

CS1Bh Lecture Note 25

Machines: What next?

25.1 Introduction

The previous two lectures described, in some detail, the internal operations of a conventional von Neumann style processor when it executes instructions from its instruction set. In this lecture, we take a higher level perspective, surveying general trends in von Neumann processor design and briefly looking at the models and architectures that will be of relevance in the future. Given its broad-brush nature, this material will not be examinable. Instead, I hope that you will read and explore it both for its own sake, and as a taster of topics to be explored in more detail in subsequent years!

25.2 Rates of progress

Computer design takes place against the backdrop of a sustained pace of technological change. The number of transistors that can be accommodated per square inch of silicon chip area doubles every 18 months, a phenomenon that has been observable for the past 25 years, and is often referred to as *Moore's Law*¹. The actual speed of the transistors also increases year on year, by around 35%. These basic technological facts about transistors in turn impact favourably on the logic gates and memory devices used to implement processors and memories. Unfortunately, to quantify the problem discussed in the previous section, while processor speeds can double roughly every 18 months, their off-chip memory access speeds only double roughly every 10 years, which is a serious mismatch.

25.3 Alleviating memory latency

The main technique to overcome such *memory latency* is the use of *caches* as faster temporary storage devices placed between the processor and main memory. These store recently-accessed data items closer to the processor, based on the assumption that such items are more likely to be referenced again. In the background, the contents

¹Moore's original formulation in 1965 was even more dramatic, predicting doubling *every year*. Note that commercial pressure has a lot to do with making this 'law' a self-fulfilling prophecy, since manufacturers have no desire to fall behind competitors.

of caches must be kept consistent with the contents of main memory, which acts as the primary persistent storage. Note the basic storage hierarchy, in order of decreasing access speed (and cost) but increasing capacity: registers; caches; then main memory. Further beyond this lie non-silicon peripheral devices such as hard discs and compact discs.

25.4 The von Neumann bottleneck

The increasing mismatch between processor speeds and memory access speeds is not just a technological challenge. It also presents a substantial intellectual challenge. The fundamental basis of the von Neumann model is that, during execution, words are sent one-by-one, to and fro between the processor and the memory, whether representing instructions or data. This narrow orifice was christened the *von Neumann bottleneck* by Backus in 1978. Not only does it act as a physical restriction on the speed of computation, but it also restricts and serialises all our thinking on computation and how it should be programmed. It can be argued that this model, which was well-suited in times when computer hardware had to be simple and computer programming (as opposed to the resulting program) was complex, is now beginning to outlive its usefulness.

To take an example, with the rapidly increasing numbers of transistors available on chips, it is clear that there is great scope for many activities to be performed in parallel. Indeed, in order to design exceedingly complex chips with any degree of success, it is necessary to replicate regular parallel structures on the chip, rather than build single complex monolithic structures. We have seen in Section 25.5 that fine-grain *instruction level parallelism* can be extracted from a sequential von Neumann style program. Smart compilers are also able sometimes to find coarser-grain parallelism in programs, allowing several tasks to be performed in parallel by separate processors. However, such parallelisation is fighting the von Neumann bottleneck, if the programmer has to express a program in sequential terms, flattening out any natural parallelism in the problem to be solved, and then compilers and processors combine to reinject some parallelism. In order to introduce a more direct mapping between problems and silicon structures, more than 50 years of intellectual tradition has to be overcome, and this is by no means an easy agenda.

25.5 Internal parallelism in processors

The datapath of the MIPS R2000 is simple and economical in its use of hardware. This simplicity means that the micro-instructions are executed sequentially. However, architectures which permit instruction execution to be overlapped are possible. For instance, if we were to add a separate ALU for address calculation, this will permit the next instruction fetch to take place concurrently with the execution of the current instruction. This is a simple example of internal parallelism within the processor, using a simple two-stage pipeline for the fetch-execute cycle. The idea can be extended by breaking down the micro-instructions for each instruction into many, for example eight, small steps which can be also be overlapped in a pipelined manner. This

means that, at any one time, the processor is performing up to eight different micro-instructions from eight different instructions. Thus, after an initial start-up period, one instruction will be completed in every micro-instruction period. As long as we have the available space, increasing the number of steps decreases the time of any one step, and hence increases the instruction throughput.

Since 1990, the capability to integrate logic and memory on single silicon chips has far exceeded the needs of simple processors, and so the trend has very much been towards increasing on-chip parallelism. Modern *superscalar* processors also contain hardware that can dynamically reorder the instruction sequence so as to make best use of the available resources. The use of *speculative execution* is also widespread — this means that instructions are executed even though we are not guaranteed that they should be executed, for example, instructions that are inside conditional statements. If a mis-prediction takes place, the processor needs to ‘roll back’ to the correct state.

While parallelism is very useful in terms of increasing the performance of the processor, it must not affect the correct support of the von Neumann machine model. As programmers, we do not wish to reason about the fact that our code is being dynamically reorganised when it is executed. So, although modern processors have increasing amounts of internal parallelism, from a programmer’s point of view, they still behave exactly like a sequential von Neumann machine.

25.6 Parallelism between processors

As we have noted, the scope for parallelism within a single thread of instructions is naturally limited by the sequential mindset within which the thread was defined. If we are prepared to make the conceptual leap to designing algorithms and programs which make explicit use of many concurrent threads or processes, then we can hope to exploit a much larger degree of parallelism. Until fairly recently, computers with multiple CPUs were rather exotic and expensive but trends in microprocessor technology are now making such machines increasingly affordable. It is now quite reasonable to consider embedding many CPUs **on the same chip** (where once each CPU would have been spread across several chips), or to build one’s own supercomputer from a collection of “off-the-shelf” PC boards². On a different scale, “Distributed” and “Grid” computing techniques are designed to harness the raw computing power available within and between organisations into temporary supercomputers as availability permits and demand requires.

25.7 Systems on chip

Chip densities have now reached a level where very complex systems can be built on a single chip³. This has led to much research into *system-on-chip* (SoC), also known as

²For example, see <http://beowulf-underground.org/>

³In April 2002, Intel announced a chip with 500 million transistors, implementing a 64-bit processor with six megabytes of cache memory, and predict release of a nine megabyte version in 2004.

system-level integration (SLI). In the first instance, one way of harnessing this technology is just to scale down a circuit board containing discrete chips on to a single chip. For example, a mother board containing a processor, main memory and input/output interfaces can be replaced by one chip. Alternatively, a control system embedded within an appliance or device can be implemented on a single chip. Architecturally, there is little difference to the system, since this is more a matter of miniaturisation.

However, system-on-chip can enable more adventurous architectures. Staying in the realms of digital electronics (i.e., the world of the bit), there are two particularly promising directions for improvement. The first concerns interconnection between system components. Historically, such interconnection was fairly conservative in scale, since it was seen as wasted resource, not available for the actual components being connected. The use of buses, as seen in the processor micro-architecture of the previous lecture, is an example of this. Here, the interconnection fabric is shared over time, as opposed to providing direct connections between any two or more components that need to communicate. This is consistent with the enforced serialisation resulting from the von Neumann model, but mitigates against parallelism. In future systems on chip however, it will be possible to allocate significant area to allow richer interconnection, both in terms of how components are connected and in terms of connection bit width (i.e. much wider than 32-bit or 64-bit).

The second direction concerns the use of programmable logic (most commonly instantiated as the field-programmable gate array, FPGA), a technology mentioned in the first lecture of this series, as a means of providing 'soft hardware', that is logic circuitry that can be changed dynamically. This introduces an important programmable counterpart to the world of the processor, allowing systems to be expressed in terms of sequential programs or parallel circuitry, or both together. In 2002, typical programmable logic chips allow circuits containing millions of arbitrarily-connected logic gates. As an example of the style of contemporary systems on chip, the Xilinx Virtex-II Pro family of chips, launched in March 2002, is not only endowed with a huge number of programmable logic gates, but also has up to four PowerPC processors amidst the logic, together with on-chip memory and high speed input/output interfaces. This illustrates the manner in which processors are just becoming simple commodities in systems, not the centralised, important focus of systems.

Beyond digital electronics, there is an increasingly diverse range of other technologies that might be integrated into systems on chip. These include analogue electronics, micromechanical devices, optoelectronic devices, even so-called 'wetware' — chemical and biological components. In other words, very diverse types of machinery can be envisaged residing on single chips. This will enable many advances in providing novel types of appliances, devices and environments. For example, the "Smart Dust" project at the University of California at Berkeley envisages a future in which tiny devices, which integrate processing power, memory, solar or vibration powered batteries, various sensors and radio transmitters can be sprinkled (for example from the air, or in the fabric of buildings, or in every appliance) across an area of interest. Those which are still working and usefully oriented will contact each other then collaborate to monitor and report upon whatever might be of interest. The possibilities are boundless.

*Includes some original material by Gordon Brebner, Mike O'Boyle.
Murray Cole, 13th March 2003.*