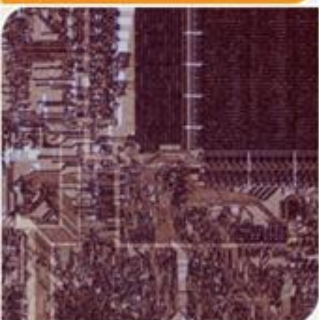


CS4803DGC Design and Programming of Game Consoles

Spring 2011

Prof. Hyesoon Kim



**Georgia
Tech**



College of
Computing

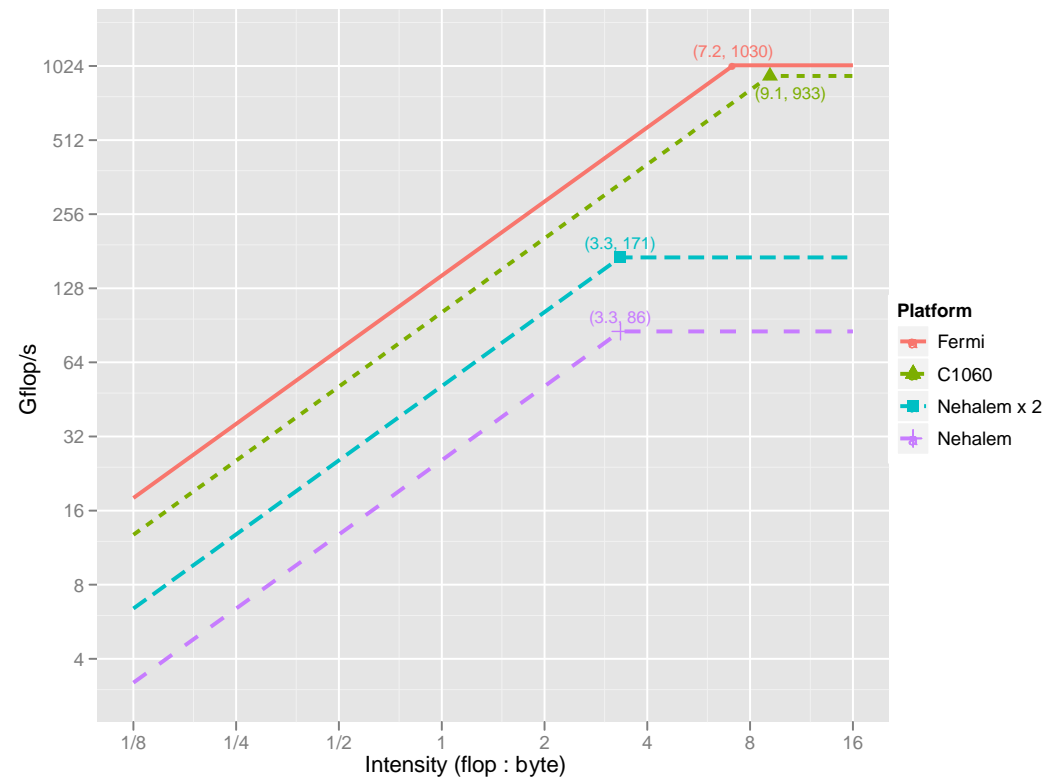


PERFORMANCE MODELING OF GPUS



Roofline Model

- Attainable GFLOP/sec
= min {Peak Floating-Point performance ,
Peak memory bandwidth X Operational Intensity}






Little's law

$$N = \lambda L$$

- N : mean number of tasks in system
- λ arrival rate
- L : latency
- Mean number of tasks in system = arrival rate x mean response time
- Q: Memory latency is 500 cycles. Memory requests are sent every 5th cycle, how many requests are in the memory system?
- $500 * 1/5 = 100$, on average 100 memory requests are in the system
- Or, every 5th cycle memory requests are sent and there are 100 memory requests are in the flight: what will be the memory latency?



Applying Little's law to GPGPU

- Memory latency is 500 cycles. Each warp (32 threads) generates 1 memory request every 5th instructions.
- How many warps do we need to hide memory latency?
 - Assume that we have only one core and as many as warps possible.
 - 
 - If there is a batch, the Number will be reduced by batch size.

Shebanow's limiter's theory



Limiter's theory: Hiding Latency

M. Shebanow, NVIDIA'10

- Principle:
 - Little's Law:

$$N = \lambda L$$

- N = “number in flight”, λ = arrival rate, L = memory latency
- Arrival Rate product of:
 - Desired execution rate (IPC)
 - Density of LOAD instructions (%)

- Use *batching*
- Group independent LDs together
- Modified law:

$$N = \frac{\lambda L}{B}$$

- B = batch size



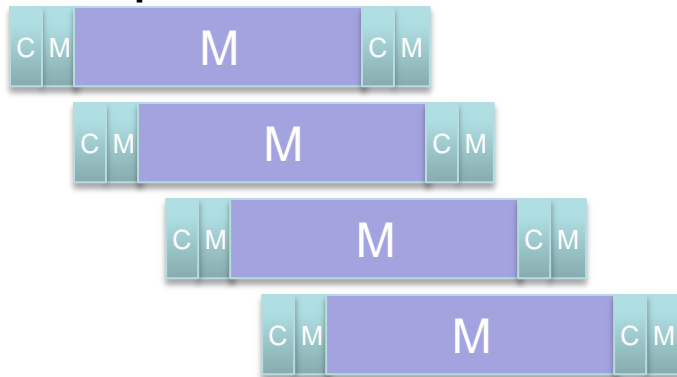
- LD, Dep inst, LD, Dep inst
- 1 warp

$$N = \frac{\lambda L}{B}$$

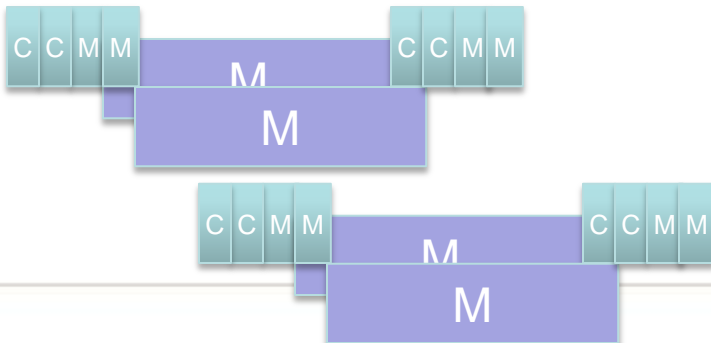
$L = 8, \lambda = 1/2,$



- 4 warps



- 2 warps, load hoist (batch size 2): How to increase batch size?





Mid-term

- Performance Modeling and analysis for G80 architecture
- Design decision based on benchmark characteristics
- Xbox 360 optimization techniques (just describe)
- Bring your calculator
- # of questions: $\sim = 3$



Design Decisions

- Clock frequency
 - Designing factors
 - Circuit technology
 - Memory latency, IPC
- Pipeline depth decisions
- Power consideration
- Area considerations
- SFU?
- SIMD unit and SIMD width?

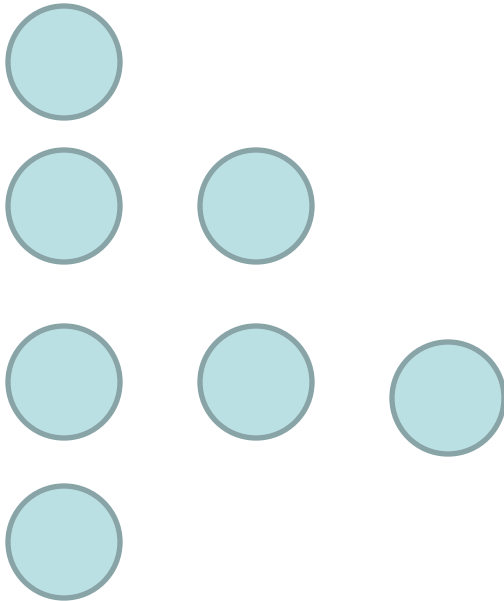


Benchmark Profiling & Design

- App1: ILP = 0.5, ILP<ooo> = 0.7 TLP= 4 (100% parallelizable), FP_MUL= 20% of instructions, ILP only FP= 32, cache hit ratio trend 256KB = 40%, 512KB=50%, 1M = 80%, 2M = 90%, 20% Mem insts
- App2: ILP=0.25, ILP <ooo> = 0.5 TLP = 8 (100% parallelizable) FP_MUL=5%, cache hit ratio trend 256KB = 30%, 512KB=30%, 1M = 50%, 2M = 50%, ILP only FP= 16, 20% Mem insts
- Budget :200 units (area), 1 level cache: latency (256K = 5, 512K=7, 1M = 9, 2M = 20} , mem latency = 100 cycle, pipeline depth = 9 cycles + execution latency (1 for INT)
- ooo-core w/o cache:2w: 40, 2w: 50 ooo-SMT core w/o cache: w2: 55
- In-core w/o cache: 1w:20, 2w: 30, in-core SMT w/o cache 2w: 35
- cache size 256KB = 20 units, FMUL(10 latency) 0.2 per 1FP, FMUL (2 latency) 0.5 per 1FP



ILP, Working Set Size



- Critical path = 4,
- ILP = $8/4 \approx 2$

CPU performance w/ different instructions



Instruction Type	Frequency	CPI
Integer	40%	1.0
Branch	20%	4.0
Load	20%	2.0
Store	10%	3.0

$$\text{CPU time} = \left(\sum_{i=1}^n IC_i \times CPI_i \right) \times \text{Clock cycle time}$$

Total Insts = 50B, Clock speed = 2 GHz

$$= (0.4 \times 1.0 + 0.2 \times 4.0 + 0.2 \times 2.0 + 0.1 \times 3.0) \times 50 \times 10^9 \times 1 / (2 \times 10^9)$$