Journal of COMPUTER AND FORENSIC SCIENCES

CFS 2022, Vol. 1, Issue 1, pp. 47–54
https://doi.org/10.5937/1-42710
Original research paper

# Towards Hybrid Supercomputing Architectures

**Nenad Korolija[1*], Kent Milfeld[2]**

[1] School of Electrical Engineering, University of Belgrade, 73 Bulevar Kralja Aleksandra, 11020 Belgrade, Serbia; nenadko@etf.bg.ac.rs

[2] The University of Texas, Texas Advanced Computing Center, J.J. Pickle Research Campus, 10100 Burnet Rd, Austin TX 78758; milfeld@tacc.utexas.edu

* Corresponding author: nenadko@etf.bg.ac.rs

Received: 2022-09-05 • Accepted: 2022-09-24 • Published: 2022-12-30

**Abstract:**In light of recent work on combining control-flow and dataflow architectures on the same chip die, a new architecture based on an asymmetric multicore processor is proposed.

The control-flow architectures are described as a most commonly used computer architecture today. Both multicore and manycore architectures are explained, as they are based on the same principles. A dataflow computing model assumes that data input flows through hardware as either a software or hardware dataflow implementation. In software dataflow, processors based on the control-flow paradigm process tasks based on their availability from the same queue (if there are any). In hardware dataflow architectures, the hardware is configured for a particular algorithm, and data input is streamed into the hardware, and the output is streamed back to the multicore processor for further processing. Hardware dataflow architectures are usually implemented with FPGAs.

Hybrid architectures employ asymmetric multicore and manycore computer architectures that are based on the control-flow and hardware dataflow architecture, all combined on the same chip die. Advantages include faster processing time, lower power consumption (and heating), and less space needed for the hardware.

**Keywords:** high performance computing; dataflow programming; manycore architectures; asymmetric cores.

## 1. INTRODUCTION

An exploration of using hardware for control-flow and hardware dataflow programming paradigms on the same chip die is presented.

In the following chapters, control-flow architecture principles, and computer architectures that are based on them are explained. Then, a description of dataflow architectures follows. Finally, a hybrid architecture that combines them is proposed. The communication between the control-flow hardware and the dataflow hardware on the same chip die might be achieved either using the cache memories that are shared between these hardware, or using the dedicated internal bus. As it will be explained, both approaches have their benefits and drawbacks.

## 2. MATERIALS AND METHODS

One of the main problems that computer vendors face in producing modern processors is the limited speed at which the hardware can operate. For decades, the speed has approximately doubled every two years. However, it appears that a 5GHz wall has been reached, and many commercial processors operate at 3GHz or less. Though the increase in speed is further possible, it would impose relatively high power consumption and cooling requirements. The alternative to improving the speed (or instruction rate) has been to increase the core count.

Hardware dataflow computing is based on the data traveling through the hardware. This solves many problems that computers based on control-flow face. However, the dataflow computing paradigm is suitable solely for certain types of high performance computing algorithms. Even algorithms that can be accelerated using the dataflow paradigm [1–5] include preparing data for parallel execution and collecting results, storing them in files, and auxiliary processing. As a result, dataflow hardware is usually connected to a multicore processor that controls it. It is the speed of the communication between the dataflow hardware and the control-flow hardware that limits the usability of this approach.

Another problem that arises from the combined control-flow and dataflow hardware is job scheduling. While certain jobs can be executed using both paradigms, other jobs can be executed solely on control-flow architectures.

Control-flow architectures are based on the von Neumann principle. Computer programs are written by programmers and compiled or interpreted. Both the compiler and the interpreter generate machine code understandable by the processor. Once the processor is ready to execute the instructions staged in memory, they are read and inserted into instruction registers. The instructions flow from the memory address register into the instruction register over the internal bus. Based on the instruction type, the processor might need to fetch an operand from memory as well. This involves the internal bus as well. Upon execution, the internal bus may be engaged once again, bringing the result into the register or memory. It should be noted that the internal bus might become the bottleneck. This limits the speed of the processing.

Control-flow hardware frequencies have been increasing for decades, approximately doubling every two years. This trend has ended, since further increases in speed impose overwhelmingly higher energy consumption and logical unit segmentation that cannot be accommodated for the projected speed-up that it might bring. Instead, as the density of packing transistors has grown lately, the number of processors has increased, which has evolved into multicore processor architectures. Increasing the number of cores does not proportionally increase performance, as many algorithms are not scalable enough. A potential solution is to exploit the dataflow paradigm [6–8].

Certain types of fast and relatively uniform processing are needed in computer graphics. In order to cope with this problem, graphics cards host thousands of small processing elements based on control-flow on the same chip. These architectures are often appropriately referred to as manycore architectures (relative to the core count of CPUs).

There are software and hardware dataflow architectures. Software dataflow architectures are based on the control-flow paradigm. Multiple processing elements are executing tasks from their common input queue, storing the results in their common output queue. The reason this is called dataflow computing is that data flows through the hardware based on the availability of processing elements and queues. Examples and discussions of these well-known architectures are available elsewhere [9–10].

Hardware dataflow works on a completely different principle. The hardware is configured for a certain algorithm, and the data flows through the hardware.

The hardware is usually implemented using FPGAs, enabling it to be reconfigured for the next job that the hardware is to execute. Dataflow processors can enhance high performance computing [11–12], as evidenced by the many high performance applications that have been accelerated using the dataflow hardware [13–18].

Transforming applications from the control-flow paradigm into the dataflow paradigm can be automated [19–20]. However, even with automating transformations, exploiting hardware dataflow still requires significant programming effort [21]. Many high performance algorithms are already implemented for dataflow hardware and available online [22–23], which can help programmers who are new to the dataflow paradigm boost the development of dataflow algorithms.

Dataflow hardware jobs usually last relatively long compared to those of control-flow type of processors. The reason for this lies in the facts that the dataflow hardware is suitable only for high-performance computing algorithms and that the hardware has to be configured prior to the execution of the algorithm. In order to be able to execute the algorithm faster than the processor based on the control-flow paradigm, it has to perform both the configuration and the calculation faster than a processor based on the control-flow paradigm would perform solely the calculation. The trend is that computers should have both control-flow and dataflow components [24].

The problem becomes more complicated when there are multiple tasks and when some can be executed solely on the control-flow hardware. These require different scheduling techniques than those used for multicore processors and cluster computing, which are discussed in existing research [25].

## 3. RESULTS

With growing capabilities in terms of the number of transistors per chip and the limited scalability of algorithms that multicore architectures usually execute, the question of whether to combine the control-flow and the dataflow hardware on the same chip die is reasonable. Further, the processor might also include the manycore architecture, besides multicore, and the dataflow on the same chip die. The research suggests the implementation of hybrid architectures might bring certain benefits, including faster processing, lower power-consumption, and less space needed for the hardware [26–27]. This is especially important when it comes to cluster and cloud computing. Previous research also proposes including the Internet of Things on the same chip die.

In addition to what has already been proposed, a hybrid processor may include asymmetric processors. Although all cores have the same instruction set, their microarchitectural properties may differ, so that there may be one core that is the fastest and the rest slower but of equal speed. Researchers have shown that appropriate scheduling can lead to a decrease in the probability of conflicts and an improvement in performance for transactions migration to the proposed asymmetric multiprocessor, based on the history of execution [28]. Their proposed solution is fully implemented in hardware. The claim is that it is possible to improve the performance by up to 14%. This serves as a proof that the hybrid processor proposed in this article can improve the performance of a similar hybrid processor with symmetrical multicore architecture [29], albeit for certain types of algorithms.

Figure 1 depicts the architecture of the proposed hybrid architecture.

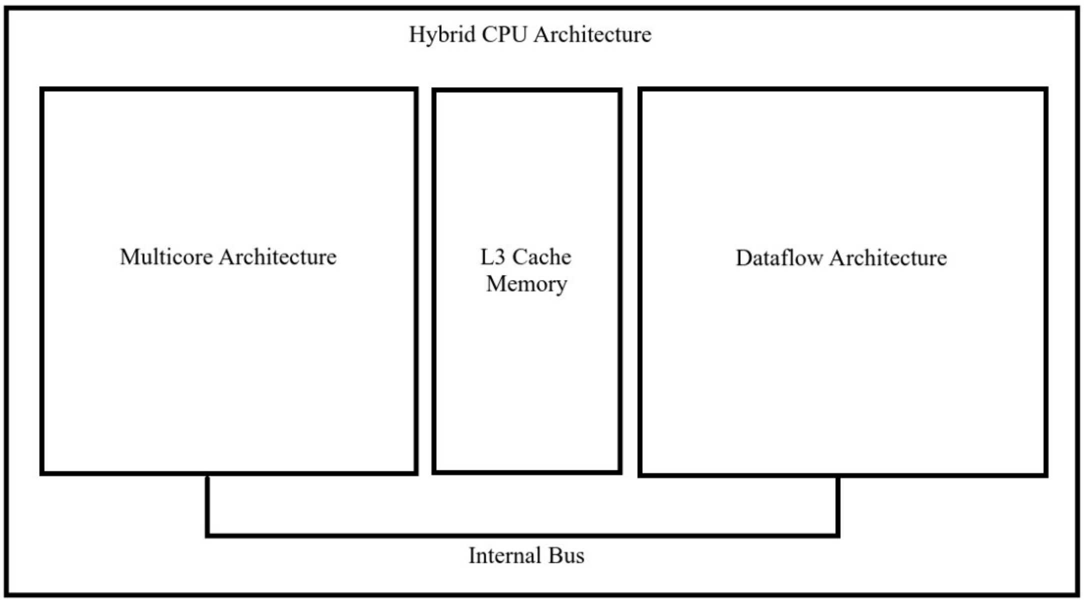

**Figure 1.** *Hybrid CPU architecture.*

## 4. DISCUSSION

The communication between the control-flow hardware (multicore) and dataflow hardware in the hybrid architecture might be achieved either using existing shared cache memories or using the internal bus. The first approach requires special care about the timing constraints, as two hardware in general do not operate at the same frequency. However, it doesn't require much extra hardware. The second approach requires additional control logic to cope with the specifics of the two hardware. This doesn't require introducing new logic but rather applying the existing mechanisms for communicating between the ordinary control-flow processor and considerably slower main memory. The communication in this case doesn't interfere with exploiting cache memories, but the drawback is that this approach requires more space on the chip die required for implementing the extra internal bus.

The proposed hybrid architecture can serve as a processor in a computer cluster or in a cloud [30], multiplying benefits with the number of processors working in parallel. Important domains of the combined control-flow and dataflow hybrid processor are the expected lifetime and counterfeit detection algorithms. The duration is expected to be the shortest of the three incorporated architectures, and possible counterfeit hybrid processor chips could be identified using existing statistical methods [31].

## 5. CONCLUSION

The improvement in computer architectures moved from incrementing the speed of processing toward increasing core counts. Additionally, computer graphics often requires much more processing per time, compared to the capacity of multicore processors, resulting in the introduction of so-called manycore architectures that include thousands of small processing elements based on the control-flow paradigm. Computer vendors that provide cloud infrastructure now offer dataflow processing that solves problems for processors based on the control-flow paradigm.

As the improvement in processor frequencies has dropped and the number of transistors per chip has increased, it is plausible for a single chip die to include multicore, manycore, and dataflow hardware. This requires special scheduling techniques to cope with the new needs. This work suggests that combining these computing architectures on the same chip is possible and has benefits, including: faster processing, lower power consumption, and less space needed for the hardware. Additional performance gains can be achieved with an asymetric multicore architecture. All these results can be especially important for cloud infrastructure and computer clusters.

**Funding:** N.K. is partially supported by the School of Electrical Engineering, University of Belgrade, Serbia, by the Institute of Physics Belgrade, contract no. 0801-1264/1, and by the Ministry of Education, Science, and Technological Development of the Republic of Serbia.

## INSTITUTIONAL REVIEW BOARD STATEMENT:

Not applicable.

## INFORMED CONSENT STATEMENT:

Not applicable.

## CONFLICTS OF INTEREST:

The authors declare no conflict of interest.

## REFERENCES

[1] A. Kos, V. Ranković, and S. Tomažič, "Sorting networks on Maxeler dataflow supercomputing systems," *Advances in Computers*, Vol. 96, pp. 139–186, 2015.

[2] V. Ranković, A. Kos, and V. Milutinović, "Bitonic merge sort implementation on the maxeler dataflow supercomputing system," *The IPSI BgD Transactions on Internet Research*, Vol. 9, No. 2, pp. 5-10, 2013.

[3] N. Korolija, V. Milutinovic, and S. Milosevic, "Accelerating conjugate gradient solver: temporal versus spatial data," *The IPSI BgD Transactions on Advanced Research*, Vol. 3, No. 1, pp. 21–25, 2007.

[4] V. Milutinovic, M. Kotlar, S. Stojanovic, I. Dundic, N. Trifunovic, and Z. Babovic, "Implementing Neural Networks by Using the DataFlow Paradigm," In *DataFlow Supercomputing Essentials*, New York: Springer Cham, 2017, pp. 3–44.

[5] V. Jelisavcic, I. Stojkovic, V. Milutinovic, and Z. Obradovic, "Fast learning of scale-free networks based on Cholesky factorization," *International Journal of Intelligent Systems*, Vol. 33, No. 6, pp. 1322-1339, 2018.

[6] M. J. Flynn, O. Mencer, V. Milutinovic, G. Rakocevic, P. Stenstrom, R. Trobec, and M. Valero, "Moving from petaflops to petadata," *Communications of the ACM*, Vol. 56, No. 5, pp. 39-42, 2013.

[7] A. Kos, S. Tomažič, J. Salom, N. Trifunovic, M. Valero, and V. Milutinovic, "New benchmarking methodology and programming model for big data processing," *International Journal of Distributed Sensor Networks*, Vol. 11, No. 8, 271752, 2015. Available: https://journals.sagepub.com/doi/10.1155/2015/271752

[8] N. Trifunovic, V. Milutinovic, J. Salom, and A. Kos, "Paradigm shift in big data supercomputing: dataflow vs. controlflow," *Journal of Big Data*, Vol. 2, No. 1, pp. 1-9, 2015.

[9] D. G. Murray, F. McSherry, R. Isaacs, M. Isard, P. Barham, and M. Abadi, "Naiad: a timely dataflow system," In Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles, 2013, pp. 439-455.

[10] A. F. Gates, O. Natkovich, S. Chopra, P. Kamath, S. M. Narayanamurthy, C. Olston, ... and U. Srivastava, "Building a high-level dataflow system on top of Map-Reduce: the Pig experience," *Proceedings of the VLDB Endowment*, Vol. 2, No. 2, pp. 1414-1425, 2009.

[11] V. Milutinović, J. Salom, N. Trifunović, and R. Giorgi, *Guide to dataflow supercomputing: Basic Concepts, Case Studies, and a Detailed Example*, Springer Cham, 2015. Available: https://link.springer.com/book/10.1007/978-3-319-16229-4

[12] A. R. Hurson and V. Milutinovic, *Dataflow Processing*. Cambridge, Massachusetts, United States: Academic Press, 2015.

[13] V. Milutinović, B. Furht, Z. Obradović, and N. Korolija, "Advances in high performance computing and related issues," *Mathematical Problems in Engineering*, Vol. 2016, 2016. Available: https://downloads.hindawi.com/journals/mpe/2016/2632306.pdf

[14] S. Stojanović, D. Bojić, and M. Bojović, "An overview of selected heterogeneous and reconfigurable architectures," *Advances in Computers*, Vol. 96, pp. 1-45, 2015.

[15] N. Korolija, T. Djukic, V. Milutinovic, and N. Filipovic, "Accelerating Lattice-Boltzmann method using Maxeler dataflow approach," *The IPSI BgD Transactions on Internet Research*, Vol. 9, No. 2, pp. 34–42, 2013. Available: http://tir.ipsitransactions.org/indexTIR_all.php

[16] S. Stojanović, D. Bojić, and V. Milutinović, "Solving Gross Pitaevskii equation using dataflow paradigm," *The IPSI BgD Transactions on Internet Research*, Vol. 17, 2013. Available: http://ipsitransactions.org/journals/papers/tir/2013july/p4.pdf

[17] I. Stanojević, M. Kovačević, and V. Šenk, "Application of maxeler dataflow supercomputing to spherical code design," In *Exploring the DataFlow Supercomputing Paradigm,* New York: Springer Cham, 2019, pp. 133-168.

[18] N. Bežanić, J. Popović-Božović, V. Milutinović, and I. Popović, "Implementation of the RSA Algorithm on a DataFlow Architecture," *IPSI BgD Transactions on Internet Research*, Vol. 9, No. 2, pp. 11-16, 2013.

[19] N. Korolija, J. Popović, M. Cvetanović, and M. Bojović, "Dataflow-based parallelization of control-flow algorithms," *Advances in Computers*, Vol. 104, pp. 73-124, 2017.

[20] V. Milutinovic, J. Salom, D. Veljovic, N. Korolija, D. Markovic, and L. Petrovic, "Transforming applications from the control flow to the dataflow paradigm," In *Dataflow Supercomputing Essentials,* New York: Springer Cham, 2017, pp. 107-129.

[21] J. Popovic, D. Bojic, and N. Korolija, "Analysis of task effort estimation accuracy based on use case point size," *IET Software*, Vol. 9, No. 6, pp. 166-173, 2015.

[22] N. Trifunovic, V. Milutinovic, N. Korolija, and G. Gaydadjiev, "An AppGallery for dataflow computing," *Journal of Big Data*, Vol. 3, No. 1, pp. 1-30, 2016.

[23] V. Milutinovic, J. Salom, D. Veljovic, N. Korolija, D. Markovic, and L. Petrovic, "Maxeler AppGallery Revisited," In *Dataflow Supercomputing Essentials,* New York: Springer Cham, 2017, pp. 3-18.

[24] A. Hurson, V. Milutinovic, "Special issue on dataflow supercomputing," *Advances in Computers*, Vol. 96, pp. 1–234, 2015.

[25] N. Korolija, D. Bojic, A. R. Hurson, and V. Milutinovic, "A runtime job scheduling algorithm for cluster architectures with dataflow accelerators," *Advances in Computers*, Vol. 126, pp. 201-245, 2022.

[26] V. Milutinović, E. S. Azer, K. Yoshimoto, G. Klimeck, M. Djordjevic, M. Kotlar, ... and I. Ratkovic, "The ultimate dataflow for ultimate supercomputers-on-a-chip, for scientific computing, geo physics, complex mathematics, and information processing," In 2021 10th Mediterranean Conference on Embedded Computing (MECO), 2021, pp. 1-6.

[27] V. Milutinović, N. Trifunović, N. Korolija, J. Popović, and D. Bojić, "Accelerating program execution using hybrid control flow and dataflow architectures," In 2017 25th Telecommunication Forum (TELFOR), 2017, pp. 1-4.

[28] Z. Sustran and J. Protic, "Migration in hardware transactional memory on asymmetric multiprocessor," *IEEE Access*, Vol. 9, pp. 69346-69364, 2021.

[29] J. Popović, V. Jelisavčić, and N. Korolija, "Hybrid Supercomputing Architectures for Artificial Intelligence: Analysis of Potentials," In 1st Serbian International Conference on Applied Artificial Intelligence (SICAAI), Kragujevac, Serbia, 2022.

[30] N. Korolija and A. Zamuda, "On Cloud-Supported Web-Based Integrated Development Environment for Programming DataFlow Architectures," In *Exploring the DataFlow Supercomputing Paradigm,* New York: Springer Cham, 2019, pp. 41-51

[31] 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*, Vol. 34, No. 6, pp. 947-960, 2015.