Computer Design: Supervision 5

Daniel Chatfield

November 27, 2014

Lecture 17

This lecture continues the examination of GPUs and introduces multithreading. At the end, students should understand the basic architecture of a GPU streaming multiprocessor and how this can be programmed in CUDA.

1. Why do GPUs rely on multithreading as well as SIMD? What bottleneck does multithreading target?

   > It takes a very long time (perhaps hundreds of clock cycles) for a memory operation to complete. CPU architectures include two or three-level cache memory hierarchies to reduce memory latency. Whilst Nvidia's *Kepler* and more recently *Maxwell* architectures do include some caching (e.g. *256KB* on the *GK107* and *2MB* on the *GM107*) they are rarely effective as GPUs are designed for stream computing.
   >
   > Instead, GPUs tolerate this latency with a high degree of *multithreading*. When one warp stalls on a memory operation the multiprocessor control unit selects another ready warp and switches to that one.

2. How can a SIMD processor pipeline be modified to allow multithreading?

   > Not sure what this really means. Loading multiple pieces of data at once?

3. The warp scheduler chooses instructions to execute. What are its criteria for choosing and what other schemes could be implemented?

   > The warp scheduler selects an eligible warp and then depending on which architecture it then dispatches 1 instruction (for Fermi and older) or up to 2 independent instructions (for Kepler and Maxwell).

4. What are the types of memory available within a GPU and what are their relative advantages or disadvantages?

> Each SIMD land is allocated private memory for stack frame and spilling registers. Each multithreaded processor has a local memory that is not shared between processors and is allocated dynamically to thread blocks on creation. The processor local memory can be used for communication between threads.
>
> There is also the global GPU memory that is available across the GPU and to the host system.

5. What types of application best suit GPUs and how can you program to take advantage of the various forms of parallelism available?

> Any computation that requires the same calculation to be done to lots of data is *embarrassingly parallel* and thus well suited to a GPU. To make best use of the GPU, computations should be programmed such that as many of the SIMD lanes are active.

Lecture 18

The final lecture gives an overview of the directions that computer architecture is heading. The aim of this lecture is to give students a perspective on the challenges of the future and an introduction to some of the research that is tackling it.

1. Why is energy efficiency the "new fundamental limiter of processor performance", as Borkar and Chien say?

> In the past, when the transistor size shrunk and more transistors could be put on a die the voltage could also be lowered (keeping the overall power consumption broadly the same). Going forward this will not be possible as the leakage will be too high and thus the energy requirements will be prohibitive.
>
> The ARM CTO described this as an era of "dark silicon" where a proportion of a chip (potentially a large proportion) at any given time must be idle to meet power (and heat) requirements.
>
> Transistor size is no longer the determining factor in CPU speed and innovation in micro-architecture will be the driving force behind performance gains by making parts more energy efficient.

2. Describe ARM's big.LITTLE system.

ARM's *big.LITTLE* system is a power saving technology aimed at the mobile market where there is a strong demand for both performance and energy efficiency. It combines a high-performance core with an energy-efficient core, by keeping track of load history during execution it can anticipate the performance needs of the thread next time it runs and runs it on the relevant core.

(a) What's the point of having two types of core and when might they be used?

One core provides high performance (would be used when rendering a webpage) and the other good energy efficiency (would be used when doing some not very demanding task like writing a text message).

(b) Would it be useful to have a third core type and, if so, what would its characteristics be?

You could argue that an ultra energy efficient core that was only used for really low performance (but long running) things like tracking motion would be beneficial but you could also argue that it would be better to have that as a co-processor (like the M7 in the iPhone 5S).

3. When might it make sense to implement functionality in a specialised accelerator rather than within a general purpose core?

If it is used often and the speedup (or energy efficiency) is significant. An example is video encoding - when implemented as a specialised accelerator it is very efficient as the data can just stream through, without this it would not be possible to achieve 15+ hours video playback on an iPad.

4. What types of application domain might benefit from approximate computing?

Audio and video computation does not need to be exact as humans tend to be quite forgiving of small distortions. If the colour of one in every million pixels was slightly off then it would not even be noticeable.

Case study

For a processor or *system-on-chip* of your choice (perhaps one you own), try to find the following:

The Apple A7 is a 64-bit SoC designed by Apple and used in the iPhone 5S. It was the first 64-bit ARM CPU to ship in a consumer smartphone.

1. Its main components

Dual core CPU An Apple designed 1.3–1.4 GHz *ARMv8-A* processor called Cyclone.

GPU A *PowerVR G6430* from Imagination in a four cluster configuration supporting OpenGL ES 3.0.

Image processor A dedicated image processing chip for reducing noise, sharpening and image stabilizing.

Secure enclave Cryptographic chip that stores the data from the Touch ID fingerprint sensor. The security is protected by ARM's TrustZone/SecurCore technology and whilst the firmware can be updated, doing so automatically wipes the flash.

Motion coprocessor The motion coprocessor is a separate (not actually on the chip so strictly speaking not part of SoC) low power ARM based processor that services the accelerometer, gyroscope and compass. It allows the phone to be continually recording this data and make it available to apps and Apple health.

2. Which instruction set(s) and extensions it supports

It supports the AArch64 and AArch32 instruction sets with TrustZone.

3. The core's pipeline structure, including numbers and types of functional units and number of available registers

The A7 features a 13-stage pipeline and is capable of dispatching up to 6 instructions per cycle. Co-issuing FP and NEON instructions are limited. The pipeline for each core is 3-wide and supports out of order execution. Each core has 4 integer ALUs, 2 load/store units, 2 branch units, 1 indirect branch unit and 3 FP/NEON ALUs.

Apple's LLVM commits indicate a massive 192 entry reorder buffer that lets the CPU reorder instructions to make them more efficient.

4. Details of the memory hierarchy

> The SoC contains 1GB of DRAM with an L1 cache consisting of 64KB for instructions and 64KB for data. There is a 1MB L2 cache and a 4MB L3 cache.
>
> The latency on the L1 and L2 caches are 3 and 10 clock cycles respectively (the latter being a huge improvement on the previous generation).

5. What it does to reduce the effect of control and data dependencies

> The penalty for mispredicting a branch is between 14 and 19 cycles (it varies). As already mentioned the CPU pipeline reorders instructions (from a huge buffer). The branch predictor is much better (even caused a lawsuit from University of Michigan).

6. Its support for parallel execution and memory consistency (if any)

> Is previously mentioned the CPU supports issuing up to 6 instructions simultaneously per core and supports a limited set of NEON instructions (SIMD instructions).

7. What it can do to improve performance and/or power efficiency

> The only real power efficiency improvements over the previous generation are a result of race-to-sleep (by being faster it spends less time working and more time idle) and from the use of the M7 motion co-processor.

8. Its clock speed and fabrication process (e.g. 28/40/65 nm)

> Its clock speed is 1.3–1.4 GHz and is fabricated at 28 nm.

9. How it has been tailored to its target market, and the compromises which had to be made

> All the components are well suited to being in a mobile phone (e.g. image processor, GPU etc.)

10. How it improved on its predecessor (if any) and how it was improved upon by its successor (if any)

> Massive performance improvement (2x) and 64 bit architecture support.