Vectors and Streams

Lecture #14: Monday, 15 May 2000
Lecturer: Melvyn Lim
Scribe: Jinyung Namkoong

References

[1] Scott Rixner, William J. Dally, Ujval J. Kapasi, Brucek Khailany, Abelardo Lopez-Lagunas, Peter R. Mattson, John D. Owens, "A Bandwidth-Efficient Architecture for Media Processing", Proceedings of the *31st Annual International Symposium on Microarchitecture*, November 1998.

[2] Christoforos E. Kozyrakis, Stylianos Perissakis, David Patterson, Thomas Anderson, Kriste Asanovic, Neal Cardwell, Richard Fromm, Jason Golbus, Benjamin Gribstad, Kimberly Keeton, Randi Thomas, Noah,Treuhaft, Katherine Yelick, "Scalable Processors in the Billion-Transistors Era: IRAM", *Computer, Vol 30 Issue 9*, September 1997.

[3] Keith Diefendorff, "Sony's Emotionally Charged Chip", *Microprocessor Report, Vol.12 No.14*, October 26, 1998.

1 Introduction

Up to now, we have only looked at techniques that exploit instruction level parallelism to increase single-thread processor performance. The instruction level parallelism dictates how many instructions can be issued at the same time, and is limited by the amount of data dependencies among instructions. (Another important bottleneck is control dependencies, which call for branch predictions, instruction prefetching, etc. But for the purpose of this discussion, we'll ignore this factor.) Closely related to the instruction parallelism is the data parallelism, which dictates whether each element of the data set for a given operation is independent of the other elements in the data set. When such data paralellism exists, it can be readily converted to the instruction level parallelism, and the combination of these two kinds of parallelism directly affects how many 'useful' instructions can be issued (and retired) each cycle.

The sections of code that exhibit data parallelism can be sped up almost linearly with the number of execution units in the processor. Since the data set is known to be parallel, (which means the instructions are known to be parallel as well,) complicated

control logic to detect and handle dependencies is not needed. Since execution units are relatively cheap, duplicating them to get almost linear speedup in performance is a very attractive idea.

There are three flavors in exploiting this data level parallelism:

Vector Processor takes vectors as input operands and performs the same primitive operations on each element of the vectors.

Stream Processor similar to a vector processor, but can perform complex operations on the vectors (now called streams.) Can be thought of as performing a subroutine on very long streams.

VLIW Perform multiple operations at the same time. Compiler schedules instructions into available execution units.

Also, there are two different approaches in implementing these vector/stream machines:

Multiple Pipelines Multiple independent execution cores can be made, all executing the same instructions, but with different data sets.

Pipeline Parallelism As always, pipeline can be substituted for multiple parallel units.

2 Imagine [1]

2.1 Imagine Architecture

The Imagine Processor is organized around a large Stream Register File (SRF), which has one port, but can simulate multiple ports because an entire line consisting of 32 words can be read or written every cycle. The execution core consists of eight independent clusters, each one comprised of a set of integer and floating point ALUs, as well as other units used for intra- and inter-cluster communication.

There was some confusion about how all the clusters actually work together. It is important to note that all the clusters work in parallel, executing the same instruction, only with different data.

2.2 Memory Hierarchy

The Imagine architecture realizes that there is typically low reuse of data (temporal locality) in typical media applications, because the data set for such applications is quite large, and at the same time, these applications have very high demand on memory bandwidth. It proposes an efficient memory hierarchy where most of the bandwidth requirement is served where it's least expensive, while the expensive bandwidth (that goes to external memory) is minimized.

Lowest (least expensive) in the memory hierarchy is the local register files. Each of the ALUs in each cluster has a small set of registers at each of its input operand ports. In all, each cluster has 17 such LRFs, each of which is 16-words deep. The next level in memory hierarchy is the SRF, which is a large, high bandwidth (achieved through large internal line width) central register file. And the highest (most expensive) memory is the DRAM outside the Imagine core.

The SRF is similar to a cache in many ways. The difference is that it can read a lot of words at the same cycle, and since the data access is guaranteed to be sequential, almost all fetches are useful. That is, when a stream is read from external memory, only the useful words are read, and in particular order, so that no space (execpt for internal fragmentation) in the SRF is wasted for unuseful fetches. Then, since these streams are read out in sequence, and in explicit programmer's control, there's no danger of corruption.

Sometimes the data set for an application can exceed the size of the SRF. Imagine handles this kind of case by strip mining, partitioning large data sets to chunks and doing operations on each of these chunks separately. Since memory operation and computation can be overlapped, strip mining can be done quite efficiently. In contrast, conventional caches will not work well in this kind of case, possibly leading to thrashing.

2.3 Stream Programming Model

An Imagine application is written as a sequence of smaller tasks, called kernels. Basically, a host processor loads Imagine with a kernel, runs input streams through Imagine, then switches to the next kernel, runs data stream through it, and so on. In this way, data streams are passed through a sequence of kernels. This is suited for media applications, since there exists a lot of data parallelism in such applications, and the overhead of loading and changing kernels is amortized by large stream sizes. Also, the memory ops and compuation can overlap, with the goal of keeping all the units busy at all times. In the best case, there's almost no overhead to kernel changes.

2.4 Modern VLSI

Modern VLSI technology exhibits the following characteristics, which become more and more pronounced:

- The number of transistors that can fit on a chip is growing rapidly, increasing the budget for extra memory and execution units
- Global communication is already critical , and will become more so as wire delays become more important.
- Memory has been a major bottleneck for a long time , and it continues to grow in importance.

S he sream processor is well suited to combat these bottlenecks and continue to convert higher transistor counts to more performance. It exploits modern VLSI by an efficient memory hierarchy. In contrast, conventional architectures suffer from too much global register file accesses, which creates a major bottleneck. Imagine avoids this problem by satisfying much of the bandwidth with local register files.

3 IRAM [2]

In this paper, the authors present the IRAM approach from Berkeley, which attempts to combine processor and DRAM memory on chip, greatly alleviating the memory bandwidth bottleneck. The main idea is as follows: As processor performance increases, it places greater demand on the memory system, requiring higher bandwidth and low latency. Since going off-chip is probably the most expensive and slow communication bottleneck, putting the processor core and memory on the same chip will dramatically improve the memory system performance, by offering low latency and high bandwidth.

Vector computers have been expensive, largely because it is for a performance driven niche market, and also because of exotic packaging designed for supplying high bandwidth from the memory to the processor. IRAM claims its high memory bandwidth is well suited for a vector processor approach. Along with a general purpose CPU, it has a vector unit with multiple pipelines. It claims that vectorizing compilers have been well researched, and thus it'll provide an easy upgrade step.

A redundant vector pipeline is provided for better yield. This is in sync with the prevalent DRAM practice of including redundant columns so that one bad column will not cause the whole chip to be unusable. Yield is a big problem in such merged architectures. Adding logic can lower the yield greatly.

There have been many billion transistor architectures that have been proposed, and most of these architectures devote most of those transistors to caches. The distinction that IRAM claims is that since IRAM can use these on-chip memory as the main memory, instead of caches which really keeps redundant copy of the data that can be found elsewhere in the system, this is a more efficient and in a way elegant solution. A potential problem with using the on-chip memory as the main memory is upgradability, since it's not possible to add more on-chip DRAM once it has been fabricated. A solution that the paper suggests is to back up the on-chip DRAM with commodity off-chip DRAM, and swap pages between them as needed. This is almost like using the on-chip DRAM as caches. The difference here is that the on-chip memory do not contain duplicate copies of the data that's on the off-chip DRAM. But here a difficulty would be to keep track of locations of data that may move between on-chip and off-chip data. So, it'll probably have to end up using the on-chip DRAM as caches after all.

Other benefits this paper claims include higher clock frequency, lower power, smaller pacakage, and superior performance.

4 Sony's Emotion Engine[3]

4.1 Emotion Architecture

Emotion is a new processor for the Sony Playstation 2, which has incredible performance for 3D applications. Emotion is composed of a general purpose CPU, two vector units, and an RDRAM interface. One of the vector units (VU0) is for application code. The other vector unit (VU1) is for the front end of the graphics pipeline. In contrast to the vector unit in IRAM, these vector units are microcoded. (On the other hand, IRAM takes super computer approach of fetching every operation from memory.) The general purpose CPU is based on MIPS R4000 RISC core, and contains two integer units and one floating point unit, running at 300Mhz. This dual issue processor is tightly coupled with VU0, and together they run MIPS instructions expanded with Sony's SIMD operations, implementing high-level modeling computations. VU1 is dedicated to 3D geometry and lighting, running independently from the CPU. VU0 can help with VU1's job when it's not otherwise occupied. Though very similar, VU0 and VU1 are actually configured differently, with different mix of execution units and storages.

There was some discussion about whether this Emotion processor, which claims almost unprecedented performance numbers, would still be quite good for scientific computing as well. With its potential for very high instruction and data throughput, it seems that it should perform admirably for any data intensive applications, where enough data parallelism exists to make the vector units busy. However, some argued that the mix of execution units for Emotion chip is very specifically targed for the game graphics.

4.2 Graphics Synthesizer

The Graphics Synthesizer is a separate chip from Emotion chip, and is almost as big as the Emotion chip itself. It is responsible for rasterizing, shading, z-buffering, etc. It has integrated 4M video memory with very high bandwidth.

The paper claims that vector units can produce 66M polygons/sec, and graphics synthesizer can consume 75M polygons/sec. This beats even the top-notch graphics station only a few years ago. Sure enough, Sony has announced a plan for graphics workstation comprising of four of such Emotion chip connected together. It would be interesting to see how this competes against Silicon Graphics workstations.

4.3 Sony's Peculiar Approach

Sony takes an interesting step to ensure full backward compatibility. Instead of being bound by legacy code in developing their new high performance processor, they opted to ignore backward compatibility for Emotion, but include an entire MIPS processor that can run old Playstation games on the sam die the as I/O processor.

Another peculiar plan is that Sony has no intention of increasing the clock frequency of their chips with future process shrinks. This, though unorthodox in general purpose computing world, is really no surprise. Because the games are targeted for a certain processor performance, it doesn't help, and even may hurt, to increase the processor performance. If the games are written such that it relies on a particular processor speed, rather than a timer for example, the game will run too fast if the processor performance is increased. Instead, Sony will simply go for increased yield that come with process shrinks.

5 Final Thoughts

Vector or stream processor does the job very well if there's enough data parallelism to exploit. Because data parallel instructions have no dependencies almost by definition, it obviates the need for complex control hardware required to detect and prevent data dependent instructions from violating program behavior. Because of this, adding more execution units can yield almost linear speedup

However, it must be noted that it is suited only for certain programs that exhibit much data parallelism, such as media programs. For many integer general processing applications, this kind of parallelism may simply not exist. However, if the forecast of multimedia applications dominating CPU cycles is to be true, vector and stream processors would certainly be the logical step to take.