Analyzing GPGPU Pipeline Latency

Michael Andersch, Jan Lucas,
Mauricio Álvarez-Mesa, Ben Juurlink
Embedded Systems Architectures, TU Berlin, Einsteinufer 17, 10587 Berlin, Germany

1. Overview

GPUs as we know them today are intrinsically throughput-focused devices designed to **hide microarchitectural latency** through heavy use of thread-level parallelism. Over the last few generations of commercial GPUs, throughput has increased substantially as a result of both architectural innovation and advancements in manufacturing technology. While the throughputs of each commercial GPU are well known, the same cannot be said for latencies of the hardware, such as instruction latencies or cache hit times. As we run or want to run both **throughput-limited** and **latency-limited** applications on such devices, this is problematic for both programmers and GPU architects, as the development of high-performance code and the design of newer and faster GPU architectures require an intricate knowledge of the state of the art. In this study, we investigate this state by performing a **GPU hardware latency analysis** on four subsequent generations of GPU architectures: NVIDIA Tesla, Fermi, Kepler, and Maxwell.

2. Arithmetic Pipelines

Operation	Tesla GT200	Fermi GF106	Kepler GK104	Maxwell GM107	Operation	Tesla GT200	Fermi GF106	Kepler GK104	Maxwell GM107
Integer & Logic					32-bit Floating Point				
ADD, SUB	24	16	9	6	ADD, SUB	24	16	9	6
MAX, MIN	24	18	9	12	MAX, MIN	24	20	9	12
MAD	120	22	9	13	MAD	24	18	9	6
MUL	96	20	9	13	MUL	24	16	9	6
DIV (unsigned)	608	286	141	210	DIV	137	1038	758	374
DIV (signed)	684	322	168	243	__fadd_*()	24	16	9	6
REM (unsigned)	728	280	138	202	__fmul_*()	26	16	9	6
REM (signed)	784	315	163	232	__fdividef()	52	95	41	34
AND, OR, XOR	24	16	9	6	__sinf(), __cosf()	48	42	18	15
SHL, SHR	24	18	9	6	__tanf()	98	124	58	49
__umul24()	24	38	18	19	__exp2f()	48	98	49	41
__mul24()	24	38	18	19	__expf(), __exp10f()	72	114	58	46
__usad()	24	20	9	6	__log2f()	28	46	22	35
__sad()	24	20	9	6	__logf(), __log10f()	52	94	49	40
__umulhi()	144	20	9	21	__powf()	75	143	62	49
__mulhi()	180	20	9	21	sqrt()	56	216	181	128

Table 1: Latencies of math datapath operations over four generations of NVIDIA GPUs.

Methodology

- Directed CUDA microbenchmarks + GT200 results from [1]
- Generate unrolled loop of dependent arithmetic instructions of desired type
- Encapsulate with timing code, i.e. reading the clock register
- Divide number of clocks measured by number of instructions in the loop

Results

- Just a handful of actual hardware datapaths
- Several operations realized by instruction sequences
- Significantly longer latencies than comparable CPU datapaths
- Trend: Arithmetic latencies have significantly decreased from Tesla to Maxwell

3. Memory Pipelines

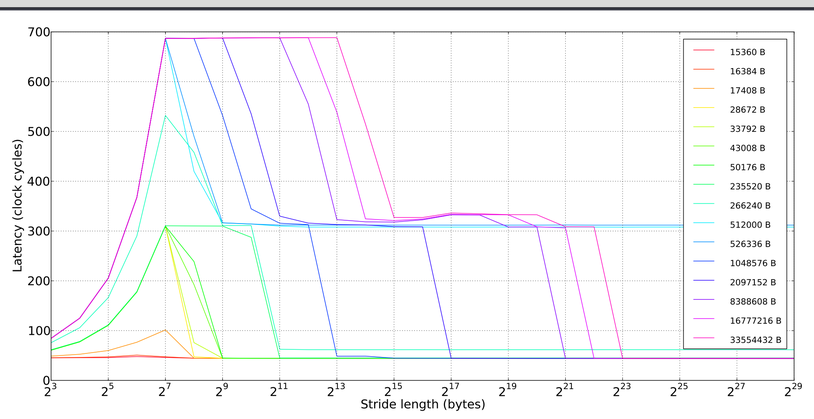

Figure 1: Global memory results on Fermi.

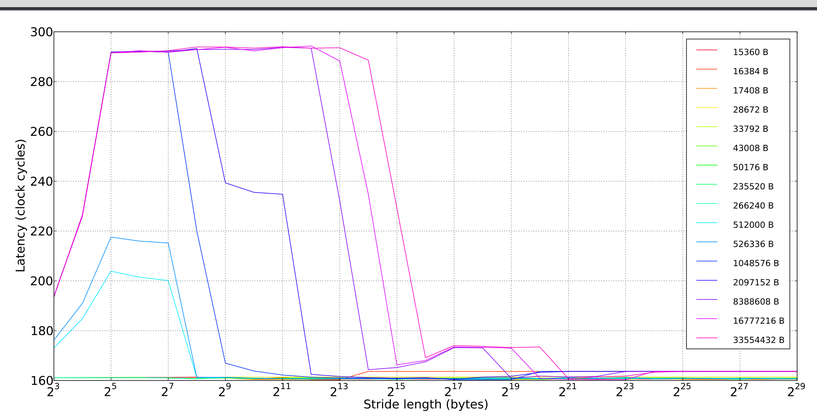

Figure 2: Global memory results on Kepler.

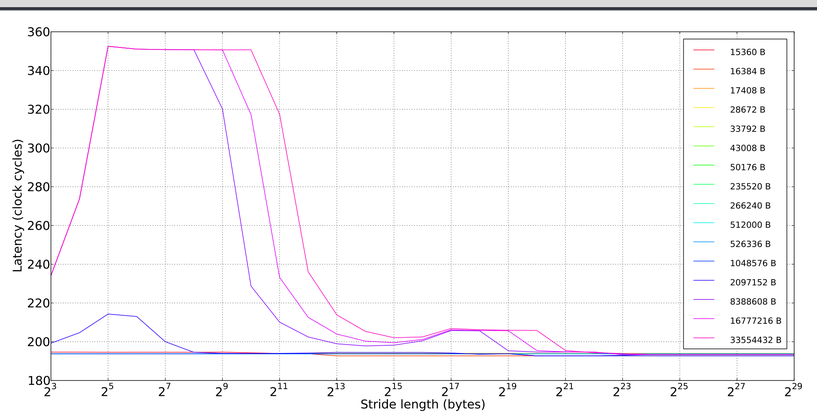

Figure 3: Global memory results on Maxwell.

Unit	Tesla GT200	Fermi GF106	Kepler GK104	Maxwell GM107
Global & Local Memory				
L1 D$	×	45	30	×
L2 D$	×	310	175	194
DRAM	440	685	300	350
Shared Memory				
SMEM	38	50	33	28
Texture Memory				
Texture L1 D$	261	224	105	92
L2 D$	371	435	215	172
DRAM	×	791	348	330
Fixed-function pipeline	×	106	48	(-20)
Constant Memory				
Constant L1 D$	56	52	42	28
Constant L1.5 D$	129	165	104	79
L2 D$	268	375	215	184

Table 2: Latencies of hardware units in the various memory pipelines over four generations of NVIDIA GPUs.

Methodology

- Directed CUDA microbenchmarks + GT200 results from [1]
- A single thread chases pointers through the desired memory space
- Stride and footprint varied, per-access memory latency in cycles measured

Global Memory

- Dramatic changes throughout the generations
- No caches on Tesla, local-only L1 cache on Kepler, no L1 cache on Maxwell
- Extremely large latencies compared to CPUs
- Trend: Minimum access latency has increased over the years

Other Memory Spaces

- Shared memory has not changed much between the generations
- Texture memory massively improved on Kepler, faster than global/local on Maxwell
- Constant memory shows per-cluster cache between core-private L1 and core-shared L2

4. Conclusions

- Conducted static latency analysis with a multitude of GPU processors
- 75% decrease in basic arithmetic latency over the last 6 years
- Global memory access minimum latency has increased in newer GPUs

5. Acknowledgement

The research leading to these results received funding from the EC's 7th Framework Programme, LPGPU project, grant agreement n° 288653. For more information about the LPGPU project, please visit `http://lpgpu.org`.

6. References

[1] Wong et al., *Demystifying GPU Microarchitecture through Microbenchmarking*, Proceedings of the International Symposium on Performance Analysis for Systems and Software (ISPASS), 2010.