A Selection of Recent Advances in Computer Systems

Oskar Mencer and Michael Flynn

Technical Report : CSL-TR-97-745

July 1999

This research was supported by DARPA Grant Nr. DABT63-96-C-0106

A Selection of Recent Advances in Computer Systems

by

Oskar Mencer and Michael Flynn

Technical Report : CSL-TR-97-745

July 1999

Computer Systems Laboratory

Stanford University

Gates Building 3A, Room 332

Stanford, California 94305

Email: oskar@umunhum.stanford.edu

Web: http://umunhum.stanford.edu/

Abstract

This paper presents a selection of recent advances in computer systems. The roadmap for CMOS technology for the next ten years shows a theoretical limit of 0.1 μm for the channel of a MOSFET transistor, reached by 2007. Mainstream processors are adapting to multimedia applications with subword parallel instructions like Intel's MMX or HP's MAX instruction set extensions. Coprocessors and embedded processors are moving towards VLIW in order to save hardware costs. The memory system of the future is going to be the next generation of Rambus/RDRAM. Finally, Custom Computing Machines based on Field Programmable Gate Arrays are one of the promising future technologies for computing – offering very high performance for highly parallelizable and pipelinable applications.

Key Words and Phrases: Computer Architecture, Processors, Memory Systems, Custom Computing Machines

Contents

List of Figures

List of Tables

1 Introduction

Computer Architecture is one of the fastest-changing fields in engineering history. Keeping up with all the recent advances results in a major challenges for researchers in computer systems.

The goal of research in computer architecture is to improve the ratio of price to performance of computer systems[Hennessy96]. Price and performance of computer systems are determined by VLSI technology, processor and memory systems architecture, compiler and operating system technology, and applications.

This paper has two major parts. First, we present a prediction of VLSI technology for the next ten years and the state-of-the-art in processor and memory systems. The specific topics where chosen in order to give an overview of the current state of the hardware part of microcomputer systems. Second, we present a new and promising field: Field Programmable Gate Arrays (FPGAs) for Custom Computing Machines (CCMs).

The fact that each one of the pieces of a computer system is rapidly changing creates an interesting phenomenon. The next generation processor is optimized to improve the performance of the computer system with current applications, compilers and operating systems. The next generation operating system optimizes performance for current processors. On top of that the new processor and operating system are optimized for current applications. Taking the discrete new inventions of each field and merging them together into one computer system does not necessarily create the expected cumulative performance gain.

A prominent example of the observation described above is multithreading and registerfiles. Multithreading is based on the observation that it takes a lot of time to switch between processes. As a result, the operating systems community creates lightweight processes called threads. Many threads are running in the same virtual memory space. Therefore switching between them is faster than switching between processes. At the same time processor designers increase the depth of the pipeline, the number of superscalar pipelines, and the size of the registerfile (i.e. increasing the amount of state for a thread).

The only way to avoid the above problem is to understand and keep track of the research done in all the relevant fields: hardware and software. Figure 1 shows the hierarchy of a computer system. Like the pieces of a puzzle each area has to make sure it fits into the overall global picture.

The second part of the paper deals with Custom Computing Machines (CCMs). CCMs consist mainly of Field Programmable Gate Arrays (FPGAs), introduced by Xilinx in 1985. The most widely used FPGA technologies for CCMs are Xilinx XC4000 and XC6200. We are currently using XC4000 FPGAs which consist of simple nibble wide nodes on a 2D mesh. This allows the programmer to exploit parallelism on the bit and nibble levels.

Research in FPGAs for general purpose computing started less than 10 years ago. Therefore the subject is still in its infancy. The major advantage over microprocessors is performance. It has been shown that for specific applications FPGAs can achieve speedups over conventional processors of 10 to 100 times. The major advantage over Application Specific Integrated Circuits (ASICs) is programmability. However, creating a new configuration on FPGAs means designing a new hardware architecture. Therefore, programming

FPGA based coprocessors is an order of magnitude more complicated than programming any conventional processor.

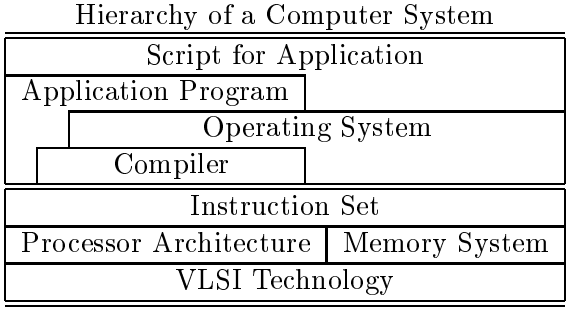

Figure 1: The hierarchy of a computer systems goes from VLSI technology all the way up to scripts written for specific applications. The instruction set serves as the interface between hardware and software. The figure shows the similarity of a computer system to a puzzle. Clearly if every piece is improved by itself, it is not sure that the resulting pieces are going to fit together in a meaningful way.

The space limitations of this paper do not permit a satisfactory treatment of all important advances in all the fields mentioned above. Instead we present a selection of research results which have some connection to Stanford University and are expected to influence the way we will build computers in the future.

Section 2 summarizes a recent PhD thesis [Farland95] about future scaling of CMOS VLSI technology. Section 3 deals with advances in memory systems. Section 4 summarizes new developments in microprocessor architectures and compilers. Finally, section 5 deals with the still more esoteric subject of Custom Computing Machines and their usefulness for computation.

2 The Future of CMOS Technology

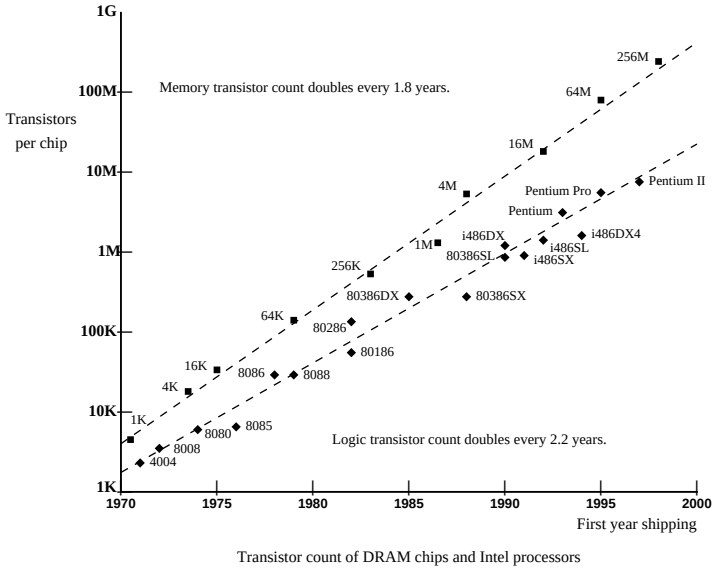

Figure 2: The above chart shows the exponential growth of transistors per chip over the last 25 years. DRAM Memory chips are more regular than processors. Therefore DRAMs keep a lead in transistor count over microprocessors.

The two major components of an integrated circuit (IC) are transistors and the wires that connect them. While CMOS transistors continue to scale down exponentially in the next decade, wires will not be able to follow.

2.1 Scaling Transistors

The number of transistors per chip has been growing exponentially now for over 25 years (see Figure2).

More transistors on a chip means increased amount of functionality. Smaller chips decrease the average number of defects per die [Flynn95] and therefore decrease the price for a chip.

Smaller transistors have less capacitance. Less capacitance means reduced power consumption at the same performance or higher performance with the same power. In addition, power consumption is further decreased by scaling down the supply voltage (V_{DD}). Reduced power results in smaller and lighter batteries for portable systems and less cost for cooling of stationary systems.

Figure 3 shows the prediction for a couple of parameters for CMOS transistors over the next decade. For more details on the physical bounds and their physical causes see [Farland97].

2.2 Scaling Global and Local Wires

As shown in Figure 3 local (i.e. short) wires are going to scale down slower than transistors. In Addition, the pitch and thickness of global wires will scale even more slowly than the local wires so that for a 0.1μ technology it would not be unreasonable to have the top most wiring layer be 10 times wider and thicker than the bottom most layer. This is to keep resistance of the long global wires to a minimum.

As a consequence chip designers will have to spend as much time and effort on wires as on transistors. Unfortunately in the current methodology chip designers deal with placement and routing of wires only in the last stage of the design process. High level simulations – before the length of the wires is known – will become less and less useful. For example, for a 0.1μ process, cache access delay will consist of 50% transistor delay and 50% wire delay [Farland97].

The layout of circuits like multipliers and dividers with critical interconnect delays are done by hand. If the software tools do not improve and adapt a new design flow which includes placement and routing very early in the design cycle, more and more layouts for integrated circuits will have to be done by hand.

CMOS Roadmap Summary					
1st DRAM Year	1995	1998	2001	2004	2007
L_{drawn} [μm]	0.32	0.23	0.15	0.12	0.08
V_{DD} [V]	3.3	2.5	1.8	1.5	1.2
FO4 [ps]	90	70	50	40	35
Clock [MHz]	375	475	700	800	1000
Local Wire Width [μm]	0.65	0.45	0.30	0.24	0.18
Resistance[Ω/μm]	0.15	0.19	0.29	0.82	1.34

Figure 3: 1st DRAM Year=First year where the technology is used for DRAM (processors usually follow some time later); V_{DD}=Supply Voltage, L_{drawn}= Drawn MOS channel length, FO4 is the delay of a gate with a fanout of four (i.e. a gate driving four other gates); Clock frequency is the official estimate of the Semiconductor Industry Association (SIA), Local Wire is a short wire connecting transistors locally on the lower metal layers of the chip. Note that transistor channel length scales down linearly, resulting in exponential growth of transistors per unit area. Wire resistance, on the other hand grows exponentially.

2.3 Consequences for Computer Systems

With each new VLSI process the complete organization of the computer system has to be rethought. The ability to put more transistors on a chip does not automatically mean that we know how to use this additional space efficiently. There is no unified methodology available to estimate how much space should be occupied by which functional unit; e.g. Should we increase the on-chip cache, the register file, or the floating point unit ? For the next generations of processors the question is: Should we put a shared cache

multiprocessor[Olukotun94] on a chip or a large out-of-order execution processor with more functional units and deeper pipelines ?

Another approach is to integrate the processor with main memory. Currently anticipated possibilities are Intelligent Memory [Patterson96] and processors with DRAM on a chip [Saulsbury96, Shimizu96]. We will discuss some of these systems in the following section.

3 Present and Future Memory Systems

3.1 Modern DRAM Technologies: Rambus RDRAM

Popular DRAM technologies for general purpose computers are Enhanced Data Out (EDO-RAM), Synchronous DRAM (SDRAM) [Przybyl96] and Rambus[1] (RDRAM). Each one of these interface technologies has it's strengths and weaknesses, thus the optimal choice of DRAM technology depends on the specific objectives for the system. Given that Intel and Rambus joined forces to create the main memory system for the next generations of Intel processors, it is very likely that Rambus technology will be very wide-spread in the future. Figure 4 shows the logical interface for an RDRAM memory system. DRAMs are connected with a 9-bit high-speed bus. Communication with the memory controller or processor is done via this high-speed communication link at frequencies above 500 MHz.

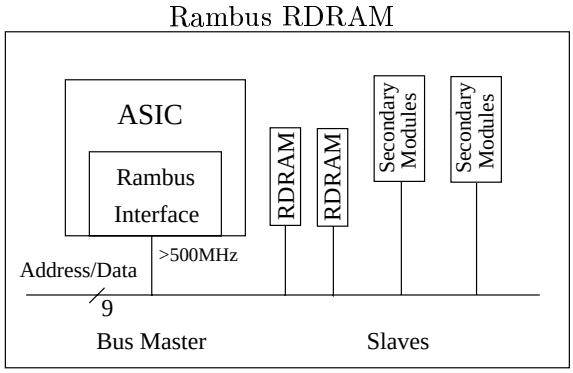

Figure 4: The figure shows the logical architecture of a Rambus memory system. Address and Data are multiplexed on a 9-bit high speed bus operating above 500MHz. The bus-master is either a specialized ASIC or a processor. The design of the Rambus interface is available from Rambus.

Compared to the other DRAM technologies, RDRAM has the following strengths [Przybylski96]:

1. Highest available bandwidth
2. Very low power consumption, ideal for portable systems.

[1] Rambus Inc. was co-founded by Mark Horowitz, on leave from Stanford.

3. Small physical volume (number of pins), resulting in low system integration costs.

4. Small granularity. Thus very small memory systems are feasible.

The single bad news from RDRAM is a very long access time. The access time for RDRAMs is basically in the range of generic Fast-Page-Mode DRAMs without any enhancements.

Rambus does not provide DRAMs, but instead licenses the technology to DRAM manufacturers – much like Dolby Labs licenses Dolby noise reduction for the audio world.

3.2 Combining Logic and Memory into One Chip

A promising project in the area of Logic-in-Memory is IRAM presented by David Patterson at HotChips '96 at Stanford [Patterson96]. The basic idea is to provide logic on the DRAM chip. Instead of increasing DRAM size, the processor is included on the DRAM

The major difficulties that arise from combining a processor with DRAM on a single chip are:

1. Achievable clock frequency for the processor is lower than for separate processor and DRAM chips.

2. DRAM market fragmentation. Conventional DRAMs are used by all computer systems. IRAMs can be used only by one computer system which uses the specific on-chip processor.

3. DRAM loses it's high testability, thus the price for manufacturing IRAM is higher than conventional DRAMs.

The advantages of IRAM are:

1. Lower latency to main memory.

2. Size and width of the on-chip DRAM can be adjusted to the processors needs.

3. Lower power consumption and less board space needed (for hand-held devices)

Another example for logic in memory is Misubishi's 3DRAM[Mitsubishi96].

> "3D-RAM is an innovative 10-Mbit cached dual-port CMOS memory device that dramatically improves the performance of a three-dimensional computer graphics system with on-chip support for Z-buffer hidden surface removal algorithm and for destination blending and logical raster operations."
>
> *3DRAM Databook*

Basically there is a specialized graphics processor added to synchronous DRAMs.

The conclusion for the development of the DRAM industry is that the DRAM market will become much more fragmented. DRAMs become specialized for a specific task. Different approaches might be necessary for graphics, main memory, low-power a.s.o. As a consequence the DRAM industry will have to adapt their business models to a wide variety of customer needs.

4 Trends in Microprocessor Design

This section deals with the heart of a computer system – the processor. First, we present MultiMedia eXtensions for general purpose instruction sets. The second part deals with recent advances in Very Long Instruction Word (VLIW) processors – a candidate technology for processors of the future.

MMX Relative Performance

	MPEG-1 Video	Speech Recognition	MPEG-1 Audio	Image Processing
Non-MMX	1.0	1.0	1.0	1.0
MMX	1.4	1.7	3.5	4

Figure 5: The table shows speedups for MMX as published by Intel. It is important to note that these speedups were achieved by experts adapting the applications to MMX by inserting assembly code instructions into the programs.

4.1 MultiMedia eXtensions

In 1994 the MAX instruction set extension for the HP-PA 8000 architecture is released. Soon, all the major microprocessor vendors follow with similar extensions to their instruction sets. Quite late, in 1997, even Intel released MultiMedia eXtensions (MMX) [Weiser96, Gwennap96] to the x86 instruction set [2].

The driving application for the development of these new instructions is the video compression standard MPEG. Computer users complain that the they can not run movies on their computers. While it is fine to wait a bit longer for a calculation to end, it is very annoying to watch a movie which runs at a few frames per second.

The solution is to realize that we have a 32 or 64-bit datapath, while most multimedia applications work on individual bytes – thus wasting most of the available resources. MMX instructions treat a 32-bit register as a 4-byte array. Splitting the datapath into byte-size blocks and performing computation in parallel is attractive due to two major reasons:

1. Minimal Hardware Overhead
2. High Speedup

The major hardware modification for the multimedia extensions is the modification of the ALU to break the carry chain e.g. carry-out of bit 8 is discarded and carry-in to bit 9 is zero. In addition there is a need for shuffle instructions. Most multimedia applications need to reorganize the location of the bytes inside the registers. The rest of the processor does not have to be modified.

Speedup can be as high as the number of bytes per register, i.e. four or eight. Figure 5 shows some speedup numbers obtained from Intel.

[2] While every major microprocessor vendor implemented MultiMedia eXtensions, we focus on MAX and MMX: HP-MAX because it was the first one and Intel MMX because it is going to be the most widely used.

4.2 VLIW: Very Long Instruction Word

The concept of VLIW was developed from early research in microcode architectures and compilers. The initial work on VLIW is reported in [Fisher83]. Since then the concept of VLIW was analyzed and explored by many research projects.

The major disadvantages of the original VLIW approach are :

1. Moving VLIW code to the next generation VLIW processor generation requires recompilation. The compiler has total control over on-chip resources.
2. The static scheduler does not take into account the instruction level parallelism (ILP) due to the specific input data that the computation is working on.
3. Performance is below conventional superscalar and out-of-order processors.

The major advantages of VLIW technology are:

1. Hardware is much simpler: shorter processor design time and smaller die area. Consequently the cost of the chip is lower.
2. Power consumption is lower than conventional processors because the data dependencies are resolved at compile-time. Thus less hardware is needed at run-time.

The Hotchips '96 conference at Stanford showed a massive acceptance of VLIW technology in the field of coprocessors and embedded systems[3](e.g.[Slavenburg96, Holmann96]).

Most recently researchers at IBM proposed a solution to the problem of binary compatibility of different generations of VLIW processors. The VLIW instruction is implemented as a tree-instruction. It contains a piece of the datagraph. Different VLIW processors split this large VLIW into the processor's own VLIW format which matches the available hardware resources [Moreno97].

[3]The most recent Digital Signal Processor (DSP) from Texas Instruments, the TMX320, consists of a VLIW core.

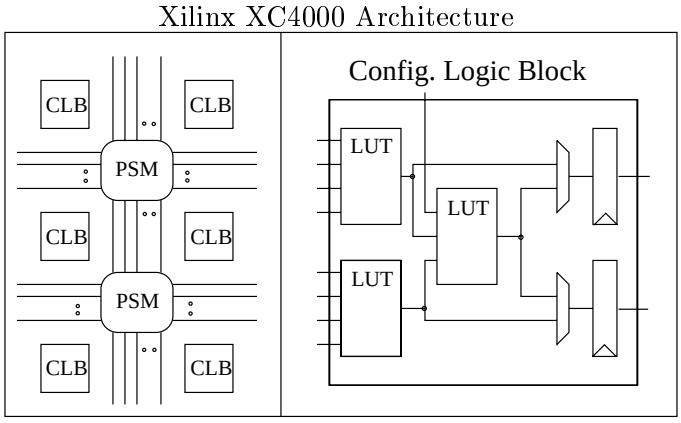

Figure 6: Configurable Logic Blocks (CLBs) are placed on a 2D-mesh, interconnected with Programmable Switch Matrices (PSMs). The datapath of a CLB is shown on the right. Eight bit-inputs from the left and one bit-input from above index into lookup-tables (LUTs) and finally go through two registers. One CLB can operate on nibbles (4 bits).

5 Custom Computing Machines

Soon after the introduction of Field Programmable Gate Arrays (FPGAs) by Xilinx in 1985 researchers at Digital (PRL) and the Supercomputing Research Center (SRC) pioneered in using FPGAs for computing. The two Custom Computing Machines (CCMs) from these two efforts are the DEC PeRLe-1 [Bertin92] and Splash [Arnold92]. FPGAs promise to speed up applications by exploiting fine grain parallelism. Although this seems to be a new approach, there are very strong connections to past efforts in exploiting fine grain parallelism.

5.1 FPGA Technology

The architecture of a generic reconfigurable FPGA is shown in Figure 6 (left). For Xilinx XC4000 FPGAs, Configurable Logic Blocks (CLB) are placed on a reconfigurable 2D mesh network. The major drawback of the reconfigurable network is that the network occupies over 80% of the total chip area. The architecture of an XC4000 CLB is shown in Figure 6 (right).

5.2 Computing with Field Programmable Gate Arrays

The configuration of a FPGA resembles a fine grain microinstruction or a very VLIW. Recent advances in VLIW compiler technology look promising for FPGA computing. Like a VLIW instruction, the configuration of the FPGA specifies the function that the hardware executes during one unit of time [Mangione98]. This unit of time has to be chosen to be much larger than the time needed to reconfigure the FPGA. With reconfiguration times around 100 ms for Xilinx 4000 series FPGAs and about a couple of hundred microseconds

for 6200 series FPGAs, it is difficult to find applications with a suitable temporal locality of computation.

The applications that have been shown to perform well on FPGAs are stream oriented, signal processing type applications [Bergmann94]. A relatively small algorithm is applied to large regular blocks of data. The data moves through the logic. Example applications include digital signal processing (DSP), data encryption, graphics and multimedia.

Programming is the basic difficulty that all approaches to exploit parallelism have in common. The major effort for making parallel computing systems accessible to a general community should be directed towards simplifying the design process of (fine grain) parallel programs. Another approach is to implement libraries as a flexible boundary between the user and the system. An additional level of abstraction is added on top of the logic level of the FPGAs. Higher level functions can be collected in a library which is called from high level languages such as Java or C.

Programming reconfigurable systems at the logic level has been shown to have the potential to improve execution time by order of magnitude. Speedups of 10-100 over conventional processors have been reported for specific applications. Figure 7 shows a summary of impressive speedups of FPGA based Custom Computing Machines (CCMs) compared to microprocessors.

CAD tools are not yet able to create efficient FPGA designs from a high-level description. As a first step, Xilinx provides XBLOX which are simple building blocks like adders, shifters, a.s.o.

We use the DEC PCI Pamette board [Shand95] to implement several sample library functions. The native design environment is PAMDC. Designs and their run-time environment are written in C++.

Our approach is to partition the compilation process into three stages:

1. Basic Building Blocks (e.g.adder, counter, ...)
2. Application Specific Library (ASL)
3. High Level Language using a Macro or Function Call to access the ASL.

In each step the goal is to decrease flexibility and simplify the interface.

The first step up is to use highly optimized basic building blocks for FPGA designs like adders, multipliers, counters, a.s.o. XBLOX, mentioned above, is a good example for such a library. XBLOX are highly optimized for the specific CLB and routing architecture of the FPGA – in our case Xilinx XC4000. Every wire is placed and routed manually in order to make optimal use of the available resources on the FPGA e.g. use of optimized carry-chains.

On top of these basic blocks we create a library of intermediate designs with an interface that is tailored to the application. The required number of different designs is much higher than the number of basic blocks. The intermediate designs correspond to the API or software library level. The programmer can then decide while writing the software to replace a call to the software library with a call to the FPGA library.

The interface to the FPGA library – a flexible hardware/software interface – is custom tailored to the needs of specific classes of applications. This approach is similar to graphics acceleration hardware like OpenGl in SGI workstations or PC graphics acceleration cards like for example Mitsubishi's 3DRAM [Mitsubishi96].

We are currently focusing on implementing encryption algorithms with the hierarchical approach described above. As a first result we found that the achievable performance gain is highly dependent on the available parallelism in the application e.g. the International Data Encryption Algorithm IDEA much better matches a generic digital signal than the XC4000 resources [Mencer98].

Performance of Custom Computing Machines (CCM)

Benchmark	CCM	Processor	Speedup
DNA Sequence comparison	Splash-2 (17 FPGAs)	CM-2 (64K Nodes)	20.0
RSA Crypt.	DEC-PAM	150MHz Alpha	17.8
Ray Casting	RIP-10	75MHz Pentium	33.8
8-bit FIR	1 Xilinx	50MHz DSP	17.9

Figure 7: The data for this figure was taken from the DARPA web-site. The figure shows a comparison of FPGA based computing machines and contemporary processors. It is important to keep in mind that these are the few applications which perform much better on a regular structure like FPGA, than on sequential processors.CM-2 stands for the supercomputer from Connection Machines. Splash and PAM are based on Xilinx FPGAs. The RIP-10 board is a CCM from Altera.

5.3 FPGAs versus VLIW and MMX

While research on compilers for VLIW is now done for over a decade, compiling software to FPGAs is a relatively new problem. MMX, MAX and similar extensions to instruction sets are currently accessed from high level language by inserting assembly code instructions by hand. Currently there is no compiler support planed for any of these extensions. While it is still relatively simple to program VLIW or MMX instructions, the achievable speedup is dictated by the available instruction level parallelism in the application.

VLIW machines move all the work which is being done dynamically on out-of-order execution processors to the compiler. The compiler is in charge of exploiting all the statically available instruction level parallelism from the application.

MMX instructions allow the programmer to use a 32-bit register as an array of 4 bytes. One MMX instruction operates on all 4 bytes in parallel. Thus MMX exploits byte level parallelism.

FPGAs have cells with a much finer granularity. Therefore FPGA designs can exploit parallelism on the nibble and bit level. In addition the depth of the pipeline and the number of parallel pipelines in the design can be adapted to the specific need of the application – thus resulting in higher performance with significantly higher programming effort.

6 Acknowledgments

Most of the information for section 2 was received from Grant McFarland. Section 3 is partly based on comparative analysis of DRAM technologies by Steven Przybylski. Section 4 was inspired by a talk of Ruby Lee who led the design of the first sub-word parallel instruction set extension MAX for the HP-PA 8000 architecture.

References

[WWW:] World Wide Web

[1] The Computer Architecture and Arithmetic Group at Stanford: *http://umunhum.stanford.edu/*

[2] Cpu Information Center: Announcements, Data and Performance: *http://infopad.eecs.berkeley.edu/CIC*

[3] IBM T.J. Watson Research Center: VLIW, Java VM Research: *http://www.research.ibm.com/vliw/*

[4] The PCI Pamette Board at DEC Systems Research Center: *http://www.research.digital.com/SRC/*

[5] Steven Przybylski: DRAM Technologies, Seminars and Tutorials, *http://www.verdande.com/*

[6] U.S. Government, Adaptive Computer Systems: *http://www.ito.darpa.mil/ResearchAreas/Adaptive_Computing_Systems/ACSintro.html*

[Papers:] Journals, Conferences and Theses

[Arnold92] J. M. Arnold, D. A. Buell, E. G. Davis, *Splash-2*, ACM Symposium on Parallel Algorithms and Architectures, ACM Press, New york, 1992.

[Bergmann94] N. W. Bergmann, J. C. Mudge, *Comparing the performance of FPGA-based custom computers with general-purpose computers for DSP applications*, Proceedings of IEEE Workshop on FPGAs for Custom Computing Machines, Napa, CA, April 1994.

[Bertin92] P. Bertin, D. Roncin, J. Vuillemin, *Programmable Active Memories: A Performance Assessment*, ACM FPGA, February 1992.

[Farland95] Grant McFarland, Michael J. Flynn, *Limits of Scaling MOSFETs* , Technical Report CSL-TR-95-662, Nov. 1995.

[Farland97] Grant McFarland, Michael J. Flynn, *CMOS Technology Scaling and Its Impact on Cache Delay* , Ph.D. Thesis, Stanford, June 1997

[Fisher83] J. A. Fisher, *Very long instruction word architectures and the ELI-52*, International Symposium of Computer Architecture, June, 1983.

[Gwennap96] Linley Gwennap, *Intel's MMX Speeds Multimedia*, Microprocessor Report, Vol. 10, Nr. 3, March 5, 1996.

[Holmann96] Y. Shimazu *A VLIW Processor for Multimedia Applications*, Hotchips VIII, Stanford, August 1996.

[Mangione98] W.H. Mangione-Smith, B. Hutchings, D. Andrews, A. DeHon, C. Ebeling, R. Hartenstein, O. Mencer, J. Morris, K. Palem, V. Prasanna, H. Spaanenburg, *Configurable Computing* , IEEE Computer Magazine, December 1997.

[Mencer98] Oskar Mencer, Martin Morf, Michael J. Flynn, *Hardware Software Tridesign of Encryption for Mobile Communication Units*, ICASSP 1998 (submitted for publication)

[Moreno97] J. H. Moreno, M. Moudgill, K. Ebcioglu, E. Altman, B. Hall, R. Miranda, S. K. Chen, A. Polyak, *Architecture, compiler, and simulation of a tree-based VLIW processor*, International Symposium of Computer Architecture, June, 1997.

[Olukotun94] K. Olukotun, J. Bergmann, K. Chang, B. Nayfeh, *Rationale, Design and Performance of the Hydra Multiprocessor*, Technical Report CSL-TR-94-645, Stanford, November 1994.

[Patterson96] D. Patterson, K. Yelick, *The Case for Intelligent DRAM: IRAM*, Slides: Hotchips VIII, Stanford, August 1996, Paper: IEEE Micro, 1997.

[Przybyl96] S. A. Przybylski, *SDRAMs Ready to Enter PC Mainstream*, Microprocessor Report, May, 1996.

[Rambus96] Rambus, Inc. *Rambus Memory: Multi-Gigabytes/Second And Minimum System Cost* , Rambus Inc. Advertisement, 1996.

[Saulsbury96] A. Saulsbury,F. Pong, A. Nowatzyk, *Missing the Memory Wall: The Case for Processor/Memory Integration*, International Symposium on Computer Architecture, Philadelphia, May 1996.

[Slavenburg96] G. A. Slavenburg, Philips Semiconductors, *The Trimedia TM-1 PCI VLIW Mediaprocessor*, Hotchips VIII, Stanford, August 1996.

[Shand95] M. Shand, *Flexible Image Acquisition Using Reconfigurable Hardware*, Proceedings of IEEE Workshop on FPGAs for Custom Computing Machines, Napa, CA, April 1995.

[Shimizu96] T. Shimizu et al. *A Multimedia 32 b RISC Microprocessor with 16 Mb DRAM*, Dig. Technical Papers, IEEE International Solid-State Circuits Conference, 1996.

[Weiser96] U. Weiser, Intel Israel, *Intel MMX Technology*, Hotchips VIII, Stanford, August 1996.

[Wirthlin95] Michael J. Wirthlin, Brad L. Hutchings, *A Dynamic Instruction Set Computer*, IEEE Workshop on FPGAs for Custom Computing Machines, Napa, CA, April 1995.

[Books:] Books

[Flynn95] M. J. Flynn, *Computer Architecture, Pipelined and Parallel Processor Design* , Jones and Bartlett, Boston, Massachusetts, 1995.

[Gupta98] A. Gupta, D. Culler, J. Singh, *Parallel Computer Architecture*, Morgan Kaufmann, San Francisco, California, 1998.

[Hennessy96] J. L. Hennessy, D. Patterson, *Computer Architecture: A Quantitative Approach* , Morgan Kaufmann, San Francisco, California, 1996.

[Milutinovic97] V. Milutinovic, *Surviving the Design of a 200 MHz RISC Microprocessor*, IEEE Computer Society Press, 1997.

[Mitsubishi96] Mitsubishi, Electronic Device Group, *3D-RAM: Frame Buffer Memory for High-Performance 3D Graphics*, Mitsubishi, Data Book, 1996.

[Przybylski96] S. A. Przybylski, *New DRAM Technologies*, MicroDesign Resources, California, 1996.