Raw: Microprocessor for Extroverted Computing Support

Nuno Alexandre Magalhães Pereira

Departamento de Informática, Universidade do Minho
4710 - 057 Braga, Portugal
nunoampereira@yahoo.com

Abstract. A simple, highly parallel, VLSI architecture that exposes the hardware details to the compiler is proposed has the base platform to empower the emergence of pervasive, human centred computing experience. This communication discloses the motivation to build such architecture, analyses its details, presents its relationship to past and present architectures, and closes by addressing issues on compiling techniques, which enable an efficient use of silicon area and I/O pins.

1 Introduction

Industry developments are, every year, pushing the number of on-chip transistors to values of extraordinary magnitude. These developments introduce three major converging forces to computer architects: a need to keep internal chip wires short so that clock speed scales with feature size; the economic constraints of quickly verifying new designs; and changing application workloads that emphasize stream-based multimedia computations.

The Raw Microprocessor architecture elects these stream-based multimedia computations as the target application for future processors and attempts to answer a key technological problem for microprocessor architects: how to leverage growing quantities of chip resources even as wire delays become substantial [1].

One approach would be to rely on a simple, highly parallel VLSI architecture that fully exposes the hardware architecture's low-level details to the compiler or, more generally, the software. This allows the software to make direct use of every processor resources, and will ultimately enable programmers to achieve the maximum amount of performance and energy efficiency in the face of wire delay.

The architecture leans on a replicated design, much simpler than today's superscalar, providing efficient support for pipelined parallelism and existing architectural abstractions, such as interrupts, caches, context switches and virtualization.

This communication builds up from the motivations to create such new microprocessor; holding out current microprocessor architectures unsuitability for the foreseeable computational needs as chip resources grow exponentially and wire delay emerges has a major constraint for chips scalability.

The rest of the paper is organised as follows. Section 3 presents the details of the architecture. Section 4 compares the Raw Microprocessor with other existing microprocessor architectures, disclosing the architecture's inspiration in previous ones. Section 5 demonstrates the commitment to replace hardware sophistication with compiler smarts, showing some advantages and obstacles imposed by such approach, and Section 6 closes the communication.

2 Paving the way for extroverted computing

Computers started out as very introverted devices. Although they communicated with each other at high speeds, the bandwidth of their interactions with the real world was very low. With the introduction of video display and sound, the output bandwidth has increased in a high magnitude. Soon, with the appearance of audio and video processing, the input bandwidth will go to similar levels.

As a result of this, computers are going to become more and more aware of their environments. Given sufficient processing and I/O resources, computers will turn from recluse introverts to extroverts.

The emergence of the extroverted computing age is upon us. Microprocessors are just getting to the point where they can handle real-time data streams coming in from and out to the real world.

2.1 Microprocessor Evolution

The fabrication industry development has been truly daunting. This progress is expected to continue in the next decades. We are entering an era in which each microchip will have thousand millions of transistors.

In this new era, we could continue advancing our chip architectures and technologies as just more of the same: building microprocessors that are simply more complicated versions of the kind built today. We would incrementally add micro-architectural mechanisms to our superscalar and Very Long Instruction Word processors, one by one, carefully measuring the benefits [2].

The problem is that the current architecture for microprocessors does not scale. Most personal computers use an interface called the Instruction Set Architecture, or ISA, between the hardware and the software. Most instruction sets do not tell the software where the memory locations or function units reside on the chip, so current microprocessors must use hardware – for example, sets of wires or buses – to connect every memory location with every function unit.

Instead of following this path, a fuzzier, less defined goal can be pursued. The extra resources can be used to expand the scope of problems that microprocessors are skilled at solving. In effect, the attention is redirected from making processors better at solving problems they are already, frankly, quite good at, towards making them better at application domains which they currently are not so good at.

The weight of wire delay. The ongoing reduction in transistor sizes will enable hardware designers to insert more storage locations and function units onto each chip. Smaller transistors will also lead to a decrease in the duration of the chip's clock cycle. But because current architectures requires that the chip's wires connect every memory location with every function unit, the lengths of the wires will remain proportional to the diameter of the chip and will not decrease along with the clock cycle.
Delays in moving data along the wires will become increasingly significant and will eventually set a limit on the chip's performance. The resulting architecture will also be less energy-efficient, because longer wires require more energy to switch signals.

The architects of Compaq’s Alpha 21264 and, more recently, Intel Pentium 4 architects where already forced to make concessions to wire delay [3]. With increasing number of transistors on chip, the wire delay problem will only get worse.

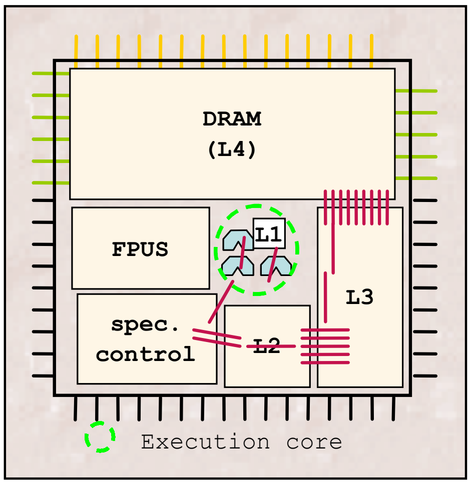

Fig. 1. How today's microarchitectures might adapt: chips that have a tiny portion in the middle, the execution core, clocked at a very high frequency and all sorts of mechanisms around it, focused on making that tiny portion in the middle run as efficiently and as fast as possible. Worsening wire delay as effective silicon area and pin resources increase.[1]

2.2 A new Processor for new Applications

Possibly, existing processor architectures can be modified to have improved performance on these new applications. But, instead of forcing chipmakers to spend time in carefully laying out the wires for each application, it is proposed that by the use of logic gates, is possible to reroute the flow of information on the chip's wires and use a compiler that enables automatic wires reconfiguration. This new model of computation is called Raw because it exposes the raw hardware on a chip to the software compiler. By using the free logic gates to direct and store the signals that run through the chip's wires, the compiler basically customizes the wiring for each application rates.

The Raw architecture is extremely simple. Its goal is to expose the maximum of the silicon and pin resources to these applications. The architecture provides a raw, scalable, parallel interface that allows the application to make direct use of all the silicon area and every I/O pin. The I/O mechanism allows data to be streamed directly in and out of the chip at extraordinary rates.

3 The Raw microprocessor architecture

Figure 2 shows the Raw architecture. A Raw microprocessor comprises a simple, replicated tile, each with its own instruction stream, and a programmable, tightly integrated interconnect between tiles.

The tile is kept simple and complex hardware resources are avoided, in order to maximize the clock rate and the number of tiles that can fit on a chip.

[1] Figure presented with kind permission of The Raw Project coordinators.

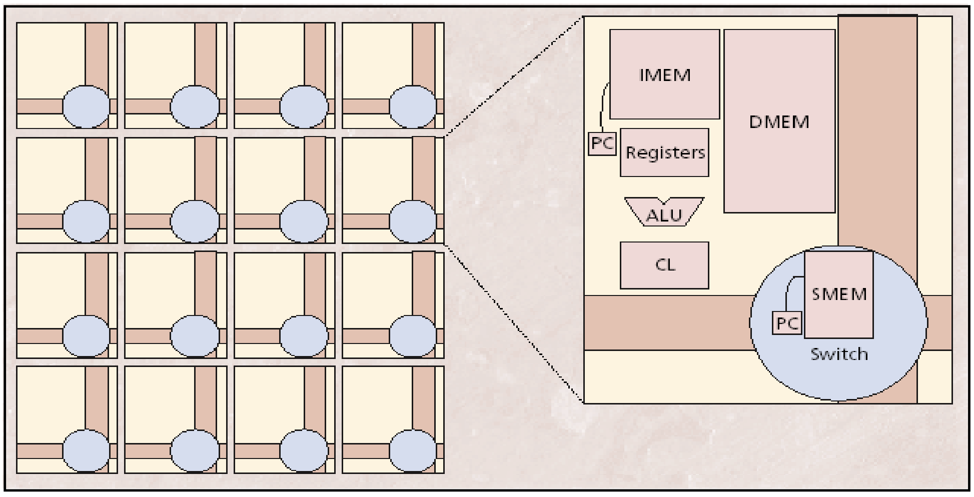

Fig. 2. A Raw microprocessor is a combination of tiles, each with a processor and a switch. The processor contains instruction memory (*IMEM*), data memory (*DMEM*), registers (*Registers*), an arithmetic logic unit (*ALU*), and configurable logic (*CL*). The switch contains its own instruction memory (*SMEM*).[2]

Each tile is sized so that the time for a signal to travel through a small amount of logic and across the tile is one clock cycle. Future Raw processors will have hundreds or perhaps thousands of tiles. Each tile only connects to its four neighbours. Every wire is registered at the input to its destination tile. This means that the length of the longest wire in the system is no greater than the length or width of a tile. This property ensures high clock speeds, and the continued scalability of the architecture. The network supports both static (routes specified at compile time) and dynamic (routes are specified at runtime) routing through the static and dynamic switches, respectively.

Raw's instruction set provides a parallel software interface to the gate, wire, and pin resources of a chip. The architecture provides direct access to all of these physical resources and lets programmers extract the maximum amount of performance and energy efficiency in the face of wire delay.

The Raw ISA exposes these on-chip networks to the software, enabling the programmer or compiler to directly program the wiring resources of the processor and to carefully orchestrate the transfer of data values between the computational portions of the tiles—much like the routing in a full-custom Application Specific Integrated Circuit (ASIC). Effectively, the wire delay manifests itself to the user as network hops.

On the edges of the network, the network buses are multiplexed in hardware down onto the pins of the chip using logical channels.

The Raw I/O port is a word-oriented abstraction that lets system designers proportion the quantities of I/O devices according to the application needs. Memory intensive domains can have a large amount of dedicated interfaces to DRAM. Other applications may not have external memory.

3.1 The Raw Tile

Each Raw tile contains a compute processor, a static switch processor and a dynamic router.

[2] Figure presented with kind permission of The Raw Project coordinators.

The compute processor is tightly integrated with the network, emphasizing its relevance in the architecture. Not only the network ports are mapped into registers, but they are also directly integrated with the processor pipeline. Figure **3** shows how this is accomplished.

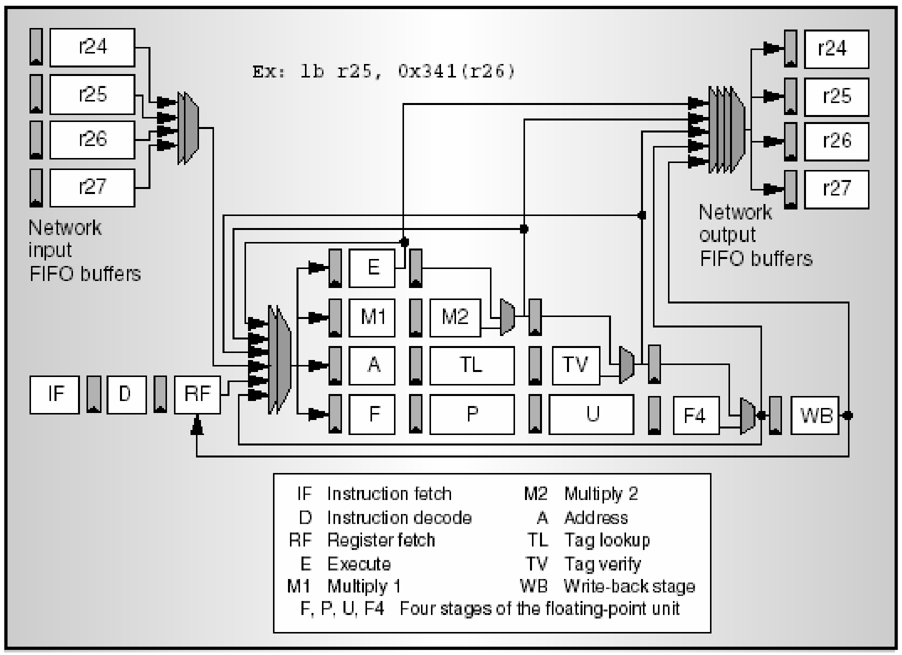

Fig. 3. Raw compute processor pipeline. Registers 24 through 27 are mapped to the four on-chip physical networks. For example, a read from register 24 will pull an element from an input FIFO buffer, while a write to register 24 will send the data word out onto that network.[3]

The switch processor multiplexes two logically distinct networks – one static and one dynamic – over the same set of wires. To distinct between accesses to the static and dynamic network ports, the compute processor uses different operation codes.

The static networks provide ordered, flow-controlled, and reliable transfer of single-word operands and data streams between the tiles' functional units. The operands need to be delivered in order so that the instructions issued by the tiles are operating on the correct data. Flow control of operands allows the program to remain correct in the face of unpredictable architectural events such as cache misses and interrupts.

For each word sent on the static network, a corresponding instruction in the instruction memory of each router must be programmed, typically at compile time, and cached in memory. In this manner, the static routers collectively reconfigure the entire network's communication paths on a cycle-by-cycle basis.

Because the static router knows what route will be performed long before the word arrives, route preparations can be pipelined. This allows for a very low latency network routing mechanism, essential for instruction level parallelism exploitation.

[3] Figure presented with kind permission of The Raw Project coordinators.

The dynamic router makes routing decisions based on each message's header. To send a message on one of these networks, the user injects a single header word that specifies the destination, a user field, and the length of the message. The user then sends the data words. While this is happening, the message is making its way through the network to the destination tile.

The primary goal of the dynamic network is to support memory accesses that cannot be statically analyzed. The dynamic network was also intended to support other dynamic activities, like interrupts, dynamic I/O accesses, speculation, synchronization, and context switches.

4 Comparing Raw to other Architectures

The Raw approach builds on several previous architectures. A Raw architecture seeks to execute pipelined applications (like signal processing) efficiently, as did earlier systolic-array architectures. Like computers based on Field-Programmable Gate Arrays (FPGAs), a Raw machine permits the construction of application-specific custom operations and communication schedules. Finally, like Very Long Instruction Word (VLIW) processors, a Raw processor simplifies and exposes the instruction-scheduling hardware to the compiler [4].

Along this section, these aspects similar to other architectures and how Raw distinguishes it self, will be clarified and explained in grater detail.

Work developed in the design of systolic arrays, emphasizes efficient processing and shares the approach of building point-to-point networks that support static scheduling. However, in these environments, the start up costs for a message is too high, forcing the compilers to focus on applications that have a uniform structure, in order to amortize this effect. Raw treats network communications in a register-like way, which allows to exploit the same types of ILP that superscalar processors do.

Raw, like an FPGA based machine, exploits fine-grained parallelism and fast communication, by accessing to its low level details and allowing the software to optimize the use of the hardware resources. However, unlike FPGA machines, compilation in Raw is fast, because it eliminates repeated low-level compilation of commonly used mechanisms. Additionally, Raw supports instruction sequencing, thus making it more flexible.

Many features in Raw are inspired in VLIW work. Like VLIW, Raw has a large register name space, a distributed register file, and multiple memory ports. Both rely heavily on compiler technology to discover and statically schedule ILP. Unlike traditional VLIWs, Raw uses multiple instruction streams. Individual instruction streams give Raw significantly more flexibility to perform independent but statically scheduled computations in different tiles, such as loops with different bounds.

The Raw processor is deceptively similar to a multiscalar processor, but the latter does not expose all its hardware resources to software. For example, a multiscalar might expose only 32 registers through a compact ISA, but relegate register renaming and dependence checking to hardware. A Raw machine exposes all registers to the compiler, and it implements similar mechanisms in software.

It is natural to compare Raw architecture with the logical evolution of multiprocessors: on-chip integration of a multiprocessor built from simple RISC processors. Like a Raw machine, such a multiprocessor uses a simple replicated tile and provides distributed memory. But unlike a Raw processor, the cost of message start-up and synchronization damages the multiprocessor's ability to exploit fine-grained ILP.

5 The “All Software Hardware” Question

The Raw project has been actively exploring the idea of replacing hardware sophistication with compiler smarts. However, it is not enough merely to reproduce the functionality of the hardware. For each alternative solution examined, it is necessary to compare its area efficiency, performance, and complexity to that of the equivalent hardware structure.

In some cases, removing the hardware structures allows better managing of the underlying resources, and results in a performance win. In other cases, as with a floating-point unit, the underlying hardware accelerates a basic function, which would take many cycles in software. If the target application domain makes heavy use of floating point, it may not be possible to attain similar performance per unit area regardless of the degree of compiler smarts. On the other hand, if the application domain does not use floating point frequently, then the software approach allows the application to apply that silicon area to some other purpose.

In the past, it has been difficult to compile for a static architecture like Raw. Today’s workloads, however, are beginning to emphasize stream-processing problems that can benefit significantly from the type of static pipelining Raw architecture supports. In addition, runtime systems can provide extra dynamic support when the compiler cannot easily identify parallelism in, for example, programs that use pointer-based data structures. In this case, the compiler identifies threads to speculatively execute in parallel. It will also construct software checks for resolving dependencies between threads, thereby making full compilation-time knowledge of such dependencies unnecessary. Runtime checks are slower than corresponding hardware mechanisms, but the compiler can optimize individual checks whenever information becomes available.

6 Conclusion

This communication presented the Raw microprocessor architecture, emphasizing it as an evolutionary response to current architectures, given the increasing number of on chip resources available. Also demonstrated, was how this approach builds on several previous architectures.

A Raw microprocessor distributes all of its resources - including instruction streams, register files, memory ports, and ALUs - over a pipelined two-dimensional network interconnect, and fully exposes them to the software system.

The software system can use Raw’s high degree of parallelism and wide, static network for algorithms that require high-bandwidth and fine-grained communication. Thus, a Raw architecture is particularly suited for traditional scientific applications and for processing streams of data.

In the near term, Raw architectures will be best suited for stream-based signal-processing computations. In 10 to 15 years, thousand million transistor chip densities, faster switching speeds, and growing compiler sophistication will allow the Raw Microprocessor performance-to-cost ratio to surpass that of traditional architectures for future, general-purpose workloads. Thus, Raw presents itself as a platform for the human-centred computational needs of the future.

The use of a prototype Raw microprocessor is currently enabling the architecture to mature, opening the way to explore the greatest amount of available parallelism, and to develop the research produced in the area of space-time compilation.

References

[1] D. Burger, J. R. Goodman: Billion-Transistor Architectures. IEEE Computer, Vol. 30, No. 9 (1997), 46-49.

[2] Michael Taylor: The Raw Processor Specification - The Raw Prototype Design Document V4.01, White Paper, ftp://ftp.cag.lcs.mit.edu/pub/raw/documents/RawSpec99.pdf, (2002).

[3] M.Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, J. Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, A. Agarwal: The Raw Microprocessor: A Computational Fabric For Software Circuits and General-Purpose Programs", IEEE Micro, Vol. 22, No. 2 (2002), 25-35.

[4] E. Waingold, M. Taylor, V. Sarkar, W. Lee, V. Lee, J. Kim, M. Frank, P. Finch, S. Devabhaktuni, R. Barua, J. Babb, S. Amarasinghe, A. Agarwal: Baring It All To Software: Raw Machines. IEEE Computer, Vol. 30, No. 9 (1997), 86–93.