Hybrid Manycore Dataflow Processor

Danko Miladinović, Miroslav Bojović, Vladisav Jelisavčić, and Nenad Korolija

Abstract— **This work addresses high performance computing architectures, presenting a hybrid processor that includes multiple computing architectures on a single chip die.**

Beside commonly used multicore processor, a personal computer might include manycore graphical processor. This work advocates for a combination of these two architectures along with a dataflow processor that usually appears in the form of a FPGA chip able to perform parallel tasks at the same time.

A quick overview of these computer architectures and appropriate programming paradigms is followed by the comparison based on flexibility and speed, price and development time, and speed and power consumption. Finally, the proposed hybrid processor is analyzed against computationally demanding algorithms that are often executed on high performance computing architectures.

Future work will include the comparison of the proposed computer paradigms and the comparison of the proposed hybrid architecture with existing ones.

Index Terms— **High performance computing; manycore processors; dataflow programming**.

I. Introduction

Many high performance computing algorithms are scalable, and as such, suitable for execution using:

- computer architectures the include graphical processing units capable of executing algorithms,
- dataflow computer architectures.

Both of these exist in the form of:

- personal computers,
- computer clusters,
- cloud computers.

Besides the differences in dataflow and conventional computing architectures, their computing paradigms also differ, as well as their suitability for executing high performance computing algorithms.

This work presents recently exploited computer architectures from the point of view of their usability for executing high performance computing algorithms. These architectures are compared based on programming flexibility that they offer, algorithm execution speed, scalability, software development effort, constraints of each of architecture type, price, and power consumption. The presentation of the work is based on the proposed method for presenting the results [1].

Danko Miladinović, Miroslav Bojović, and Nenad Korolija are with the School of Electrical Engineering, University of Belgrade, 73 Bulevar kralja Aleksandra, 11020 Belgrade, Serbia (e-mails: danko @ etf.bg.ac.rs, mbojovic @ etf.bg.ac.rs, and nenadko @ etf.bg.ac.rs).

Vladisav Jelisavčić is with the Mathematical Institute of the Serbian Academy of Sciences and Arts, Kneza Mihaila 36, Belgrade 11001 (e-mail: vladisavj @ gmail.com).

Following sections describe these computer architectures in a uniform manner. A brief overview is followed by the estimation of effectiveness. Each computing architecture is subjected to the following criteria:

- order of number of transistors per number of instructions that can run in parallel,
- speed of the hardware,
- suitability for high performance computing algorithms,
- independence from other computer architectures,
- price performance ratio,
- power consumption performance ratio,
- required space for computer architecture per performance ratio.

On the top of the work, authors propose a hybrid processor that includes multiple computing paradigms on a single chip.

II. Multicore architectures

Most of personal computer architectures are based on von Neumann paradigm. Programs written in programming languages are compiled and linked, and the resulting instructions are stored on the disk. Once a program starts, instructions are loaded into the RAM memory, from where they get copied into the cache memory, so that they could be read faster.

The processor is responsible for executing instructions. Long ago, processors included a single arithmetical logical unit (ALU) for performing arithmetical and logical operations. The speed of processors was increasing for decades by approximately doubling each second year. However, once the speed reached around 3GHz, the trend stopped. The constraint was and still is that the wave length became around 10 cm. Given the fact that the clock cycle must be stable during the whole instruction execution, and the signal has to travel with the speed of light multiple times in different directions, options for further acceleration were:

- to decrease the size of the chip,
- to decrease the size of transistors,
- to decrease the number of transistors per logical gates,
- to implement multiple ALUs that would work in parallel.

Decreasing the size of the chip die implies reducing the number of transistors, leading to deteriorating performances. Reducing the size of transistors would affect their functions considerably (i.e. more failures would appear). Logical gates are already optimized so that further reduction in the number of transistors would affect their capacities for producing output to multiple logical gates.

As a result, both research community and the industry opted for multiplying ALUs on a single chip die.

Multicore architectures can execute an order of 10 operations simultaneously. The transistor count is around 1 billion. Clock cycle is order of GHz. As such, their usability for high performance computer algorithms is limited in terms of utilizing algorithms in the real time applications. One of the most important characteristics of multicore architectures is their independence from other computer architectures. As a result, personal computers usually contain only one processor of this type. Typical power consumption is around 500 watts. When used in computer clusters, a single node could include multiple processors of this type.

III. Manycore architectures

Graphical processing units are often referred to as manycore computer architectures, as the number of processing units is larger than those of multicore processors.

So-called manycore architectures are logical successor of multicore processors. Certain fields, like computer graphics, require more instructions per second than multicore architectures produce, so that the picture can be refreshed many times per second (e.g. 60 times). With shading effects and full-hd or even 4k resolutions this implies updating millions of pixel colors based on performed operations with appropriate matrices. As a result, companies started producing chips for graphical cards that include thousands of cores that have lower number of available instructions supported by their architectures but are capable of executing more instructions per second. Soon after, utilization of the processing power of new graphical cards started. Many algorithms are implemented using the CUDA programming model.

Manycore architectures have an order of 10 billion transistors with an order of 1000 processor cores, and an order of 1000 instructions that can run in parallel. The clock cycle has an order of 1 GHz. This makes it suitable for high performance computing algorithms. Although each core is based on von Neumann paradigm, being capable of executing any instruction defined by the architecture at any given moment, it relies on the multicore processor as the one that assigns jobs to cores. The manycore architectures proved to be efficient for high performance computing algorithms, including data mining and coin mining. Power consumption is of the same order of magnitude as of multicore processors, while manycore processor offer superior performance. They usually come in the form of a PCIe card that attaches to the mainboard.

IV. Dataflow architectures

Dataflow architectures are based on a separate programming paradigm called dataflow paradigm. Data flows through a hardware in terms of electrical signals, resulting in transforming an input to the output [2, 3]. One of the main advantages over the previously mentioned computer architectures is that there is no need for an order of 1 billion transistors to execute only around 10 instructions in parallel. Dataflow hardware can execute even 1000 instructions simultaneously. As such, it is capable of accelerating many algorithms [4, 5]. The main disadvantage is that the multicore processor is needed for preparing the data to be processed using a dataflow hardware and for handling results.

The order of number of transistors is comparable to previously mentioned computer architectures, while the number of instructions that can run in parallel can be much higher. The speed of the hardware is around 0.1 GHz. Since they do not support executing any instruction defined by the architecture at any given moment, they are suitable for only a portion of high performance computing algorithms and more programming effort is needed for creating programs [6]. A mitigating circumstance is that there is a way to automatically translate certain algorithms from the control-flow into the dataflow paradigm [7]. Having significantly lower space occupation and power consumption per instruction, dataflow hardware has good price performance ratio.

V. Hybrid architectures

While each of the available computer architectures has its own advantages and disadvantages, it is a logical step but also a challenge to try to merge multiple programming paradigms into a single computer architecture. The task is even harder to achieve if they are to be put on the same chip die. However, the need for computing justifies the effort needed to merge existing computing paradigms.

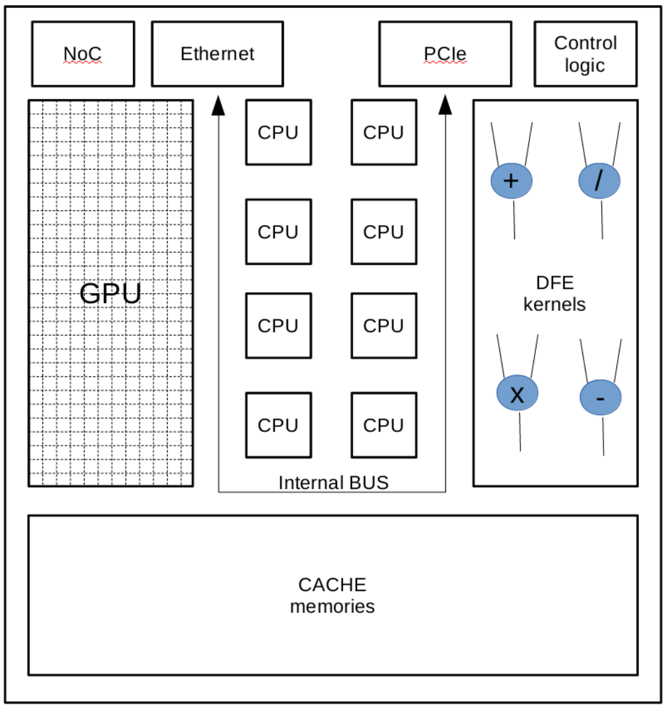

Fig. 1. Hybrid control-flow dataflow architecture.

Fig. 1 depicts a hybrid control-flow dataflow architecture on a single chip. Compared to the typical multicore processor architecture, the proposed hybrid architecture includes graphics processing unit (GPU) cores and dataflow kernels beside central processing unit (CPU) cores and cache memories. Additionally, network on chip (NoC) is suggested as a good way of handling the communication between GPU cores. Compared to the Maxeler dataflow hardware PCIe cards, proposed dataflow engine (DFE) kernels are connected directly to the slowest internal cache memory, because of the necessity for dataflow hardware to execute at the same speed. Constraint that has to be taken into account is that cache locking mechanism has to be implemented for cache memory connected to the dataflow, which would enable granted access to DFE kernels, but also to GPU if it is required by the application that is executed using both computing paradigms.

This architecture offers the best that any of the paradigms offer. On the other side, the complexity is equal to the sum of complexities of incorporated architectures. As such, the proposed hybrid architecture is suitable for executing multiple jobs simultaneously, where some of them are suitable for executing using the dataflow paradigm, while others achieve better performance when executed using manycore architecture. Finally, there are software applications that are not based on scalable algorithms, making them suitable solely for the multicore processor.

The emerging problem on heterogeneous computer architectures that include both dataflow and control-flow hardware is scheduling program execution. Authors have presented their novel algorithm for optimal scheduling of both dataflow and control-flow jobs [8]. The algorithm is general but is limited in number of jobs it can schedule due to the computational complexity. Based on this optimal algorithm, two heuristic algorithms for optimizing the throughput and minimizing total execution time are derived, producing near-optimal schedules for both dataflow and control-flow jobs at large job counts at the cost of negligible scheduling penalty. The heuristic algorithms performance gain decreases slightly as job count increases and only at the beginning, proving that the performance of existing cluster structures with appropriate dataflow accelerators can be considerably improved.

The drawback of combining multiple computing architectures on a single chip is that it increases the probability of failure. However, the probability is relatively high at the very beginning, and once a chip enters the so-called wear-out phase, and is relatively low in the meanwhile [9].

The order of number of transistors per number of instructions that can run in parallel depends on the type of job. If a job is suitable for dataflow architectures, the acceleration would be similar to those of dataflow architectures, while the number of transistors would be around three times higher, assuming that included architectures consume the same number of transistors. The same applies for jobs suitable for manycore architectures, and jobs that cannot be efficiently accelerated using neither manycore architecture, nor a dataflow architecture. However, this applies solely to scheduling a single job that can be executed using one of these three types of architectures. If we combine multiple jobs, those suitable for dataflow architectures would be executed there, based on their hardware requirements and on acceleration they achieve using the dataflow architecture. Jobs suitable for manycore architectures could run in parallel, while jobs that are based on algorithms that are not scalable could run in parallel on the multicore processor. Therefore, achieving fair comparison of the number of instructions that can be run on the proposed architecture is not a straightforward task. As a result, new benchmarks are needed to compare the proposed architecture to the existing ones. For now, one could consider the worst-case scenario, where the acceleration of a job is the same as for the best suited architecture for the job, and the number of transistors is equal to the sum of the number of transistors included on each part of the proposed hybrid processor [10].

The speed of the proposed hardware is also hard to define. As the processor includes different architectures that naturally run on different clock rates, the proposed processor would have a clock speed for the multicore processor that is a multiple of the clock speed of a dataflow architecture and manycore architecture. Therefore, the multicore would be able to efficiently communicate with other parts of the processor using input and output communication buffers.

The proposed computer architecture is not only suitable for high performance computing algorithms, but it is more efficient that aforementioned architectures, as it offers the most suitable hardware type for any particular job. The proposed processor includes multicore processor on the same chip die, making the architecture independent from other computer architectures. The price performance ratio for any given job is lower than those of the best suited of the aforementioned architectures. However, given a set of jobs, where each is suited for one particular architectures, the proposed architecture exploits advantages of all three types of architectures and could achieve the best speed-up in all categories of jobs. The price performance ratio is also lower for a particular job suited for a single computer architecture but is better than any of the aforementioned architectures if there are jobs that could approximately equally occupy all resources of the proposed processor. If completely utilized, the power consumption performance ratio is better than any of the three underlying architectures. For a single job, it is around three times lower, as it is estimated that this processor would consume as much electrical power as all three underlying architectures combined. The space required for the proposed computer architecture is one of its main advantages. Having in mind that it would be able to execute any type of jobs that any of the three computer architectures can, the space per performance ratio cannot be outperformed by any of the existing architectures.

This is not the unique case of proposing combination of manycore and dataflow architectures. Similar tries have been by researchers in the past [11, 12, 13].

VI. Comparison

This section summarizes the advantages and disadvantages of described computer architectures and compares them based on various criteria. For each of the given criteria, the proposed hybrid computer architecture is compared to all three underlying computer architectures.

The research [2] summarizes in their Table 2 the achieved speedups of algorithms implemented using the same type of the dataflow hardware that includes a memory on a chip comparing to the control-flow implementations of the same algorithms. The Lattice-Boltzmann algorithm is presented in detail, along with the dataflow code. Authors have compared execution time of Lattice-Boltzmann algorithms using the MAX2 card with 6GB of RAM and using Intel i5 650 processor with the clock speed of 3.2GHz. The computer used 4GB RAM memory at the speed of 1333MHz. The conclusion that can be drawn is that the speed-up of all observed dataflow algorithms ranges from 25% up to the multiplication factor of 150. Based on the comparison of these algorithm implementations for control-flow and for dataflow paradigms, we can summarize the advantages and disadvantages of both programming paradigms.

Table I presents a comparison of computing architectures in terms of flexibility to execute any instruction at any given moment and the speed of execution measured in number of instructions per second. As it can be seen, the multicore computing paradigm offers the highest flexibility by being able to execute any type of job, as it can execute any instruction defined by the architecture at any given moment. Although manycore architectures work on the same principle, they are considered to be utilized if many processing units may work in parallel. Therefore, their flexibility is limited to scalable algorithms suitable for manycore architectures. Dataflow architectures introduce new constraints by not being able to adapt to new needs before re-configuring the hardware. Hybrid architectures offer the advantages of both paradigms.

The speed of execution measured in number of instructions is lowest for the multicore architectures, as they are limited to processing up to an order of 10 instructions simultaneously. Any other of presented computer architectures can execute two or three orders of magnitude more instructions simultaneously.

TABLE I
FLEXIBILITY AND SPEED COMPARISON OF VARIOUS COMPUTING ARCHITECTURES

Type of architecture	Flexibility	Speed
Multi core	+++	+
Many core	++	++
Dataflow	+	++
Hybrid	+++	++

Table II presents a comparison of computing architectures in terms of price and the development time measured by effort needed for producing the software. The price raises as we lean towards more optimized computer architectures. The same applies to software development time. The only exception is that hybrid computer architecture, which is suitable for executing any of the given type of software, which means that the development time depends on the type of software being executed on the architecture.

As dataflow architectures are not as utilized as multicore and manycore computer architectures, the price tag of dataflow architectures is higher that it would be if each personal computer would include dataflow engines as well.

TABLE II
PRICE-DEVELOPMENT TIME COMPARISON OF COMPUTING ARCHITECTURES

Type of architecture	Price	Development time
Multi core	+	+
Many core	++	++
Dataflow	+++	+++
Hybrid	++++	+ - +++

Table III shows a speed to power consumption comparison of these computing architectures. The speed of the multicore computer architecture is slower than others, as it can run a smaller number of instructions in parallel.

When it comes to power consumption, it is similar for all types of architectures, which leads us to the following conclusion. The power consumed per a single instruction is the highest in the case of the multicore computer architecture, while it is the lowest for dataflow and hybrid computer architectures, assuming that they are not underutilized.

TABLE III
SPEED-POWER CONSUMPTION COMPARISON OF COMPUTING ARCHITECTURES

Type of architecture	Speed	Power consumption
Multi core	+	++
Many core	++	++
Dataflow	+++	++
Hybrid	+++	++

Based on these comparisons, we could conclude that the proposed architecture has the potential for achieving better results in terms of speed, flexibility, and power consumption comparing to the existing computer architectures, while the programming effort might be higher in the case of the algorithms implemented using the dataflow paradigm.

Comparing to the research available in the open literature, the proposed architecture is more high performance computing oriented than the Ultimate dataflow processor [12], while it doesn't support internet of things. Authors of SambaNova [14] also recognized the potentials of dataflow computing paradigm. Their Reconfigurable Dataflow Unit (RDU) enables accelerating algorithms with the flexibility to build custom dataflow pipelines as well as large memory capacity to run big models such as Natural Language Processing (NLP) and high-resolution computer vision efficiently. However, it is dedicated to algorithms that consist predominantly of the source code that can be most efficiently accelerated using the dataflow paradigm.

Research [15] exploits the opportunities from digital Processing-in-memory (PIM) bit-serial processing and in-memory customization, to tackle the above challenges by co-designing sparse algorithm, multiplication dataflow, and PIM architecture.

VII. CONCLUSION

Along with a multicore processor, a personal computer might include manycore graphical processor and a dataflow processor on the same chip die. This work advocates for a combination of these two architectures in order to create the type of the computer architecture that is able to execute jobs suitable for any of these three types of architectures in parallel.

Presented comparison between computing architectures suggests what kind of algorithms are suitable for execution using existing computing paradigms.

The proposed hybrid processor is analyzed against computationally demanding algorithms that are often executed on high performance computing architectures. As it includes multiple computing architectures on a single chip die, it could achieve the best acceleration for a job suitable for any of aforementioned computer architectures. At the same time, if the amount of jobs suitable for these three computer architectures matches the amount of resources of the proposed processor, the proposed processor with appropriate job scheduling can achieve the performance of combined architectures.

Future work includes the simulation comparison of the paradigms and the comparison of the proposed hybrid architecture with existing ones.

ACKNOWLEDGMENT

DM, NK, and MB are partially supported by the School of Electrical Engineering, University of Belgrade, Serbia. VJ is partially supported by the Mathematical Institute of the Serbian Academy of Sciences and Arts. NK is partially supported by the Institute of Physics Belgrade, contract no. 0801-1264/1. All authors are partially supported by the Ministry of Education, Science, and Technological Development of the Republic of Serbia.

REFERENCES

[1] V. Milutinovic, "The best method for presentation of research results," IEEE TCCA Newsletter, 1-6 (1996).

[2] N. Korolija, J. Popović, M. Cvetanović, and M. Bojović, “Dataflow-based parallelization of control-flow algorithms,“ Advances in computers, Elsevier, 104, 73-124 (2017).

[3] N. Trifunovic, V. Milutinovic, J. Salom, A. Kos, “Paradigm shift in big data super-computing: dataflow vs. controlflow,” J. Big Data, vol. 2, issue 4, 1–9 (2015).

[4] N. Trifunovic, V. Milutinovic, N. Korolija, G. Gaydadjiev, "An AppGallery for dataflow computing," Journal of Big Data, vol. 3, issue 1, 1-30 (2016).

[5] N. Korolija, T. Djukic, V. Milutinovic, and N. Filipovic, “Accelerating Lattice-Boltzman method using Maxeler dataflow approach,” The IPSI BgD Transactions on Internet Research, 34 (2013).

[6] J. Popovic, D. Bojic, and N. Korolija, “Analysis of task effort estimation accuracy based on use case point size,” IET Software, 9(6), 166-173 (2015).

[7] V. Milutinovic, J. Salom, D. Veljovic, N. Korolija, D. Markovic, and L. Petrovic, “Transforming applications from the control flow to the dataflow paradigm,” Dataflow supercomputing essentials, Springer, Cham, 107-129 (2017).

[8] N. Korolija, D. Bojić, A. R. Hurson, and V. Milutinovic, “A runtime job scheduling algorithm for cluster architectures with dataflow accelerators,” Advances in computers, Elsevier, 126 (2022).

[9] K. Huang, Y. Liu, N. Korolija, J. M. Carulli, and Y. Makris, “Recycled IC detection based on statistical methods,” IEEE transactions on computer-aided design of integrated circuits and systems, 34(6), 947-960 (2015).

[10] V. Milutinović, N. Trifunović, N. Korolija, J. Popović, and D. Bojić, “Accelerating program execution using hybrid control flow and dataflow architectures,” 25th Telecommunication Forum, IEEE, 1-4 (2017).

[11] V. Milutinović, E. S. Azer, K. Yoshimoto, G. Klimeck, M. Djordjevic, M. Kotlar, M. Bojovic, B. Miladinovic, N. Korolija, S. Stankovic, N. Filipović, Z. Babovic, M. Kosanic, A. Tsuda, M. Valero, M. de Santo, E. Neuhold, J. Skorucak, L. Dipietro, I. Ratkovic, "The ultimate dataflow for ultimate supercomputers-on-a-chip, for scientific computing, geo physics, complex mathematics, and information processing," 10th Mediterranean Conference on Embedded Computing, IEEE, 1-6 (2021, June).

[12] V. Milutinović, M. Kotlar, I. Ratković, N. Korolija, M. Djordjevic, K. Yoshimoto, and M. Valero, "The Ultimate Data Flow for Ultimate Super Computers-on-a-Chip, " Handbook of Research on Methodologies and Applications of Supercomputing, IGI Global, 312-318 (2021).

[13] V. Milutinović, B. Furht, Z. Obradović, and N. Korolija, “Advances in high performance computing and related issues,” Mathematical problems in engineering, (2016).

[14] R. Prabhakar, S. Jairath, and J. L. Shin, “SambaNova SN10 RDU: A 7nm Dataflow Architecture to Accelerate Software 2.0,” 2022 IEEE International Solid-State Circuits Conference (ISSCC), IEEE, vol. 65, 350-352 (2022).

[15] F. Tu, Y. Wang, L. Liang, Y. Ding, L. Liu, S. Wei,... and Y. Xie, “SDP: Co-Designing Algorithm, Dataflow, and Architecture for in-SRAM Sparse NN Acceleration,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, (2022).