EMERGING ACCELERATOR PLATFORMS FOR DATA CENTERS[1]

Muhammet Mustafa Ozdal
Computer Engineering Department
Bilkent University
Ankara, Turkey
mustafa.ozdal@cs.bilkent.edu.tr

Today's server architectures are designed considering the needs of a wide range of applications. For example, superscalar processors include complex control logic for out of order execution to extract instruction-level parallelism (ILP) from arbitrary programs. However, not all workloads utilize the features of a superscalar processor effectively. For example, a workload that exhibits a regular execution pattern (e.g. a dense linear algebra kernel) may not require the expensive ILP control logic for parallelism. Instead, it can be run on a throughput-oriented architecture with thousands of simple cores, such as a GPU, which can lead to much better performance and power efficiency. On the other hand, only a limited class of data-parallel applications can utilize the high throughputs provided by such architectures. As a matter of fact, existing CPU and GPU platforms may not be the most efficient choices for the compute patterns of a wide range of applications.

For big data workloads, access to data is typically at least as important bottleneck as computation. The memory subsystems of today's CPU architectures are optimized for workloads that have reasonable data access locality. CPU cache hierarchies include different sizes of caches, which help capture different levels of access localities in different applications. However, if an application exhibits very little or no locality, the data access operations become inefficient for these architectures.

As an example, let us consider graph applications that run on very large and unstructured datasets. Typically, the data of a vertex is computed/updated based on the data of its neighbors. In an unstructured graph, the neighbors of a vertex are stored in memory locations that may be far from each other. So, traversing the neighbors of a vertex may involve a random memory access per neighbor. If the graph is large enough so that it does not fit into the last level cache (LLC), each access to a neighbor's data may require a random DRAM access, which is typically hundreds of clock cycles. However, existing CPU architectures are not optimized for frequent random DRAM accesses. For example, each Intel Haswell Xeon core has 10 line-fill-buffers (LFBs), which means that each core can handle at most 10 L1 cache misses at a given time. However, an off-chip DRAM latency of hundreds of cycles requires hundreds of outstanding memory requests to be able to utilize the full DRAM bandwidth available in the system [1]. It was reported that 10 or more Xeon cores were needed for various graph applications to fully utilize the available DRAM bandwidth [2]. Furthermore, due to the low compute to memory-access ratios in graph applications, these cores are frequently stalled while waiting for data from off-chip memory. This leads to high power consumption by 10+ superscalar cores while not doing useful work. It was shown that custom architectures that target such communication patterns have the potential to improve power efficiency by a factor of 50x or more compared to the general-purpose CPUs [3].

[1] This work has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement no 704476.

It is possible to design domain-specific hardware to achieve significant power and performance improvements for specific workloads. Although custom hardware accelerators can significantly improve power and performance of certain workloads, they may not always be the best choices in terms of data center economy. There may be various reasons for this, such as the overhead of managing heterogeneous resources (each with a potentially different set of custom accelerators) in a reliable way, rapid changes in the workloads relative to the lifetimes of the servers, and the extra cost of designing and manufacturing custom parts for different applications. From these perspectives, FPGAs are good candidates for deployment into data centers because they allow programmability and homogeneity while allowing custom hardware acceleration for different workloads. That is why several vendors have started incorporating FPGAs into their platforms. In the following, we provide a summary of the recent industrial prototypes and the recent research on FPGA and ASIC hardware accelerators.

IBM's Coherent Accelerator Processor Interface (CAPI)

IBM implemented CAPI for their Power8 Servers to make it easy to integrate FPGA and custom accelerators into server processors. CAPI allows attaching an accelerator to the I/O interface of a processor chip without incurring significant device driver and operating system software latencies. Using this interface, accelerators and the host processors can have coherent access to a homogeneous virtual address space, where caching and coherency is managed by special hardware modules. This interface hides from the users the complexities of caching and communications by allowing the user-designed accelerators to access the system memory by simple load and store requests [4].

Different workloads have been accelerated by researchers using CAPI-enabled FPGAs, including genomics algorithms [5, 6], matrix algebra [7], and graph processing [8]. Furthermore, several commercial and special-purpose FPGA and ASIC-based CAPI accelerators are now being developed by different companies for the OpenPOWER platform [9].

Intel's Xeon + FPGA Integrated Platform

Although IBM's CAPI helps reduce the driver and operating system latencies, the communication speed between CPU and FPGA is still limited by the PCIe based interconnect. Intel's prototype Xeon + FPGA platforms try to address this limitation by integrating the host CPU and FPGA in a multi-chip package and enabling communication through the faster QuickPath Interconnect (QPI) interface in addition to PCIe.

Figure 1 shows the high-level hardware/software architecture of the Xeon + FPGA platform [10, 11]. The in-package FPGA has coherent access to the host memory through QPI and PCIe interfaces as shown in this figure. The communication between the host application, the accelerator functional units (AFUs) on

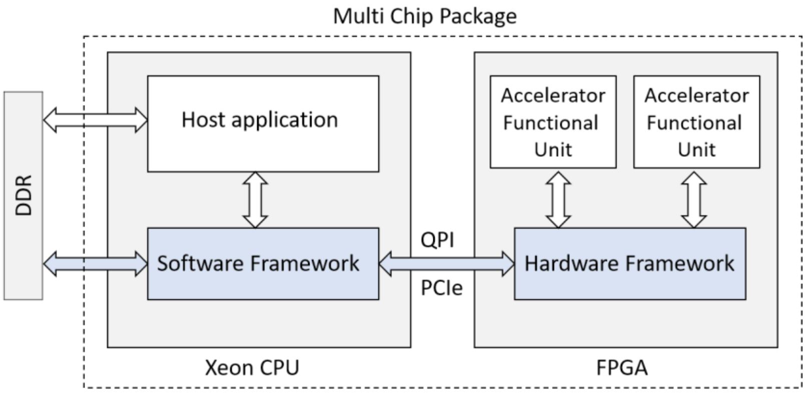

Figure 1: Intel's Xeon+FPGA Integrated Platform [10, 11]

the FPGA, and the system memory is facilitated through the provided software framework (running on the host CPU) and hardware framework (running on the integrated FPGA).

The prototype Xeon + FPGA platforms have recently been made available for academic research and workloads have been optimized in different domains, including genomics [10], machine learning [10], and databases [12, 13, 14].

Microsoft's Configurable Cloud

As part of the original Catapult project [15], Microsoft researchers integrated FPGA accelerators into a production data center to accelerate the search ranking algorithm used by Bing. In this architecture, a single FPGA card was connected to each server and the communication between the host CPU and FPGA was done through PCIe. In addition, 48 FPGA cards within half-a-rack were connected through a secondary mesh network. This allowed the FPGAs within the same network to communicate with each other without going through host CPUs. This fabric was deployed at a medium-scale data center of 1632 servers, and 95% throughput improvement was reported for Bing's web search ranking algorithm on this fabric compared to the pure software implementation.

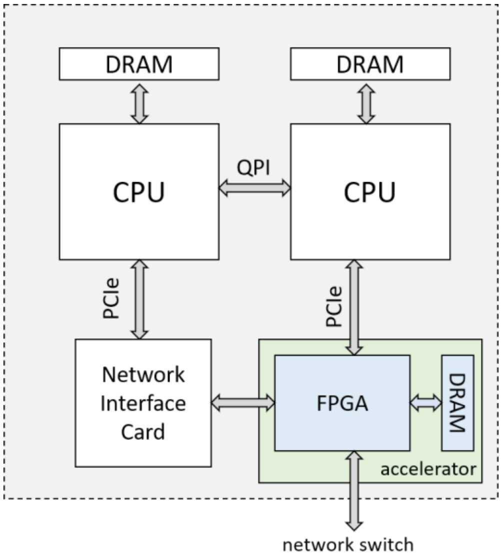

Figure 2: A two-socket Xeon blade server configuration for the Configurable Cloud architecture [16].

Despite the good initial results, the original Catapult architecture was deemed to have severe limitations for real data centers, due to the extra cost of the secondary network, ineffective handling of failures, relatively small number of FPGAs communicating directly with each other, and the limited acceleration opportunities [16].

These limitations were later addressed by a new architecture, called Configurable Cloud (CC), which is reported to have been deployed in most of the new Microsoft datacenters since 2016. A server configuration is illustrated in Figure 2 for a two-socket Xeon blade server [16]. Observe that an FPGA accelerator card has been placed between the network interface card (NIC) and the ethernet network switch. In addition, the FPGA is connected to one of the host CPUs through PCIe.

This configuration is reported to be flexible enough not only for local workload acceleration, but also for acceleration of networking applications and distributed workloads [16]. For local acceleration, a server CPU can communicate with the FPGA through the direct PCIe connection shown in Figure 2. A special network bridge mode allows all network traffic to pass from NIC to the network switch without interacting with the local workload running on the FPGA. For network applications, FPGAs can be treated as “bumps-in-the-wire” so that they can accelerate networking flows without incurring additional loads on the host CPUs. It was shown that host-to-host line-rate encryption/decryption can be performed on these FPGAs without involvement of the host CPUs [16]. This architecture also allows acceleration of distributed applications by enabling low-latency FPGA-to-FPGA communication across the data center without the need for a secondary network among FPGAs. Each FPGA can generate and consume network packets without the interference of the host software.

The Configurable Cloud architecture also offers flexibility in the data center scale. Since hosts can utilize remote FPGAs with low latency, FPGAs can be managed as a global pool of resources orthogonal to the CPU resources. Services may request different numbers of CPUs and FPGAs based on the workloads they run, while failing nodes are removed from the corresponding pools accordingly [16].

Google’s Tensor Processing Unit

Based on a projection that voice-based search will significantly increase the computational demands of Google’s datacenters, a custom ASIC chip – called Tensor Processing Unit (TPU) -- was designed and deployed by Google in 2015 [17]. TPU is aimed at accelerating the inference phase of different types of neural network applications, including multi-layer perceptrons (MLP), convolutional neural networks (CNN), and recurrent neural networks (RNN) [18].

TPU has been designed as a coprocessor connected to the host CPU through PCIe, thus it can be directly plugged into existing server platforms. The host CPU simply sends instructions through PCIe to an instruction buffer, and these instructions are executed by the TPU. The main computational block in the TPU architecture is a *Matrix Multiply Unit*, which can perform 64K multiply-and-add operations per cycle on 8-bit integer operands. In addition, 28MB of software-managed on-chip memory is included to store the intermediate results and the inputs of the Matrix Multiply Unit. The datapath occupies 67% of the TPU floorplan, while the area occupied for control is only 2% [17]. This contrasts with the state-of-the-art server CPUs and GPUs, in which the control structures occupy significant chip area and lead to increased power consumption.

It is reported that the performance of TPU on neural networks is 15 and 29 times better than a K80 GPU and an 18-core Haswell CPU, respectively. In terms of performance per Watt, the corresponding improvements are reported to be 29x and 83x [17]. The main reason for these improvements is that CPUs and GPUs have expensive control logic to extract good performance from different types of applications, while most of the TPU logic is dedicated to a datapath that is specifically designed for a single application domain.

Other Commercial and Research Platforms

There are many other commercial, prototype, and research accelerator platforms that have not been covered in detail in this paper. In this part, we briefly outline some of those platforms.

As the potential benefits of hardware acceleration have become apparent, more and more cloud service providers are making FPGA resources available to their users. For example, Amazon Web Services (AWS) has recently announced the general availability of EC2 F1 compute instances with FPGAs, where each instance can contain up to 8 dedicated FPGA boards connected with a PCIe fabric [19].

At the platform level, FPGAs have been integrated with host CPUs in different ways. An early example is the Cray XD1 hybrid system with 12 AMD Opteron processors and 6 Xilinx Virtex II Pro FPGA coprocessors in a single chassis, where the FPGAs can access the host system memory through a special communications processor [20].

Another early example is the Novo-G supercomputer with 24 compute nodes, each containing a quad-core Xeon CPU and 8 FPGAs connected to each other with a fast interconnect [21]. Although direct communication between FPGAs within the same node is possible, communication between inter-node FPGAs is done through a centralized network (with host involvement), which makes it impractical to accelerate communication-intensive distributed applications. A new version of this system – called Novo-G# -- has been developed recently to alleviate the communication bottleneck by connecting 64 FPGA boards with a dedicated 4x4x4 torus network [22]. This system has been targeted for high performance computing research.

On the commercial side, Convey's hybrid core computers integrate FPGA coprocessors with commodity processors in rack-mountable enclosures to improve performance and power efficiency of certain workloads. The high-level architecture of the Convey HC-2 Computer is shown in Figure 3 [23]. The coprocessor consists of four Xilinx Virtex FPGAs and 16-channel word-addressable scatter-gather DRAM. The communication with host CPU is realized through the hybrid-core memory interconnect (HCMI). The memory subsystem of the coprocessor is physically separate from the host memory subsystem. However, there is shared logical memory -- called hybrid-core globally shared memory (HCGSM) – that can be accessed by both the host CPU and the coprocessor through simple load/store instructions using virtual addresses. The data movement between different physical memories is handled by a data mover engine built into the coprocessor through PCIe interface.

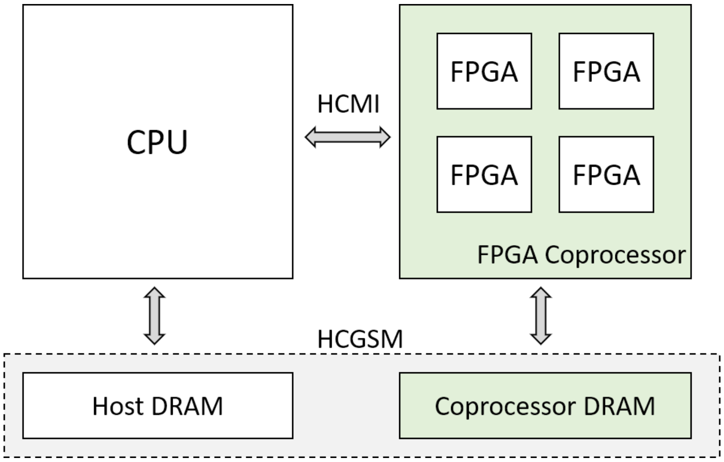

Figure 3: High-level architecture of Convey HC-2 [23], where HCMI stands for "hybrid-core memory interconnect" and HCGSM stands for "hybrid-core globally shared memory".

While most of the previously mentioned platforms view FPGAs as coprocessors connected to CPUs through standard interfaces (e.g. PCIe), it is also possible to decouple FPGAs from CPUs. Weerasinghe, et. al. have proposed a hyperscale data center platform where FPGAs can be connected to the network as standalone appliances [24]. In their proposed architecture, a module contains an FPGA and an optional off-chip memory, where the FPGA has three main parts: 1) custom user logic, 2) network service layer to enable communication within the data center, and 3) a management layer to enable virtual memory accesses and remote management of this module. Furthermore, the authors have proposed a new provisioning service for OpenStack so that standalone FPGAs can be requested by cloud users. In a later work, this architecture has been implemented as prototype, and a distributed text analytics application has been ported onto this multi-FPGA fabric with significant performance improvements compared to PCIe-attached FPGAs [25].

FPGAs allow flexibility by allowing application-specific hardware to be reprogrammed as the workloads change over time. However, this flexibility comes at the expense of area, power, and performance overheads compared to application specific integrated circuits (ASICs). For frequently executed and power/performance-critical workloads, it may make economic sense to integrate domain-specific ASIC accelerators. Google's Tensor Processing Unit described above is an example that is targeted for neural network applications. Other workloads of interest that may justify ASIC accelerators include cryptography [26], compression [27], machine learning [28], database [29], and large-scale graph processing [3, 30, 31].

Even if computational bottlenecks can be alleviated through the use of accelerators, memory access costs can become the limiting factors for achieving substantial power and performance improvements, especially for big data workloads. There have been efforts to bring computation closer to where the data resides. For example, Minerva is an FPGA-based solid-state drive (SSD) architecture, which allows offloading computation code to the FPGA modules inside SSD [32]. Such an architecture is especially well-suited for applications with poor memory access locality, because it avoids moving data from disk to CPU across a slow I/O interface through main memory and multiple levels of cache hierarchies.

Other near-data processing (NDP) architectures propose to integrate simple compute units next to memory controllers to avoid moving data from main memory to CPU. Vermij et. al. have proposed a system-level architecture for an NDP-enhanced multi-core CPU, and they have studied various issues such as data placement, coherency, communication, and virtual memory support [33]. Similarly, Xi et. al. have proposed an NDP accelerator, called JAFAR, for common database operations [34].

It is also possible to take NDP one step further to perform computations *inside* memory. Although processing in memory (PIM) computing paradigm was proposed in 1990s to overcome the memory wall problems [35, 36, 37, 38], it was not deemed practical in industry due to the high costs of integration. However, with the advanced 3D integration technologies available today, the PIM concept has gained renewed research interest due to the potential of practical feasibility.

Hybrid Memory Cube (HMC) architecture was proposed by Micron Technology as a high-performance 3D memory [39]. HMC consists of four-to-eight layers of stacked DRAM dies with an additional logic die at the base. Since then, different research groups have proposed utilizing this logic die for processing in memory. For example, Pugsley, et. al. have proposed adding general-purpose in-order cores to the base die of a stacked DRAM to run Map-Reduce workloads more efficiently [40]. For throughput-oriented computing, the TOP-PIM architecture has been proposed to add programmable GPU compute units inside stacked DRAM to achieve significant energy efficiency improvements compared to the traditional GPU architectures [41]. Active Memory Cube (AMC) architecture is another example, which targets scientific exa-scale computing by adding vector processing elements at the base layer of a 3D memory platform [42]. For large-scale graph processing, Ahn et. al. have proposed the Tesseract architecture, where programmable graph accelerators are integrated into a 3D memory [43].

In conclusion, we are starting to see a shift in data center platforms towards more customizable hardware to achieve energy efficiency and performance improvements. There are open algorithmic research problems, because existing algorithms targeted at traditional von Neumann architectures may not be optimal for these emerging heterogeneous platforms. There are also open research problems about how to choose the best architecture and how to design the most efficient hardware for a domain of applications. With the wide-spread availability of FPGA accelerators, there is a need to raise the abstraction layer for hardware design to enable more application developers to utilize these systems. We are already seeing improvements in high-level design and synthesis tools to allow C or OpenCL-based application development for FPGAs. In summary, these platforms have brought new research opportunities across multiple domains, including algorithms, architecture, design, automation, and technology.

Bibliography

[1] M. Ozdal, S. Yesil, T. Kim, A. Ayupov, S. Burns and O. Ozturk, "Architectural requirements for energy efficient execution of graph analytics applications," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2015.

[2] S. Beamer, K. Asanovic and D. Patterson, "Locality exists in graph processing: Workload characterization on an ivy bridge server," in *IEEE International Symposium on Workload Characterization (IISWC)*, 2015.

[3] M. Ozdal, S. Yesil, T. Kim, A. Ayupov, J. Greth, S. Burns and O. Ozturk, "Energy efficient architecture for graph analytics accelerators," in *ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2016.

[4] J. Stuecheli, B. Blaner, C. R. Johns and M. S. Siegel, "CAPI: A Coherent Accelerator Processor Interface," *IBM Journal of Research and Development,* vol. 59, no. 1, pp. 1-7, 2015.

[5] M. Ito and M. Ohara, "Power-efficient FPGA accelerator: Systolic array with cache-coherent interface for pair-HMM algorithm," in *IEEE Symposium in Low-Power and High-Speed Chips (COOL CHIPS XIX)*, Yokohama, Japan, 2016.

[6] M. J. Jaspers, *Acceleration of read alignment with coherent attached FPGA coprocessors, MSc Thesis,* 2015.

[7] C.-C. Chung, C.-K. Liu and D.-H. Lee, "FPGA-based accelerator platform for big data matrix processing," in *IEEE International Conference on Electron Devices and Solid-State Circuits (EDSSC)*, 2015.

[8] J. Lee, H. Kim, S. Yoo, K. Choi, H. P. Hofstee, G.-J. Nam, M. R. Nutter and D. Jamsek, "ExtraV: boosting graph processing near storage with a coherent accelerator," *Proceedings of the VLDB Endowment,* vol. 10, no. 12, pp. 1706-1717, 2017.

[9] A. Shilov, "Several CAPI-Enabled Accelerators for OpenPOWER Servers Revealed," 12 April 2016. [Online]. Available: https://www.anandtech.com/show/10240/several-capi-accelerators-for-openpower-revealed. [Accessed 18 October 2017].

[10] P. K. Gupta, "Accelerating Datacenter Workloads," [Online]. Available: http://www.fpl2016.org/slides/Gupta%20--%20Accelerating%20Datacenter%20Workloads.pdf. [Accessed 18 October 2017].

[11] D. Sheffield, "IvyTown Xeon + FPGA: The HARP Program," 2016. [Online]. Available: https://cpufpga.files.wordpress.com/2016/04/harp_isca_2016_final.pdf.

[12] M. Owaida, D. Sidler, K. Kara and G. Alonso, "Centaur: A Framework for Hybrid CPU-FPGA Databases," in *IEEE Int'l Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2017.

[13] D. Sidler, M. Owaida, X. Istvan, K. Kara and G. Alonso, "doppioDB: A hardware accelerated database," in *Int'l Conference on Field Programmable Logic and Applications (FPL)*, 2017.

[14] D. Sidler, Z. Istvan, M. Owaida and G. Alonso, "Accelerating Pattern Matching Queries in Hybrid CPU-FPGA Architectures," in *ACM International Conference on Management of Data (SIGMOD)*, 2017.

[15] A. Putnam, A. M. Caulfield, E. S. Chung, D.Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, J. Gray, M. Haselman, S. Hauck, S. Heil, A. Hormati, J.-Y. Kim, S. Lanka, J. R. Larus, E. Peterson, G. Prashanth, A. Smith, J. Thong, P. Y. Xiao and D, "A reconfigurable fabric for accelerating large-scale datacenter services," in *International Symposium on Computer Architecture (ISCA)*, 2014.

[16] A. Caulfield, E. Chung, A. Putnam, H. Angepat, J. Fowers, M. Haselman, S. Heil, M. Humphrey, P. Kaur, J.-Y. Kim, D. Lo, T. Massengil, K. Ovtcharov, M. Papamichael, L. Woods, S. Lanka, D. Chiou and D. Burger, "A Cloud-Scale Acceleration Architecture," in *International Symposium on Microarchitecture (MICRO)*, 2016.

[17] N. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, Borchers, A. and R. Boyle, "In-Datacenter Performance Analysis of a Tensor Processing Unit," in *International Symposium on Computer Architecture (ISCA)*, 2017.

[18] Y. LeCun, Y. Bengio and G. Hinton, "Deep Learning," *Nature,* vol. 521, no. 7553, pp. 436-444, 2015.

[19] J. Barr, "EC2 F1 Instances with FPGAs - Now Generally Available," 19 April 2017. [Online]. Available: https://aws.amazon.com/blogs/aws/ec2-f1-instances-with-fpgas-now-generally-available/.

[20] D. Strenski, "The Cray XD1 Computer and its Reconfigurable Architecture," 11 July 2005. [Online]. Available: http://www.ncsa.illinois.edu/Conferences/RSSI/2005/docs/Strenski.ppt.

[21] A. George, H. Lam and G. Stitt, "Novo-G: At the forefront of scalable reconfigurable supercomputing," *Computing in Science & Engineering,,* vol. 13, no. 1, pp. 82-86, 2011.

[22] A. D. George, M. C. Herbordt, H. Lam, A. G. Lawande, J. Sheng and C. Yang, "Novo-G#: Large-scale reconfigurable computing with direct and programmable interconnects," in *High Performance Extreme Computing Conference (HPEC)*, 2016.

[23] Convey Computer, "The Convey HC-2 Computer," 29 10 2015. [Online]. Available: https://www.micron.com/resource-details/c803edd0-ff6a-4807-b08c-b0a2d75e7156.

[24] J. Weerasinghe, F. Abel, C. Hagleitner and A. Herkersdorf, "Enabling FPGAs in hyperscale data centers," in *IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and IEEE 12th Intl Conf on Autonomic and Trusted Computing and IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*, 2015.

[25] J. Weerasinghe, R. Polig, F. Abel and C. Hagleitner, "Network-attached FPGAs for data center applications," in *IEEE International Conference on Field-Programmable Technology (FPT)*, 2016.

[26] L. Bossuet, M. Grand, L. Gaspar, V. Fischer and G. Gogniat, "Architectures of flexible symmetric key crypto engines—a survey: From hardware coprocessor to multi-crypto-processor system on chip," *ACM Computing Surveys,* vol. 45, no. 4, 2013.

[27] AHA Products Group, "AHA 374/ AHA 378: PCI Express Compression and Decompression Accelerator Card," [Online]. Available: http://www.aha.com/Uploads/aha374-378_brief_rev_c1.pdf. [Accessed 6 November 2017].

[28] Y. Chen, T. Chen, Z. Xu, N. Sun and O. Temam, "DianNao family: energy-efficient hardware accelerators for machine learning," *Communications of the ACM,* vol. 59, no. 11, pp. 105-112, 2016.

[29] S. Haas, O. Arnold, S. Scholze, S. Höppner, G. Ellguth, A. Dixius, A. Ungethüm, E. Mier, B. Nöthen, E. Matúš and S. Schiefer, "A database accelerator for energy-efficient query processing and optimization.," in *Nordic Circuits and Systems Conference (NORCAS)*, 2016.

[30] M. Ozdal, S. Yesil, T. Kim, A. Ayupov, J. Greth, S. Burns and O. Ozturk, "Graph Analytics Accelerators for Cognitive Systems," *IEEE Micro,* vol. 37, no. 1, pp. 42-51, 2017.

[31] T. Ham, L. Wu, N. Sundaram, N. Satish and M. Martonosi, "Graphicionado: A high-performance and energy-efficient accelerator for graph analytics.," in *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2016.

[32] A. De, M. Gokhale, R. Gupta and S. Swanson, "Minerva: Accelerating data analysis in next-generation SSDs," in *IEEE 21st Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2013.

[33] E. Vermij, C. Hagleitner, L. Fiorin, R. Jongerius, J. van Lunteren and K. Bertels, "An architecture for near-data processing systems.," in *ACM International Conference on Computing Frontiers*, 2016.

[34] S. Xi, O. Babarinsa, M. Athanassoulis and S. Idreos, "Beyond the wall: Near-data processing for databases," in *ACM International Workshop on Data Management on New Hardware*, 2015.

[35] M. Gokhale, B. Holmes and K. Iobst, "Processing in memory: The Terasys massively parallel PIM array. Computer, 28(4), pp.23-31.," *Computer,* vol. 28, no. 4, pp. 23-31, 1995.

[36] M. Hall, P. Kogge, J. Koller, P. Diniz, J. Chame, J. Draper, J. LaCoss, J. Granacki, J. Brockman, A. Srivastava and W. Athas, "Mapping irregular applications to DIVA, a PIM-based data-intensive architecture," in *ACM/IEEE Conference on Supercomputing*, 1999.

[37] P. Kogge, "EXECUBE-a new architecture for scaleable MPPs," in *IEEE International Conference on Parallel Processing (ICPP)*, 1994.

[38] M. Oskin, F. T. Chong and T. Sherwood, "Active pages: A computation model for intelligent memory," in *International Symposium on Computer Architecture (ISCA)*, 1998.

[39] J. T. Pawlowski, "Hybrid Memory Cube (HMC)," in *IEEE Hot Chips Symposium (HCS)*, 2011.

[40] S. Pugsley, J. Jestes, H. Zhang, R. Balasubramonian, V. Srinivasan, A. Buyuktosunoglu, A. Davis and F. Li, "NDC: Analyzing the impact of 3D-stacked memory+ logic devices on MapReduce workloads," in *IEEE International Symposium on Performance Analysis of Systems and and Software (ISPASS)*, 2014.

[41] D. Zhang, N. Jayasena, A. Lyashevsky, J. Greathouse, L. Xu and M. Ignatowski, "TOP-PIM: throughput-oriented programmable processing in memory," in *ACM International Symposium on high-performance parallel and distributed computing*, 2014.

[42] R. Nair, S. Antao, C. Bertolli, P. Bose, J. Brunheroto, T. Chen, C. Cher, C. Costa, J. Doi, C. Evangelinos and B. Fleischer, "Active memory cube: A processing-in-memory architecture for exascale systems.," *IBM Journal of Research and Development,* vol. 59, no. 2/3, 2015.

[43] J. Ahn, S. Hong, S. Yoo, O. Mutlu and K. Choi, "A scalable processing-in-memory accelerator for parallel graph processing," in *Proc. of the ACM/IEEE Int'l Symp. on Computer Architecture*, 2015.

[44] M. C. Popescu, V. E. Balas, L. Perescu-Popescu and N. Mastorakis, "Multilayer perceptron and neural networks," *WSEAS Transactions on Circuits and Systems,* vol. 8, no. 7, pp. 579-588, 2009.

[45] Y. Kang, W. Huang, S. Yoo, D. Keen, Z. Ge, V. Lam, P. Pattnaik and J. Torrellas, "FlexRAM: Toward an advanced intelligent memory system.," in *IEEE International Conference on Computer Design (ICCD)*, 2012.