A SURVEY OF DIFFERENT APPROACHES FOR OVERCOMING THE PROCESSOR-MEMORY BOTTLENECK

Danijela Efnusheva, Ana Cholakoska and Aristotel Tentov

Department of Computer Science and Engineering, Faculty of Electrical Engineering and Information Technologies, Skopje, Macedonia

ABSTRACT

The growing rate of technology improvements has caused dramatic advances in processor performances, causing significant speed-up of processor working frequency and increased amount of instructions which can be processed in parallel. The given development of processor's technology has brought performance improvements in computer systems, but not for all the types of applications. The reason for this resides in the well known Von-Neumann bottleneck problem which occurs during the communication between the processor and the main memory into a standard processor-centric system. This problem has been reviewed by many scientists, which proposed different approaches for improving the memory bandwidth and latency. This paper provides a brief review of these techniques and also gives a deep analysis of various memory-centric systems that implement different approaches of merging or placing the memory near to the processing elements. Within this analysis we discuss the advantages, disadvantages and the application (purpose) of several well-known memory-centric systems.

KEYWORDS

Memory Latency Reduction and Tolerance, Memory-centric Computing, Processing in/near Memory, Processor-centric Computing, Smart Memories, Von Neumann Bottleneck.

1. INTRODUCTION

Standard computer systems implement a processor-centric approach of computing, which means that their memory and processing resources are strictly separated, [1], so the memory is used to store data and programs, while the processor is purposed to read, decode and execute the program code. In such organization, the processor has to communicate with the main memory frequently in order to move the required data into GPRs (general purpose registers) and vice versa, during the sequential execution of the program's instructions. Assuming that there is no final solution for overcoming the processor-memory bottleneck, [2], today`s modern computer systems usually utilize multi-level cache memory, [3], as a faster, but smaller memory which approaches data closer to the processor resources. For instance, up to 40% of the die area in Intel processors, [4], [5] is occupied by caches, used solely for hiding memory latency.

Despite the grand popularity of the cache memory, we must emphasize that each cache level presents a redundant copy of the main memory data that would not be necessary if the main memory had kept up with the processor speed. Although cache memory can reduce the average memory access time, it still demands constant movement and copying of redundant data, which contributes to an increase in energy consumption into the system, [6]. Besides that, the cache memory adds extra hardware resources into the system and requires implementation of complex mechanisms for maintaining memory consistency.

Other latency tolerance techniques, [7], include combining large caches with some form of out-of-order execution and speculation. These methods also increase the chip area and complexity. Other architecture alternatives, like wide superscalar, VLIW (very long instruction word) and EPIC (explicitly parallel instruction computing) suffer from low utilization of resources, implementation complexity, and immature compiler technology, [8], [9]. On the other hand, the integration of multiple processors on a single chip die brings even greater demands on the memory system, increasing the number of slow off-chip memory accesses, [10].

Contrary to the standard model of processor-centric computing, [11], some researchers have proposed alternative approaches of memory-centric computing, which suggest integrating or approaching the memory and the processing elements into a same chip die or closer to each other, [12]. This research resulted in creating a variety of memories that include processing capabilities, known as: computational RAM, intelligent RAM, processing in memory chips, intelligent memory systems, [13]-[23] etc. These smart chips usually integrate the processing elements into the DRAM memory, instead of extending the SRAM processor memory, basically because DRAM memory is characterized with higher density and lower price, [1], comparing to SRAM memory.

The merged memory/logic chips have on-chip DRAM memory which allows high internal bandwidth, low latency and high power efficiency, eliminating the need for expensive, high speed inter-chip interconnects, [12]. Considering that the logic and storage elements are close to each other, smart memory chips are applicable for performing computations which require high memory bandwidth and stride memory accesses, such as fast Fourier transform, multimedia processing, network processing etc., [24], [25].

The aim of this paper is to explore the technological advances in computer systems and to discuss the issues of improving their performances, especially focusing on the processor-memory bottleneck problem. Therefore this paper reviews different techniques and methods, which are used to shorten memory access time or to reduce (tolerate) memory latency. Additionally, the paper includes a comparative analysis of the advantages, disadvantages and applications of several memory-centric systems, which provide a stronger merge between the processing resources and the memory.

The paper is organized in five sections. Section two presents the reasons for occurrence of memory-processor bottleneck in modern processor-centric computer systems and discuss the problems that it can cause. Section three and four present an overview of the current state of research, related to overcoming the processor-memory performance gap. Actually, section three discusses the progress of processor-centric systems, exploring several techniques for decreasing the average memory access time. On the other hand, section four reviews the architecture and the organization of other alternative solutions, which provide a closer tie between the processing resources and the memory, by utilizing a memory-centric approach of computing. The last section gives a summary of the research presented in the paper and points out the possible directions for continuation of this research.

2. PROCESSOR-MEMORY PERFORMANCE GAP

The technological development over the last decades has caused dramatic improvements in computer systems, which resulted in a wide range of fast and cheap single- or multi-core processors, compilers, operating systems and programming languages, each with its own benefits and drawbacks, but with the ultimate goal to increase the overall computer system performances. This growth was supported by the Moore`s law, [26], [27], which allowed doubling of the number of transistors on chip, roughly every 18 months. Although the creation of multi-core systems has

brought performance improvements in computer systems for specific applications, it has also caused even greater demands to the memory system, [26]. Actually, it is well known that the increase in computing resources without a corresponding increase in the on-chip memory leads to an unbalanced system. Therefore, a problem that arises here is the bottleneck in the communication between the processor and the main memory, which is located outside of the processor. Assuming the difference between the memory and processor speeds, each memory read/write operations can cause many wasted processor cycles, waiting for data to be transferred between the GPRs and the memory or vice versa.

The main reason for the growing disparity of memory and processor speeds is the division of the semiconductor industry into separate microprocessor and memory fields. The former one is intended to develop fast logic that will accelerate the communication (by using faster transistors, according to the Moore`s law), while the latter one is purposed to increase the capacity for storing data, (by using smaller memory cells), [28]. Considering that the fabrication lines are tailored to the device requirements, separate packages are developed for each of the chips. Microprocessors use expensive packages that dissipate high power (5 to 50 watt) and provide hundreds of pins for external memory connections, while main memory chips employ inexpensive packages which dissipate low power (1 watt) and use smaller number of pins.

DRAM technology has a dominant role in implementing main memory in the computer systems, because it is characterized with low price and high density. The technological development of the DRAM industry has shown that the capacity of DRAM memory is doubling every two or three years, during the last two decades, [1]. Therefore, we can say that not long ago, off-chip main memory was able to supply the processor with data at an adequate rate. Today, with processor performance increasing at a rate of about 70 percent per year and memory latency improving by just 7 percent per year, it takes a dozens of cycles for data to travel between the processor and the main memory.

Several approaches, [7], have been proposed to help alleviate this bottleneck, including branch prediction algorithms, techniques for speculative and re-order instructions execution, wider and faster memory connections and multi-level cache memory. These techniques can serve as a temporal solution, but are still not able to completely solve the problem of increased memory latency. Even the development of a powerful superscalar, VLIW and EPIC processor, [29], (which are capable of executing several millions of instructions per second), didn't achieve the expected performance improvement as a result of the memory stalls, during the slow memory accesses.

The processor-memory bottleneck problem is also called a memory wall problem, which was first introduced by V. Wolf in 1995, [2]. In order to emphasize the importance of this problem, in figure 1 we present the difference in time that is required for executing an instruction into the processor and other instruction that requires access to the DRAM memory. The comparison given in fig. 1 refers to the period of last several decades. According to fig. 1, we can say that the number of tact cycles required for executing memory access instructions increases over the time. This is result of the steady acceleration of the processor working speed in comparison with the DRAM memory speed. The presented dissimilarity can be also noticed in figure 2, which illustrates the processor's speed improvement, by measuring the execution time of standard SPECint benchmark programs on different generation of processors; and the rate of DRAM memory access time reduction, measured through RAS (Row access strobe) and CAS (Column access strobe) time parameters of DRAM components with different capacity; for the period of three decades, [1]. The results given in figure 2 present that DRAM memory speed is annually improved by 1,07 times, (twice in every 10 years), while processors reach an annual improvement of 1,25 times until 1986, 1,52 times until 2003 and 1,20 times starting from 2004, [3]. As a result

of the given trends, the gap between the achieved annual improvements in processor and DRAM memory speeds attains an exponential growth.

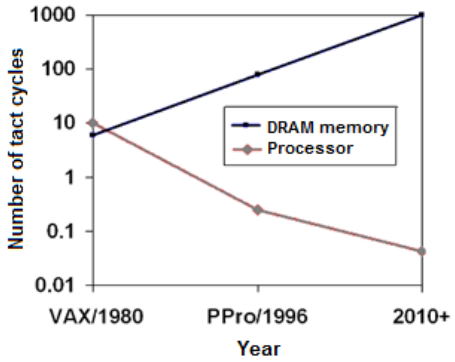

Figure 1. Number of tact cycles required for executing an instruction that accesses to DRAM memory and other instruction that is executed into the processor.

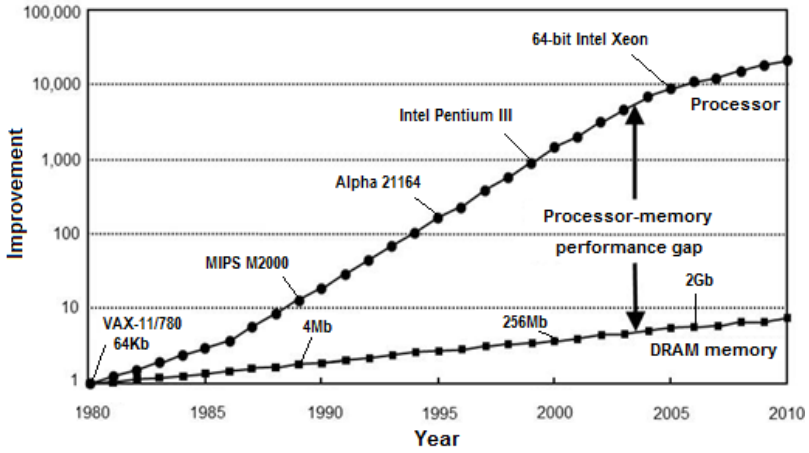

Figure 2. Yearly improvement of processor and DRAM memory speeds, for a time period of three decades.

3. Overview Of Techniques For Improving Memory Latency In Processor-Centric Systems

Today's computer systems are based on the Von Neumann architecture, [11], implemented as a stored-program machine with a strict separation of the memory resources intended to store data and programs and the central processing unit dedicated to perform control and execute arithmetical/logical operations. Considering that the processor has the main role in this kind of processor-centric system, computer designers have spent a great amount of time researching processor architectures, in order to propose some novelties that will improve the internal processor organization and its instruction set, and also will provide high level of parallelism, [1]. Besides the evolvement of different techniques for parallel computing on instruction, thread or data level, computer systems have been developing towards creating a multi-level memory hierarchy intended to decrease the average memory access time, [3]. Moreover, a transition from one-processor to multiprocessor/core systems which integrate several processors/cores on a single chip was increasingly coming to the fore. This type of multi-processing has brought even greater requirements to the memory system. As a result, the capacity and area of cache memory on chip

has been constantly increasing, as shown in figure 3. Actually, fig. 3 presents that Intel's 65nm processors use up to 40% of the processor chip's surface to implement 10MB of on-chip cache memory.

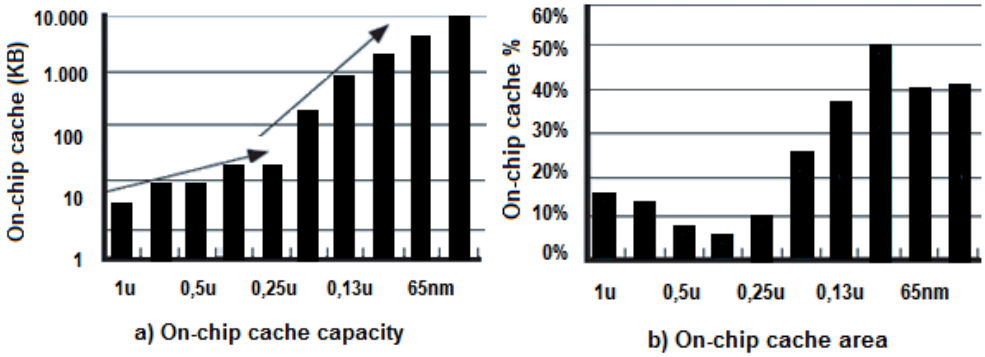

Figure 3. Evolution of on-chip cache memory for Intel microprocessors

The performances of the computing system mainly depend on the communication between the processor and the main memory. This behaviour is defined with the parameters: memory latency and memory bandwidth, [30]. Memory latency is the time between the initiation of a memory request and its completion, while the bandwidth is the rate at which data can be transferred to or from the memory system. According to that, the computer system may achieve maximal performances, only in the case when memory latency is approximately zero, and memory bandwidth is almost infinite. Such characteristics describe an ideal memory system, which cannot be achieved in practice. Considering that memory components use packages with small number of pins (limited bandwidth), computer architects have focused their research in the direction of investigating novel methods and techniques for improving memory latency, [6]. Therefore, two major groups of techniques for latency reduction and latency tolerance have been introduced.

The latency reduction group includes different techniques, which intend to decrease the time between the issuing of a memory request and getting the requested data as a response. The most widely used method from this group is the creation of multi-level memory hierarchy. The basic idea of this approach is to place several high-speed cache memory levels inside and close to the processor, in order to avoid the slow off-chip memory accesses. Other scientists have suggested innovations into DRAM memory architecture itself, [31]. This research has resulted with several DRAM solutions, including: asymmetric DRAM (provides non-uniform access to DRAM banks), Reduced Latency DRAM (RLDRAM), Fast Cycle DRAM (FCRAM divides each row in several sub-rows), SALP systems (Subarray-Level Parallelism System allows overlapping of different components of the bank access latencies of multiple requests that go to different subarrays within the same bank), Enhanced DRAM and Virtual Channel DRAM add a SRAM buffer to DRAM memory in order to cache the mostly accessed data, Tiered-Latency DRAM (TL-DRAM uses shorter bit lines with fewer cells), hybrid memory cube (places several memory modules dies on top of each other in a 3D cube shape) and embedded DRAM (eDRAM is integrated on the same chip die with the processor), [23], [32]-[35].

The latency tolerance group consists of different techniques, [28] which allow the processor to execute other operations while a memory request is being processed. One of the most popular techniques in this group is multithreading, [3], which is used to change the processor control flow, by starting an execution thread when some memory instruction has caused a miss in the cache memory. On the other hand, non-blocking cache memory, [36], allows execution of other requests in cache memory while a miss is being processed. In addition to that, other technique,

known as pre-fetching allows placing data or instructions in cache memory before they are actually needed, [1]. However, the research has shown that the usage of the previous techniques contributes to reducing memory latency, but causes increased memory traffic (higher instruction rate and need for operands). As a result of the limited bandwidth on the memory interface, additional latency is caused. This introduces more intensive work with the memory resources, causing a bottleneck in the system again.

4. OVERVIEW OF MEMORY-CENTRIC SYSTEMS

Computer architects have perceived the complex relationship between memory latency and bandwidth and have decided to create "smart memories", which will be able to achieve simultaneous latency reduction and memory bandwidth increase. According to that, several memory-centric approaches of integrating or placing the memory near to the processing elements have been introduced, including: register less processor which uses only cache memory (inside and outside of the processor) to communicate with the main memory, [37], systems which extend the on-chip cache memory with separate and small software-managed high-speed Scratchpad memory, [38], [39], chips that integrate processing and memory into the same chip die (Mitsubishi M32R/D, Terasys, DIVA, intelligent RAM, parallel processing RAM and DataScalar), [12] - [21], and intelligent memory systems, like the active pages model, [22], which adds reconfigurable logic blocks to each virtual page in the memory.

4.1. Scratchpad memory

The Scratchpad memory maps into a memory address space which is independent of the main memory, (see Figure 4), and is basically used for storing in-between results and most frequently used data, [39]. The main difference between Scratchpad and cache memory is that the Scratchpad memory guarantees access time of one tact cycle, while the cache access time depends on hit (1 cycle) or miss (10-20 tact cycles) occurring, [38]. Actually, the Scratchpad memory is software-managed, while the cache memory is hardware-managed. Therefore, implementing Scratchpad memory introduces more complications from a software point of view and requires developing of complex compiler methods for efficient data allocation.

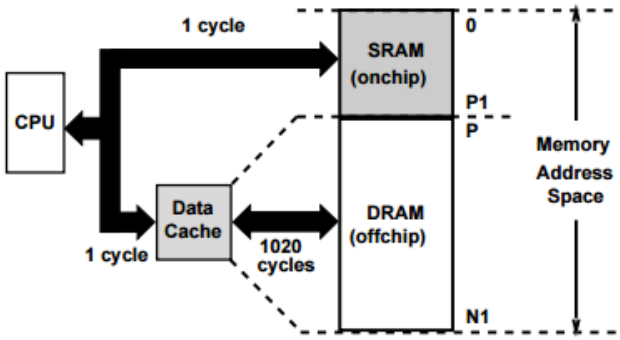

Figure 4. Memory address space division between on-chip Scratchpad SRAM memory and DRAM memory outside of the chip.

4.2. Performance enhanced register less processor

PERL is a performance enhanced register less processor which operates directly with the memory, breaking the standard memory hierarchy model. In fact, the PERL processor excludes the use of registers, and implements only on-chip cache memory inside the processor, as shown in figure 5, [37]. This is so called memory-to-memory architecture, and it allows the processor to

address the data directly, without the use of explicit register namespace. As a consequence, PERL instructions are 128 bits wide and include 32-bit memory addresses of both input operands and the result. Given that all the operands are directly addresses, the PERL instruction set does not need to include explicit load and store instructions. As a result, the programs instructions number is decreased, so the program's execution time is also reduced.

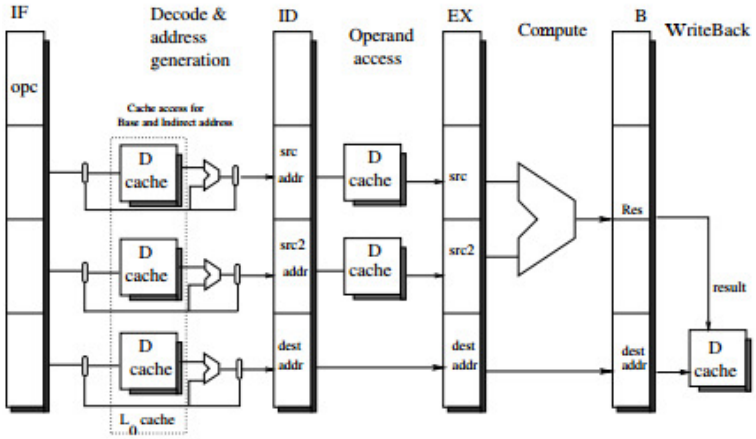

Figure 5. PERL Datapath.

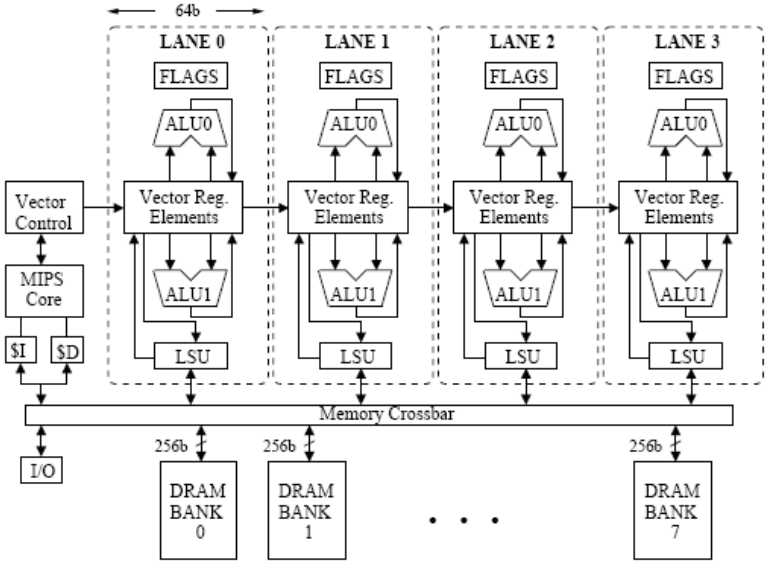

Figure 6. IRAM architecture.

4.3. Intelligent RAM

Intelligent RAM [17], [18], is a merged DRAM-logic processor, designed at the Berkeley University of California by a small group of students, led by Professor D. Patterson. An implementation of IRAM, known as VIRAM (vector IRAM) is a processor that couples high bandwidth on-chip DRAM with vector processing unit which provides high level of fine-grained data parallelism, [19]. VIRAM instruction set consists of standard MIPS instructions, extended with special vector instructions used for performing floating-point, integer and load/store operations. The VIRAM architecture is shown in figure 6, where it can be seen that it is consisted of MIPS general purpose processor, attached to a vector register file, which is connected to an

external I/O network and a 12MB DRAM memory organized in 8 banks. The vector register file includes 32 general purpose vector registers, each holding up to 32 64-bit elements and allowing parallel execution of vector operations over all the elements in the vector register. Actually, the VIRAM architecture is divided into four 64-bit lanes (pipelines) that can be further subdivided, resulting in eight 32-bit lanes. Every lane has two integer functional units, whereas one of them can be also used as a floating-point functional unit. According to that, the theoretical peak performance of VIRAM is 3.2 GFLOPS (Giga floating point operations per second) when operating with 32-bit floating-point operands and 6.4 GOPS (Giga operations per second) when operating with 32-bit integer operands.

4.4. Comparative Analysis

The technology of integrating processing logic in memory (PIM) defines merging of processor and memory resources into a single CMOS chip. Within this organization, the processor can be implemented as some sophisticated standard superscalar processor and may contain a vector unit, as is the case with the intelligent RAM (IRAM), [17]. The integrated memory into the chip can be realized as SRAM or embedded DRAM, which is basically accessed through the processor's cache memory, [23]. Considering that the processor and the memory are physically close, the integrated chip can achieve higher memory bandwidth, reduced memory latency and decreased power consumption, compared to today's conventional memory chips, and cache memories in multi-processing systems, [12]. Further and more detailed analysis of the most important features of several processing in memory chips is given in Table I. Besides that, this table also present the features of several approaches, [37] - [39], which modify the common multi-level memory hierarchy and provide nonstandard faster access to the main memory.

Table 1. A comparison between different memory-centric systems.

System	*Features*
PERL processor without registers,[37]	• **Advantages:** doesn't use explicit load and store instructions; lower instruction's count in a program, compared to DLX; shorter or similar program execution time, compared to DLX; • **Drawbacks:** long instructions (128b = 16B); twice bigger program size, compared to DLX; uses a C cross compiler (based on GCC), instead of dedicated compiler for the system; • **Application:** general purpose;
Scratchpad memory, [38], [39]	• **Advantages:** high speed;; smaller chip area and lower energy consumption in comparison with cache memory; uses separate part of the memory address space • **Drawbacks:** small capacity (several KB); software complexity (software - managed data allocation in memory); complex techniques for data allocation in scratchpad SRAM on-chip memory and DRAM main memory; • **Application:** embedded processing systems;
SUN PIM chip, [12]	• **Advantages:** high memory bandwidth; low price; scalability; decreased number of cache conflicts, by using a victim cache; shorter program execution time, compared to a superscalar processor which uses multi-level cache memory; • **Drawbacks:** limited amount of memory in the chip; the victim cache introduces additional complexity on hardware level; complex techniques for maintaining the coherence of shared cache memory in a distributed multiprocessor environment; • **Application:** general purpose;
Mitsubishi M32R/D chip, [14]	• **Advantages:** wide data bus in the chip (128 bits); low power consumption; uses a specific multiply and accumulate unit; flexibility (two modes: data and programs in DRAM on chip and programs in ROM outside of chip); uses an optimized C compiler;

System	*Features*
	• **Drawbacks:** limited amount of memory in the chip (16Mb); slow memory access to the ROM outside of the chip; intended only for applications whose data sets can be placed in the DRAM memory embedded inside the chip; • **Application:** embedded systems (PDA devices, multimedia equipment), digital signal processing;
DIVA system, [15]	• **Advantages:** flexibility; DIVA PIM chips improve memory bandwidth from 10 up to a 100 times and decrease memory latency, compared to DRAM; DIVA PIM chips are the first smart memory devices that support virtual addressing and can execute several parallel threads; provide direct communication between DIVA PIM chips without main processor interaction; DIVA PIM nodes implement 256-bit wide data path; program execution time in DIVA system with one PIM chip is 3,3 times shorter, compared to a standard processor which uses standard DRAM main memory; • **Drawbacks:** limited amount of memory in a DIVA PIM node (several MB); the ring topology, which is used for the PIM-PIM interconnects is not very suitable for expansion of the system; adds dedicated hardware and page tables in every PIM node in order to support virtual addressing; the mechanisms for maintaining coherence between PIM chips and cache memory adds extra complexity into the system; requires parallel model of programming and design of specific compiler; • **Application:** algorithms witch perform regular (image processing) and irregular data accesses (databases);
Terasys system, [16]	• **Advantages:** high memory bandwidth; flexible design; massive parallelism; implements specific mechanisms for communication between the processing elements; uses dbC high-level programming language (data-parallel bit C); • **Drawbacks:** requires development of specific software support for the system; requires defining a new (parallel) programming model; more expensive (higher price, compared to DRAM); inefficient utilization of computing resources; • **Application:** algorithms that include many data-parallel calculations, low-level image processing, matrix algorithms;
Intelligent RAM, [17], [18], [19]	• **Advantages:** 5-10 times shorter memory latency and 50 - 100 times higher memory bandwidth, than standard DRAM; 2-4 times lower energy consumption; wide data bus; the vector coprocessor achieves maximum performance of 1,6/3,2/6,4 GOPS (64b/32b/16b) while processing integers and 1.6 GFLOPS (32b) while processing floating-point numbers; • **Drawbacks:** limited amount of memory in IRAM chip; divergence of speed and cost depending on the production process; insufficient utilization of the vector coprocessor; testing time is longer compared to a standard DRAM; problems with chip overheating; difficulties with acceptance, because currently processors and DRAM memory are produced separately (different industries);
Intelligent RAM, [17], [18], [19]	• **Application:** embedded systems and portable devices, multimedia (image, video and audio processing), databases (sorting, searching, data mining), cryptography (RSA, DES/IDEA, SHA/MD5), data-level parallelizable algorithms;
Parallel processing RAM, [20]	• **Advantages:** high internal memory bandwidth; shorter memory latency compared to a standard DRAM; lower energy consumption; flexibility and scalability; high-level of parallelism; 2,22 times better performances compared to a corresponding superscalar system; the possibility of selecting an optimal PPRAM configuration, depending on the application requirements; • **Drawbacks:** limited amount of memory in a PPRAM node; difficulties in maintaining the logic speed and DRAM density in chip; increased design and test time; use of PPRAM link interface, as complex communication mechanism; • **Application:** any kind of usage, depending on the chip configuration;

System	*Features*
DataScalar system, [21]	• **Advantages:** the main memory is distributed between MOP elements, so each MOP accesses fast to its local memory; reduced number of accesses outside of the chip due to one-way communication between MOP elements; reduction of communication delay and traffic between MOP elements; faster execution time of programs that are not parallelizable • **Drawbacks:** limited amount of memory in MOP element; higher hardware complexity, compared to a single-processor system; problems with synchronization; requires complex mechanisms for maintaining cache consistency; the ring bus topology which interconnects the MOP elements adds complexity to the system; • **Application:** programs that can't be parallelized and whose data sets can't be placed in the internal on-chip memory granted to one processor;
Active pages model, [22]	• **Advantages:** intelligent memory system; flexibility; data-level parallelism; support for virtual addressing; use of reconfigurable logic in active pages; use of synchronization variables as a specific mechanism for coordination between active pages and the processor; 1000 times faster program execution, compared to an identical system with standard memory hierarchy; • **Drawbacks:** reconfigurable logic is characterized with lower speed than ASIC (application specific integrated circuit); requires supplementing the programming model with support for synchronization variables; requires developing of a specific compiler and operating system; the active pages model uses hand-coded libraries instead of a compiler; insufficient utilization of the active pages; programs with a big data set make the main processor bottleneck in the system; • **Application:** general purpose;

5. CONCLUSION & FUTURE WORK

Generally, the biggest problem in the reviewed memory-centric systems is the limited amount of integrated or coupled memory to the processor chip, as well as the divergence of the processing speed and chip cost, depending on the production process (memory process - lower speed and cost, processor process - higher speed and cost). Although the processing in/near memory brings latency and bandwidth improvement, still the system has to perform unnecessary copying and movement of data between the on-chip memory, caches and GPRs. Besides that, it is an even greater challenge to develop suitable compiler support, which will recognize the program parallelism and will provide effective utilization of the internal memory bandwidth, by generating specific instructions (ex. vector instructions in IRAM).

Having in mind that modern processors are lately dealing with both technical and physical limitations, while the memory capacity is constantly increasing, it seems that now is the right moment to reinvestigate the idea of placing the processor closer to the memory in order to overcome their speed difference. The latest research in that field is held from the HP (Hewlett Packard) international information technology company, which suggests novel computer architecture, called the Machine, [40]. This proposal presents a computer system which utilizes non-volatile memory (NVM) as a true DRAM replacement.

Considering that some of the previously discussed approaches (ex. PERL, Machine) break the standard memory hierarchy, we can say that the extension or revision of their work can be a promising direction for further research. First of all, we can assume that the relatively small number of fast GPRs in the processor is the major obstacle for achieving high data throughput. This is especially expected in the case of executing a program that works with large amount of data that need to be placed into the processor for a short time period. Examples for such

applications are: processing of data flows, calculating vast logical-arithmetical expressions and traversing complex data structures. In such cases, the high speed of access to GPRs doesn't bring many advantages, because all the required data cannot be placed into the register set at the proper time. After that, we can consider that cache memory brings unpredictability in the timing of the program, which is not very suitable for real-time systems. This means that the exclusion of these redundant resources (GPRs and caches) will simplify the memory access, will remove the unnecessary data copying and block transmissions into the cache memory, and will avoid the use of complex mechanisms for memory management. Therefore, our research will continue into the direction of developing a novel memory-centric architecture similar to PERL, which will provide direct access to the memory that is integrated into the processor chip (without the use of GPRs and cache memory).

ACKNOWLEDGEMENTS

This work was supported by the Faculty of Electrical Engineering and Information Technology in Skopje, Macedonia [national project: "Design and implementation of novel memory-centric processor architecture", 2017].

REFERENCES

[1] Patterson, D. A. & Hennessy, J. L. (2014) Computer organization and design: the hardware/software interface., Elsevier.
[2] Wulf, W. A. & McKee, S. A., (1995) "Hitting the memory wall: implications of the obvious", ACM SIGARCH Computer Architecture News, Vol. 23, Issue 1.
[3] Hennessy, J. L. & Patterson, D. A. (2012) Computer architecture: a quantitative approach, Elsevier.
[4] Borkar, S. & Chien, A. A., (2011) "The future of microprocessors", Communications of the ACM, 54(5), Vol. 54 No. 5, pp 67-77.
[5] Intel Corporation, (2013) "New Microarchitecture for 4th gen. Intel core processor platforms", Product Brief.
[6] Carvalho, C., (2002) "The gap between processor and memory speeds", Proceedings of 3rd. Internal Conference on Computer Architecture, Braga, Portugal.
[7] Machanick, P., (2002) "Approaches to addressing the memory wall", Technical Report, University of Queensland Brisbane, Australia.
[8] Smotherman, M., (2002) "Understanding EPIC architectures and implementations", Proceedings of ACM Southeast Conference.
[9] Silc, J., Robic, B., Ungerer, T., (1999) Processor architecture: from dataflow to superscalar and beyond, Springer.
[10] Jakimovska, D., Tentov, A., Jakimovski, G., Gjorgjievska, S., Malenko, M., (2012) "Modern processor architectures overview", Proceedings of XVIII International Scientific Conference on Information, Communication and Energy Systems and Technologies, Bulgaria, pp. 239-242.
[11] Eigenmann, R., Lilja, D. J., (1998) "Von Neumann computers", Wiley Encyclopaedia of Electrical and Electronics Engineering, Vol. 23, pp. 387-400.
[12] Saulsbury, A., Pong, F., Nowatzyk, A., (1996) "Missing the memory wall: the case for processor/memory integration", Proceedings of 23rd international symposium on Computer architecture, USA.
[13] Cojocaru, C., (1995) "Computational RAM: implementation and bit-parallel architecture", Master Thesis, Carletorn University, Ottawa.
[14] Tsubota, H., Kobayashi, T., (1996) "The M32R/D, a 32b RISC microprocessor with 16Mb embedded DRAM", Technical Report.
[15] Draper, J., Barrett, J. T., Sondeen, J., Mediratta, S., Kang, C. W., Kim, I., Daglikoca, G., (2005) "A prototype processing-in-memory (PIM) chip for the data-intensive architecture (DIVA) system", Journal of VLSI Signal Processing Systems, Vol. 40, Issue 1, pp. 73-84.
[16] Gokhale, M., Holmes, B., Jobst, K., (1995) "Processing in memory: the Terasys massively parallel PIM array", IEEE Computer Journal.

[17] Keeton, K., Arpaci-Dusseau, R., Patterson, D.A., (1997) "IRAM and SmartSIMM: overcoming the I/O bus bottleneck", Proceedings of the 24th Annual International Symposium on Computer Architecture.
[18] Kozyrakis, C. E., Perissakis, S., Patterson, D., Andreson, T., Asanovic, K., Cardwell, N., Fromm, R., Golbus, J., Gribstad, B., Keeton, K., Thomas, R., Treuhaft, N., Yelick, K., (1997) "Scalable processors in the billion-transistor era: IRAM", IEEE Computer Journal, 30(9), Vol. 30, Issue 9, pp 75-78.
[19] Gebis, J., Williams, S., Patterson, D., Kozyrakis, C., (2004) "VIRAM1: a media-oriented vector processor with embedded DRAM", 41st Design Automation Student Design Contest, CA, USA.
[20] Murakami, K., Shirakawa, S., Miyajima, H., (1997) "Parallel processing RAM chip with 256 Mb DRAM and quad processors", Proceedings of Solid-State Circuits Conference,
[21] Kaxiras, S., Burger, D., Goodman, J. R., (1999) "DataScalar: a memory-centric approach to computing", Journal of Systems Architecture.
[22] Oskin, M., Chong, F. T., Sherwood, T., (1998) "Active pages a computation model for intelligent memory", Proceedings of 25th Annual International Symposium on Computer architecture, pp. 192-203.
[23] Azarkhish, E., Rossi, D., Loi, I., Benini, L., (2016) "Design and evaluation of a processing-in-memory architecture for the smart memory cube", Proceedings of the 29th International Conference Architecture of Computing Systems, Germany.
[24] Blahut, R. E. (2010) Fast Algorithms for Signal Processing, Cambridge University Press.
[25] Madan, N., (2006) "Asynchronous micro engines for network processing", Master Thesis, School of Computing, University of Utah.
[26] IEEE, (2015) "Moore's law is dead - long live Moore's law", IEEE Spectrum Magazine.
[27] Hruska, J., (2014) "Forget Moore's law: hot and slow DRAM is a major roadblock to exascale and beyond", Extreme Tech Magazine.
[28] Bakshi, A., Gaudiot, J., Lin, W., Makhija, M., Prasanna, V. K., Ro, W., Shin, C., (2000) "Memory latency: to tolerate or to reduce?", Proceedings of 12th Symposium on Computer Architecture and High Performance Computing.
[29] Kozyrakis, C., Patterson, D., (2002) "Vector vs. superscalar and VLIW architectures for embedded multimedia benchmarks", Proceedings of 35th International Symposium on Microarchitecture, Instabul, Turkey.
[30] Patterson, D., (2004) "Latency lags bandwidth", Communications of the ACM, Vol. 47, Num. 10, pp. 71-75.
[31] Yu, W. S., Huang, R., Xu, S. Q., Wang, S., Kan, E., Suh, G. E., (2011) "SRAM-DRAM hybrid memory with applications to efficient register files in fine-grained multi-threading", Proceedings of 38th IEEE International Symposium on Computer Architecture.
[32] Son, Y. H., Seongil, O., Ro, Y., Lee, J. W., Ahn, J. H., (2013) "Reducing memory access latency with asymmetric DRAM bank organizations", ACM SIGARCH Computer Architecture News, Volume 41, Issue 3.
[33] Hamzaoglu, F., Arslan, U., Bisnik, N., Ghosh, S., Lal, M. B., Lindert, N., Meterelliyoz, M., Osborne, R. B., Park, J., Tomishima, S., Wang, Y., Zhang, K., "A 1GB 2GHz embedded DRAM in 22nm tri-gate CMOS technology", Proceedings of IEEE International Solid-State Circuits Conference.
[34] Barth, J., Plass, D., Nelson, E., Hwang, C., Fredeman, G., Sperling, M., Mathews, A., Kirihata, T., Reohr, W. R., Nair, K., Cao, N., (2011), "A 45 nm SOI embedded DRAM macro for the POWER™ processor 32 MByte on-chip L3 cache", IEEE Journal of Solid-state Circuits, Vol. 46, Num. 1.
[35] Jacob, B.L., (2002) "Synchronous DRAM architectures, organizations, and alternative technologies", Technical Paper.
[36] Li, S., Chen, K., Brockman, J. B., Joupp, N. P., (2011) "Performance impacts of non-blocking caches in out-of-order processors", Technical Paper.
[37] Suresh, P., (2004) "PERL - a register-less processor", PhD Thesis, Department of Computer Science & Engineering, Indian Institute of Technology, Kanpur.
[38] Panda, P. R., Dutt, N. D., Nicolu, A., (2000) "On-chip vs. off-chip memory: the data partitioning problem in embedded processor-based systems" ACM Transactions on Design Automation of Electronic Systems.
[39] Wang, P., (2013) "Designing Scratchpad memory architecture with emerging STT-RAM memory technologies", Proceedings of IEEE International Symposium on Circuits and Systems.
[40] Hewlett Packard Labs, (2016) "The Machine: the future of technology", Technical Paper.

AUTHORS

Danijela Efnusheva, M.Sc. is PhD student and teaching and research assistant at the Faculty of Electrical Engineering and Information Technologies, Univ. "Ss. Cyril and Methodius" – Skopje, R. Macedonia. Her main areas of research include: computer architectures; computer networks; routing protocols; digital hardware design; network processors; multi-gigabit networks; application specific processors FPGA programming; and high performance computing;

Ana Cholakoska is a MSc student and laboratory and teaching assistant at the Faculty of Electrical Engineering and Information Technologies, Univ. "Ss. Cyril and Methodius" – Skopje, R. Macedonia. Her main areas of research include: computer architectures; computer networks; computer and system security; and digital hardware design;

Aristotel Tentov, PhD is a full professor of computer science and engineering at the Faculty of Electrical Engineering and Information Technologies, Univ. "Ss. Cyril and Methodius" – Skopje, R. Macedonia. His main areas of research include: computer architectures; processor architectures; wired, wireless, and mobile networking; mission critical systems and networks; avionics; multiprocessor and multi-core systems; embedded systems; high performance computing; and process control;