364.87	392.38	393.8
5 MB	258.73	260.2	273.42	285.55	298.02	304.65	312.19	368.42	404.14	412.82
6 MB	273.68	278.19	289.93	299.94	310.77	312.49	316.84	381.05	416.06	411.42
7 MB	286.89	293.29	304.67	312.66	318.3	319	319.83	387.87	417.56	424.25
8 MB	304.29	302.3	316.44	319.69	320.6	321.25	322	389.3	419.35	428.99
9 MB	318.28	312.4	321.94	322.53	322.88	323.53	323.86	324.09	437.51	440.37

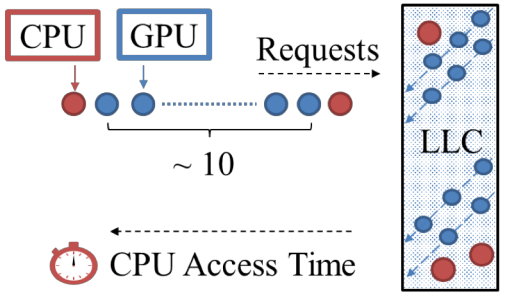
- The GPU's access time in the LLC region is relatively unaffected by the type of CPU interference
  - CPU's interference is not on the critical path.
- The GPU's access time increases dramatically when there is streaming interference from the CPU due to contention at DRAM

GPU Interference Measure CPU



(c) 20 intervening CPU requests b/w GPU requests

CPU Working Set	(Pointer Chasing Interference) GPU Working Set									
	0 MB	1 MB	2 MB	3 MB	4 MB	5 MB	6 MB	7 MB	8 MB	9 MB
1 MB	9.8	10	9.8	9.75	9.8	10	9.96	9.88	9.76	9.75
2 MB	10.91	10.74	10.45	10.48	10.45	10.76	10.69	9.44	9.21	10.49
3 MB	9.23	10.97	10.71	10.7	10.71	9.51	9.51	9.61	9.24	9.25
4 MB	9.75	10.04	9.88	9.98	9.88	10.04	9.7	9.39	9.12	9.45
5 MB	9.97	11.51	10.39	10.1	10.39	10.66	9.45	9.81	9.94	9.61
6 MB	10.36	10.67	11.61	11.69	11.61	10.5	10.72	9.26	9.49	10.05
7 MB	11.69	21.74	16.83	17.19	16.83	17.15	13.24	12.89	11.25	13.3
8 MB	20.58	35.26	35.27	35.23	35.27	35.51	33.72	34.63	33.61	32.78
9 MB	31.96	45.67	50.19	48.85	50.19	50.01	49.55	51.38	50.29	49.71



(d) 10 intervening GPU requests b/w CPU requests

CPU Working Set	(Streaming Interference) GPU Working Set									
	0 MB	1 MB	2 MB	3 MB	4 MB	5 MB	6 MB	7 MB	8 MB	9 MB
1 MB	9.8	10.42	10.22	10.74	12.78	18.5	29.86	42.14	67.07	88.09
2 MB	10.91	10.9	11.92	11.02	18.02	23.95	38.86	58.51	78.91	98.46
3 MB	9.23	9.64	9.[illegible]	10.96	25.63	34.37	46.18	64.34	80.9	96.3
4 MB	9.75	10.11	12.16	[illegible]	[illegible]	41.39	53.71	68.46	82.62	98.4
5 MB	9.97	10.16	16.69	31.38	39[illegible]	50.23	61.39	72.67	82.22	98.68
6 MB	10.36	12.56	27.17	40.22	49.06	56.91	65.35	75.33	83.11	99.48
7 MB	11.69	18.37	35.44	48.84	55.02	60.58	68.94	77.9	83.91	99.64
8 MB	20.58	35.07	46.75	54.51	58.11	62.9	70.37	78.58	83.37	99.71
9 MB	31.96	46.48	53.45	59.1	60.52	64.61	71.18	79.67	83.97	99.59

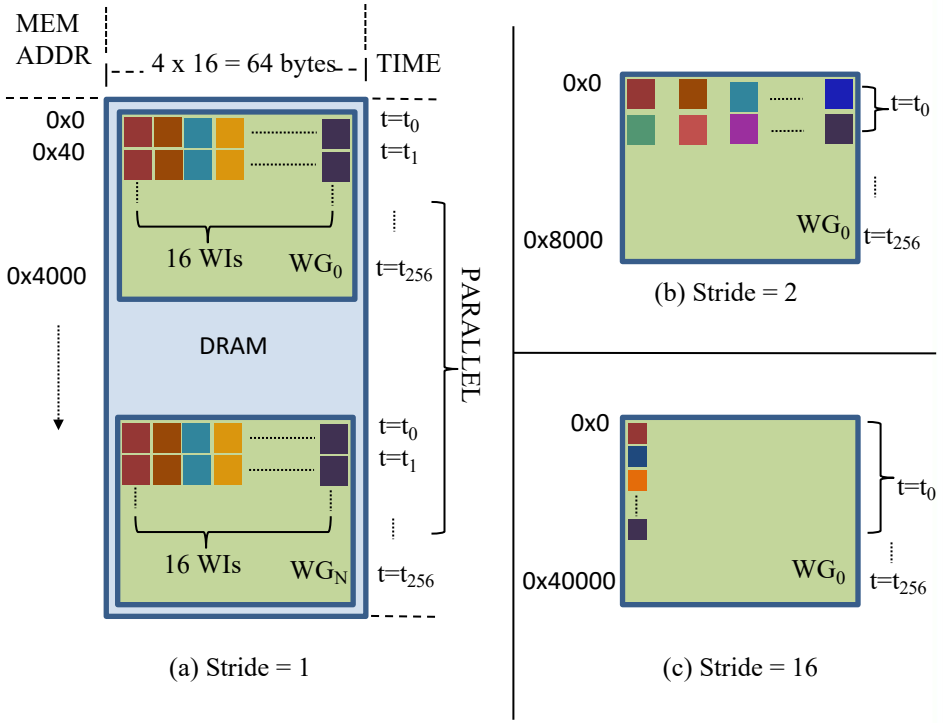
- The CPU is unaffected by single threaded GPU interference
- The CPU's latencies steadily increase along the lower diagonal in presence of streaming interference from the GPU

Memory Coalescing and Bandwidth

- Memory coalescing: Combining multiple memory accesses into a single transaction
- Coalesced memory requests show better performance (low latency)
- Uncoalesced memory requests will generate more memory requests → Higher raw bandwidth. Wasted bandwidth in practice.

Experimental Setup:

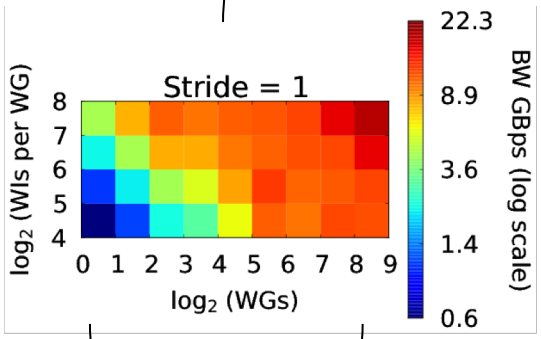
- Vary WGs, WIs and stride
- No reuse, no sharing

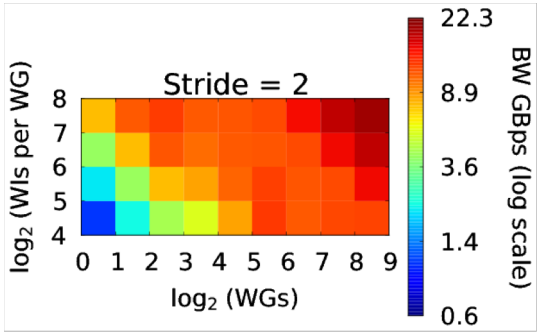


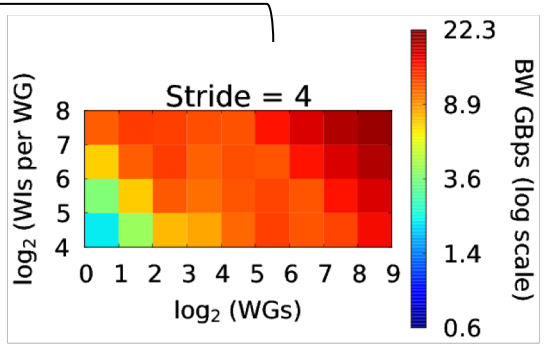
Higher stride distance → Less coalescing opportunity

Memory Coalescing and Bandwidth

Higher stride -> More requests -> Fewer WGs & WIs to saturate B/W

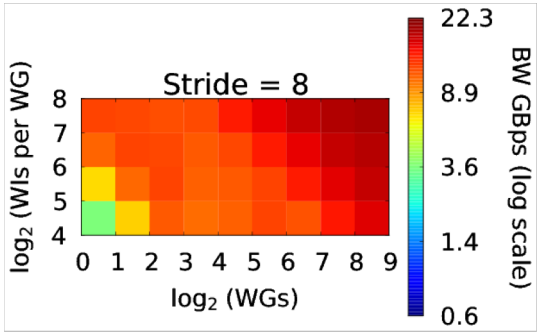


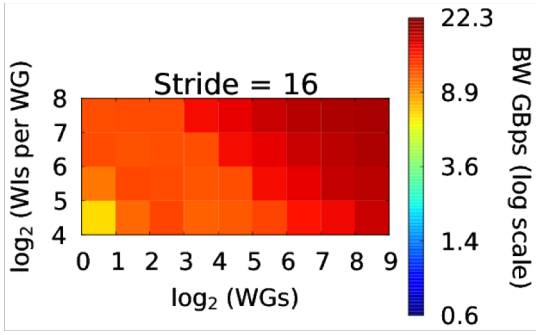




More WGs -> More parallel requests -> Higher B/W

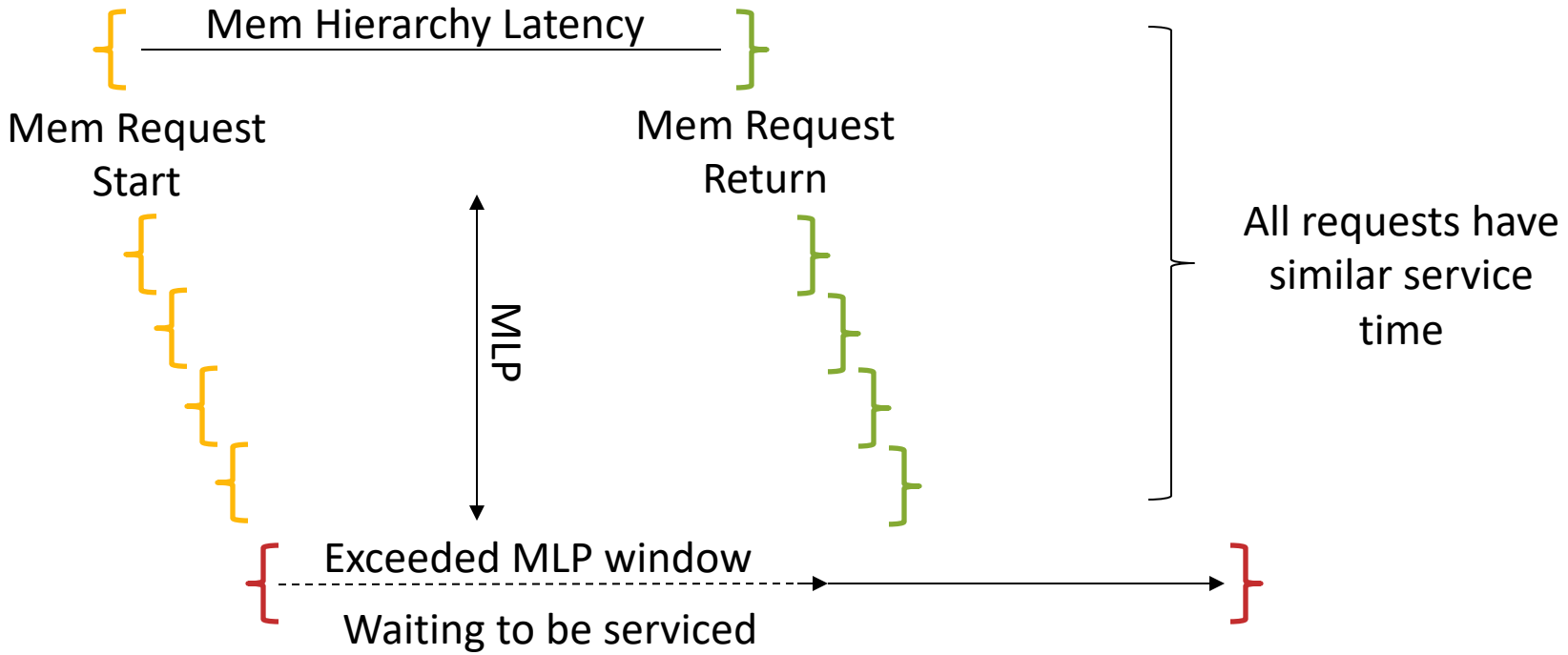
More WIs ->
More parallel requests ->
Higher B/W





Georgia Tech comparch

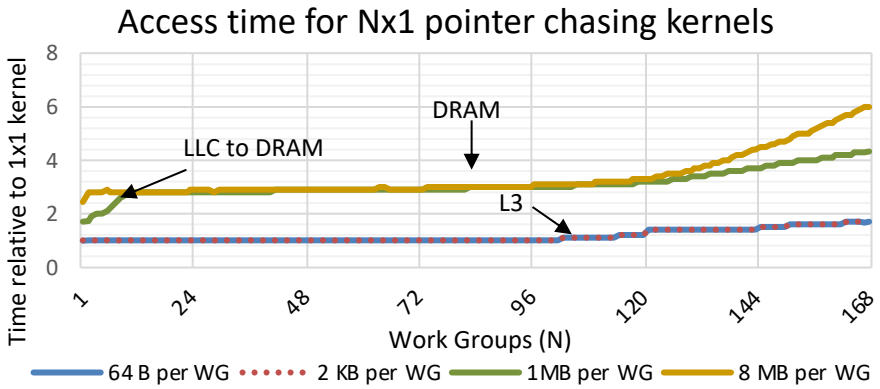
Memory Level Parallelism (MLP)



- MLP = Number of in flight memory requests
- Hardware's ability to handle concurrent memory requests is an important consideration in modelling and simulation

Memory Level Parallelism (MLP)

- Use work groups with a single work item (Nx1)
  - Increase work groups one at a time
  - Each work group generates dependent (pointer chasing) requests. No overlap between requests of different work groups

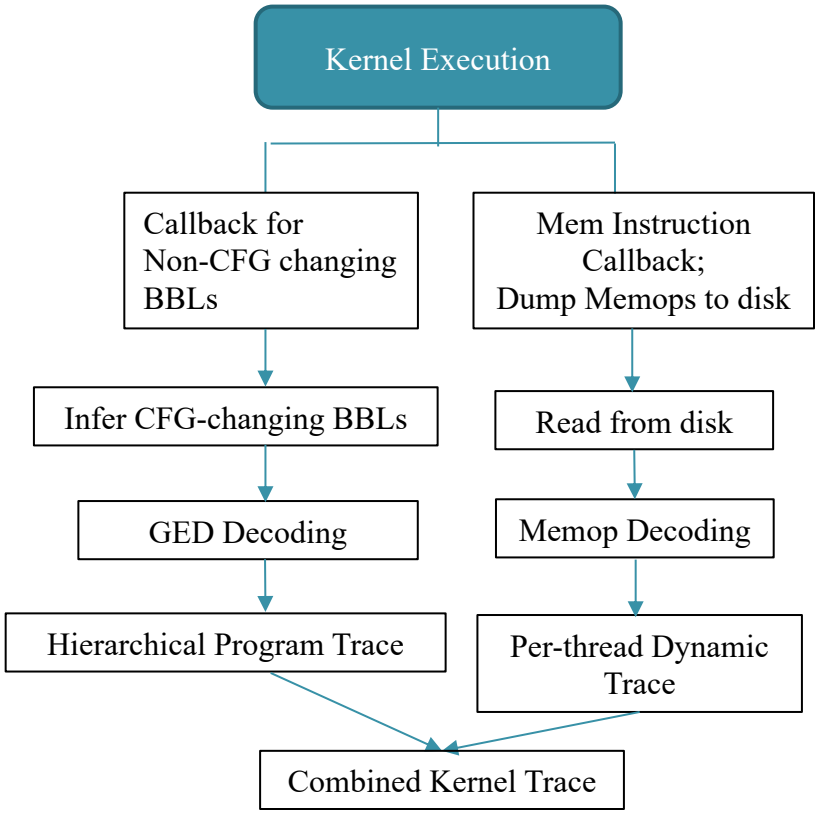


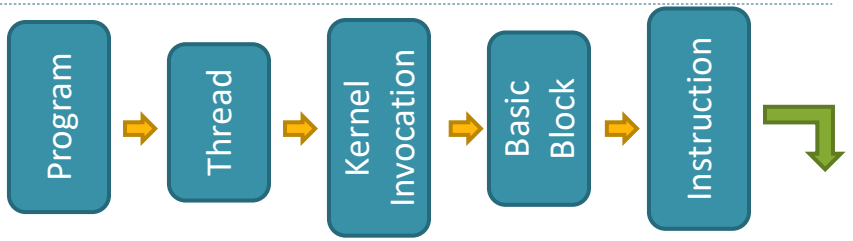
- Almost no increase in time up to 100 work groups
- Similar behaviour for different working set sizes
- The system has MLP of about 100 at all levels of the hierarchy

Trace Generation

- We built our trace generation framework by leveraging the GT-Pin toolkit
- GTPin:
  - Binary instrumentation framework for Intel GPUs developed by Intel. Analogous to Pin for Intel CPUs.
- Trace Generation Goals:
  - OS agnostic:
    - We tested GT-Pin + tracegen on GNU/Linux and Windows
  - Multiple Programming Frameworks
    - We use OpenCL kernels in this work. Extension to DirectX, OpenGL etc. in future plans
  - Does not need the source code of apps
    - Binary instrumentation at the native ISA layer
  - Simulator agnostic
    - We use Google Protobuf format for traces to encourage interoperability and reuse

Trace Generation



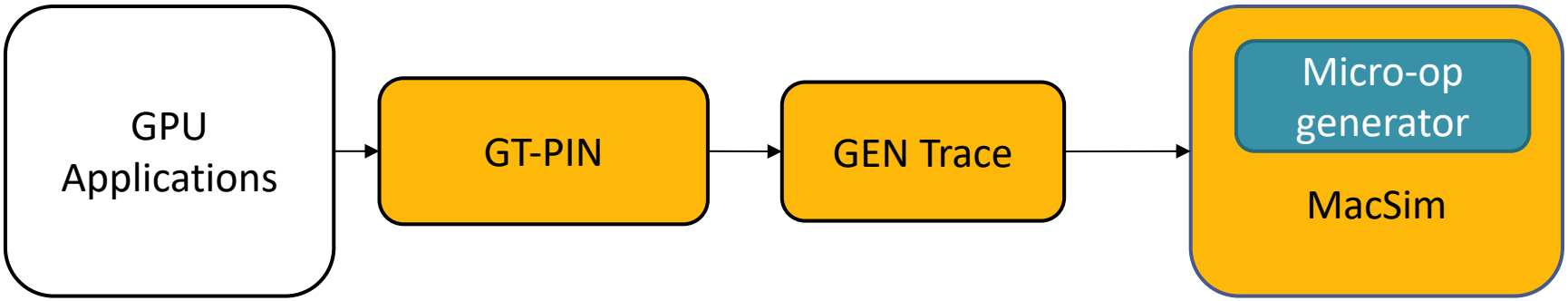



```
opcode:              GED_OPCODE_send
exec_size:           16
dst_reg_file:        GED_REG_FILE_GRF
dst_data_type:       GED_DATA_TYPE_w
src0_reg_file:       GED_REG_FILE_GRF
src0_data_type:      GED_DATA_TYPE_uq
src1_reg_file:       GED_REG_FILE_INVALID
src1_data_type:      GED_DATA_TYPE_INVALID
dst_chan_en:         16
dst_addr_imm:        -1
ins_uid:             "4_3_34"
src0_reglist:        "39_0", "39_1", "39_2", ...
...                  ...
...                  ...
mem_addr:            FF00007F04, FFB886CC88, ...
```

Sample instruction in the trace

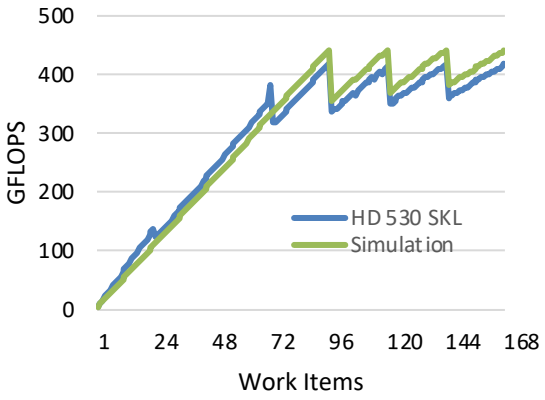
- Trace generation proceeds along two different paths for basic blocks and memory which are merged in the end
- We additionally do some processing while decoding like flattening 2D register accesses to a list

Simulator

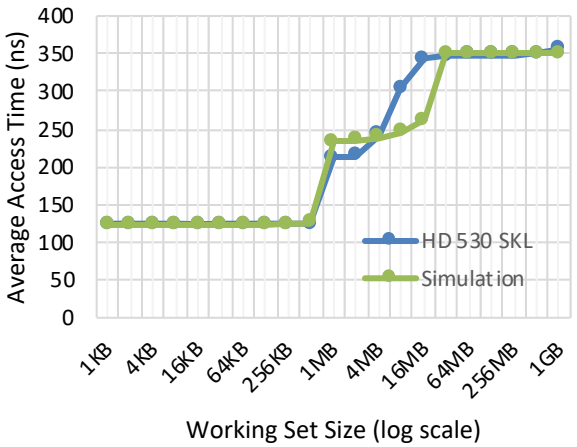

GPU Applications
GT-PIN
GEN Trace
Micro-op generator
MacSim

Evaluation

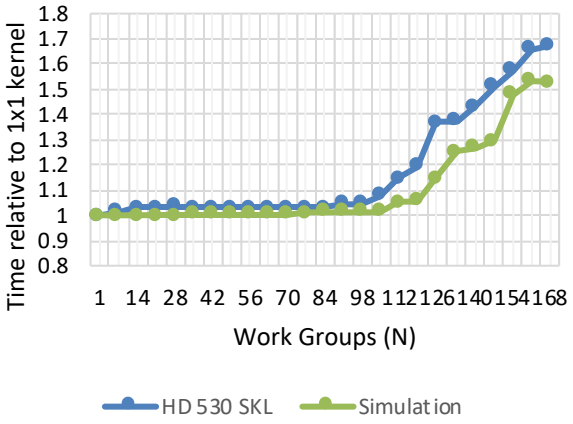
Floating Point Charcteristics



Latency Characteristics



Access time for Nx1 pointer chasing kernels



- We model the compute units, scheduler's behavior, and workload imbalance scenarios for FP workload faithfully.
- For the memory hierarchy's latency, the only deviation is in the LLC region, where the iGPU's time increases possibly due to the inability to utilize the full capacity. We leave this modelling for future work.
- Similar trends of MLP close to 100. We model this with structures such as the MSHR for caches.

Conclusion

- 4 threads per EU are sufficient to reach peak GFLOPs.
- The GPU's memory hierarchy latency is higher than the CPU's, even when accessing the same resources like the LLC.
- The GPU is unable to use the LLC's total capacity.
- The GPU's accesses to the LLC are relatively unaffected by CPU's interference whereas the CPU's LLC accesses are affected by streaming interference from the GPU.
- The GPU has an MLP of around 100 memory requests across the memory hierarchy.
- To the best of our knowledge, this is the first work that takes a detailed look at the Intel iGPU architecture, and provides a complete flow of modelling, trace collection and simulation.



Thank You !

Georgia Tech comparch