Hardware Accelerator Design for Data Centers
Invited Paper

Serif Yesil†, Muhammet Mustafa Ozdal*, Taemin Kim*, Andrey Ayupov*, Steven Burns*, and Ozcan Ozturk†
* {mustafa.ozdal, taemin.kim, andrey.ayupov, steven.m.burns}@intel.com
Intel Corp.
Hillsboro, OR 97124
† {serif.yesil, ozturk}@cs.bilkent.edu.tr
Bilkent Univ.
Ankara, Turkey

***Abstract*—As the size of available data is increasing, it is becoming inefficient to scale the computational power of traditional systems. To overcome this problem, customized application-specific accelerators are becoming integral parts of modern system on chip (SOC) architectures. In this paper, we summarize existing hardware accelerators for data centers and discuss the techniques to implement and embed them along with the existing SOCs.**

I. Introduction

With the end of Dennard scaling, computing systems are becoming increasingly power limited. New transistor technologies allow us to pack more logic in a chip, but only a small fraction of available logic gates can be used at a given time due to power limitations; this phenomenon is known as dark silicon. Esmaeilzadeh et al. [1] have predicted that in the next ten years, more than 50% of chip area will remain unpowered for 8nm process technologies. Although this is a serious limitation, it also brings new opportunities for energy efficient computation. One possible way to address this problem is to add custom hardware accelerators targeted for specific tasks which are significantly more efficient in terms of power and performance.

While power limitation has been becoming a bottleneck, the size of data processed online has been increasing rapidly. Google estimates that the total number of web pages active today exceeds 1 trillion. Similarly, social networks have been growing exponentially. For example, number of Facebook members increased from 1 million to 1 billion between 2004 and 2012 [2]. In addition to online data, scientific advancements are forcing us to have more processing power. Healthcare and genome research are examples of these scientific areas. [3] shows that the genome sequencing costs have been dropping faster than Moore's law, which makes genome sequencing affordable for many people. This implies that huge amount of genome data is becoming part of the health care industry.

To deal with big data, companies and government establishments are investing on large data centers. According to NRDC, data centers in U.S. consumed 91 billion KW hours of energy, which is the annual output of 34 power plants with 500 MW capacity. It is predicted that in 2020, data centers will consume 140 billion KW hours of energy [4].

Taylor et.al. [5] explain several techniques to overcome the dark silicon problem. In this work, authors discuss Coda (co-processor dominated architectures) as a potential solution which is also the focus in this paper. In such a system it is expected to have several custom & reconfigurable accelerators. Coda systems are 100x-1000x more energy efficient than existing general purpose processors. As stated in [5], reconfigurable logic may become handy in the dark silicon era by integrating field programmable gate arrays (FPGAs), coarse-grain reconfigurable arrays (CGRAs) along with processors. Also, many data centers in the world run a small subset of dedicated workloads such as search engines from Google and Bing, recommendation systems from Amazon and Netflix, etc. It is predicted by ITRS that custom on-demand accelerators will become important parts of existing SOC systems [6].

In the rest of this paper, we will start with a brief survey on existing general purpose and custom accelerators in Section II. Then, in Section III, we will discuss integration techniques for accelerator rich systems, followed by a brief introduction for design and programming techniques of custom accelerators in Section IV. Finally, we will switch our focus to graph applications and briefly summarize our findings in Section V.

II. Accelerators

A. Accelerators in Data Centers

There are several types of accelerators that are used in data centers today. Most commonly used ones are embedded accelerators(ISA extensions), and general purpose graphics processing units (GPUs).

High-performance general purpose processors may include vector instruction extensions to enable SIMD style of computation. SSE and AVX extensions are examples used in x86 architectures, and they are capable of processing packed integers and floating point values. Moreover, AVX provides other extensions for certain applications such as cryptography, signal processing etc. For example, AVX-512 implements effective acceleration for secure hash algorithms like SHA1 and SHA256.

A real world example of AVX usage for big data acceleration is IBM DB2 database [7]. Collaboration between IBM and Intel on IBM DB2 with BLU acceleration improved the performance of query processing by 246 times [8].

However, IBM's wire-speed processor might be considered as one of the first accelerator rich SOCs. Wire-speed

978-1-4673-8388-2/15/$31.00 ©2015 IEEE

processor includes conventional processing cores which cover 60% of chip area and 4 different accelerators working with the processors. Wire-speed processor includes specialized units for common data center operations such as a compression/decompression unit, XML parser unit, cryptography unit, and a regular expression, pattern matching unit. These 4 common operations can target applications from different domains such as databases, service-oriented architectures, and secure multitenant cloud computing.

In parallel, GPUs have also been used extensively for data analytics and big data applications. In June 2015, 15 super computers among top 100 of top500 list [9] include NVIDIA GPUs. Furthermore, IBM and NVIDIA are collaborating on integration of GPUs in data centers [10]. OpenPower initiative has allowed many data analytic applications to be ported to GPUs. They also target a major problem in GPU integration: implementation of a high speed link for CPU and GPU interconnection. The initiative is working on NVLink interconnect to overcome this problem.

GPU computation also attracts many researchers from different fields due to the availability of high processing power. An example of big data acceleration is Mars [11]. Mars is a successful implementation of MapReduce framework on GPUs, which takes away the complexity of GPU programming models and brings the familiar MapReduce framework to GPUs. Mars also implements synchronization systems that are required for MapReduce systems. GPUGrid [12] project is another project that tries to accelerate highly computational tasks on GPUs. There are also GPU libraries such as GpuMiner [13] that implement various algorithms.

Especially, machine learning algorithms that are used for big data solutions are accelerated in several libraries. Caffe is an example of deep learning library [14]. The work in [15] is another example which focuses on deep neural networks. NVIDIA's CUDA framework also provides useful libraries for sparse matrix vector multiplications (SpMxV) like calculations in their CuBLAS library.

Several benchmarks also adapted many data mining and data analysis applications for GPUs, such as Rodinia [16] and Parboil [17] benchmarks.

There have also been significant attention on accelerating graph applications on GPUs. In [18], authors propose a warp-centric execution model to avoid control divergence and work imbalances in irregular graph applications. Additionally, Medusa [19] is a processing framework which focuses on bulk synchronous processing and is targeted for GPUs. They also consider multi-GPU acceleration and optimize graph partitioning to reduce the communication between GPUs. Another benchmark suite called Lonestar [20] targets irregular graph applications for GPUs.

It is well known that GPUs are best at accelerating massively parallel applications with regular computational patterns. Some recent works have proposed architectural improvements for GPUs to target irregular applications. For example, [21] proposes a hardware worklist mechanism for GPUs to make it feasible for irregular applications to have data driven execution.

B. Custom & Reconfigurable Logic Accelerators

Custom & reconfigurable accelerators are becoming increasingly more popular. FPGAs are used in different ways to accelerate applications and they are now becoming part of clusters and data centers. For example, Catapult [22] is an example of an FPGA system created by Microsoft targeted at data centers. On the other hand, CGPA [23] is a study that tries to extract parallelism using HLS synthesis. Another example, Cube [24], is a system that can integrate 512 FPGAs to create a cluster environment. Axel [25] is a heterogeneous computing cluster, where a node can include multiple types of accelerators such as FPGAs and GPUs. Recently, a new cluster system called Saturn 1 is released by SRC Computers. This system is composed of a single conventional microprocessor and a huge reconfigurable logic [26]. Yoshimi et al. [27] proposes an FPGA based accelerator that is tightly coupled with the flash storage and optical network interface. It is a complete system which has high level resource sharing in terms of accesses from FPGA.

Moreover, a different study called FPMR [28] has shown a successful implementation of MapReduce framework on FPGAs. ZCluster [29] is a work that focuses on both MapReduce [30] and Hadoop [31] frameworks. By using a specific bus design between master processor and slave processing elements, ZCluster achieves the streaming behavior of Hadoop systems.

In addition to the aforementioned FPGA accelerators, there are several reconfigurable accelerator designs for general purpose computing. DySER [32] is a reconfigurable general purpose accelerator design example, which uses compile time profiling to extract dataflows from a given program and reconfigures a general purpose heterogeneous functional unit and creates a specific datapath for the program execution. Other examples of such a system are VEAL [33], PPA [34], and CHARM [35].

FPGAs are also widely used to accelerate specific types of applications. PageRank, belief propagation, and neural networks are some of them. McGettrick et al. [36] proposes an FPGA implementation of eigenvector based PageRank algorithm. Similarly, [37] and [38] are examples of belief propagation algorithms implemented on FPGAs.

DianNao [39] is an example of application specific hardware which is designed for accelerating Neural Networks by capturing their common operations. The motivation behind this research work is twofold: 1) An application specific design allows small footprint per accelerator, which allow including a rich set of accelerators in the system. 2) Many modern applications can be solved by using neural networks [40].

There have also been studies about graph applications. A recent work is PIM (processing in memory) [41]. This work provides a system that uses 3D integration technology, and tries to maximize the available memory bandwidth. On the other hand, GraphGen [42] is a framework to create application-specific synthesized graph processors and memory layout for FPGAs. GraphGen also uses a vertex centric execution model to represent graph applications. GraphStep [43] implements a bulk synchronous message passing execution model on FPGAs for graph applications.

III. System Integration of Accelerators - Accelerator Rich Systems

While application specific accelerators promise significant power and performance improvements, integration of these accelerators to existing systems is still an open problem. Following are some of the issues that need to be addressed by architects and designers:

- Interconnection between host CPU to accelerators and accelerator to accelerator
- Memory hierarchy design for accelerators
- Programming and management of accelerators

A. Interconnect

While accelerator design is a widely researched topic, design of interconnect for communication between cores, uncore elements and DRAM has not drawn as much attention. Cong et al. [44] state that accelerators are generally >100x faster than traditional cores and they require higher memory bandwidth. They address this problem by using a high throughput crossbar interconnect.

Beside the on chip accelerators, systems which have multiple FPGAs connected to each other may also suffer from communication latency if they utilize the traditional inter-node communication mechanisms. Directly connecting the accelerators through a special network can be a better option, especially for big data applications where the data is distributed among many nodes. For example, Catapult [22] uses a dedicated high speed torus network between accelerators in different nodes.

B. Memory Hierarchy

Coupling between system memory components and accelerators can be categorized as follows:

- **Tightly coupled:** This kind of accelerators are placed very close to the core. It has access to the whole memory hierarchy [45].
- **Loosely coupled shared memory:** These accelerators share the same system memory with the host to which they have reasonably fast access. They can be connected to the last level cache (LLC) or to DRAM [45].
- **Separate memory:** GPUs and Intel's Xeon Phi are examples of this category. These accelerators have their own memory subsystems with independent address space. Communication between host memory and device memory is provided by a dedicated interconnect such as PCIe.

C. Programming and Management

Typically, special mechanisms are provided for programming and management of accelerators. For example, special ISA extensions are typically provided for tightly-coupled accelerators such as SSE and AVX. Another way is to provide special APIs for accelerators such as Nvdia's CUDA library. In addition, programming environments such as IBM's Coherent Accelerator Processor Interface (CAPI) [46] and Intel's QuickAssist Accelerator Abstraction Layer [47] provide generic interfaces to manage all accelerators connected to a system. These interfaces can create a coherent memory representation between the host CPU and accelerators.

IV. Designing Custom Accelerators

Different languages and design tools can be used to build custom accelerators. RTL design is still very common, however, it is time consuming and hard to use especially for domain experts and software programmers.

When many custom accelerators need to be designed (e.g. on a reconfigurable fabric), fast turn around time becomes important. Therefore, higher level design languages such as C/C+ or SystemC can be used in conjunction with high level synthesis (HLS) tools. Other high level languages can be used as input to HLS tools such as Java, Python, OpenCL, and CUDA. A survey on HLS models can be found in [48].

High level power and performance analysis is also important for design space exploration. Aladdin [49] is an example that enables power and performance estimation quickly without going through the detailed design process. Instead, it creates and analyzes dynamic data dependence graphs for quick estimation.

V. Case Study: Graph Applications and Irregular Applications

Graph analysis is becoming very popular these days. While analysis of social networks gives us insights about human behavior, biological networks are used for finding flow of diseases or finding genetic relations between diseases. Applications like centralities, community detection, graph sampling, shortest paths, markov networks, graph matching are very common in graph applications domain [50].

The importance of graph applications in the context of big data has increased in this decade with the rise of internet and social networks. By 2008, Google claimed to have indexed one trillion pages. In 2012, Facebook had one billion active users and 140.3 billion friend connections.

Beside all social network and web graph data, big brain graph is a new challenge for big graph processing. It is expected that there will be 10 trillion vertices and 100 trillion edges in human brain graph, where neurons are considered as vertices and synapses as edges. It is stated that the graph of our brain would occupy over one petabyte [2].

A. Big Data Solutions for Graph Applications

Pregel [51] is one of the most well known parallelization frameworks for graph applications. Pregel follows an abstraction technique called *think like a vertex* to parallelize graph algorithms mainly focused on web applications. Pregel's framework follows a bulk synchronous execution model with supersteps separated by barriers. It allows execution of large number of vertex programs in parallel. Pregel API provides a message passing mechanism for vertices to communicate and transfer data between each other across different supersteps. Graphlab/PowerGraph is a more recent framework that follows the vertex parallel execution model, and is getting attraction

especially for irregular applications that prefer asynchronous execution.

Other systems that are used for graph processing are Giraph, Socialite [52], CombBLAS [53] and Galois [54]. Socialite is a domain specific graph processing language, which defines recursive operations. On the other hand, Galois provides 3 types of high level structures which are foreach loops, data structures which can be accessed in parallel, and work-lists to be iterated by foreach constructs. Galois does not force a specific programming paradigm.

Among all solutions, Graphlab provides a more flexible and a promising framework. Both asynchronous execution mode and sequential consistency are favorable options. We will explain gather apply scatter model and aforementioned options in detail in the following subsections.

1) Gather-Apply-Scatter Model: Graphlab uses the Gather-Apply-Scatter (GAS) model to define vertex programs. For a given vertex v, the vertex program goes through these three stages in order. In the Gather stage, the incoming edges of v are iterated over, and a reduction operation is performed to compute an accumulated data object. In the Apply stage, this accumulated value is used together with the old data of v to compute v's new data. In the scatter stage, the result of the Apply stage is distributed to the outgoing edges.

2) Synchronous and Asynchronous Execution Model: Another advantage of Graphlab is the asynchronous execution model as opposed to Pregel, which provides a bulk synchronous model. Synchronous execution model has 2 drawbacks: 1) costly barriers that separate supersteps, 2) slower convergence in general. In the asynchronous execution mode, there are no well defined iterations and no explicit synchronization. Instead, a set of active vertices is maintained during executions. In the scatter stage of vertex v, if the data of v has changed significantly, its neighbors are scheduled for future execution by adding them to the active set. Vertices are processed until the active set is empty.

Another distinction is related to the way data is accessed by neighboring vertices. In the synchronous execution model, data from the previous iteration is accessed by the neighbors. In contrast, asynchronous model allows access to the latest data available. For iterative solvers, the synchronous model corresponds to Jacobi iterations, whereas the asynchronous model corresponds to Gauss Seidel. For example, for the PageRank application, it was shown that Gauss Seidel iterations converge by up to 2x faster compared to Jacobi [55]. The authors of [56] have shown that asynchronous model has better convergence characteristics for other iterative graph applications as well.

Sequential consistency can be important in the context of asynchronous execution model. Specifically, sequential consistency ensures that there exists a serial order corresponding to the parallel execution of different vertex programs. For some applications (e.g. Gibbs Sampling), sequential consistency is needed for correctness, while it can improve convergence of some other applications (e.g. Alternating Least Squares) [56]. However, sequential consistency can be costly to implement on traditional systems due to the fine grain locking mechanisms needed.

B. Custom Accelerator Design for Graph Applications

There are many common operations for graph applications such as vertex and edge data access, synchronization and communication between neighboring vertices, maintenence of the active set of vertices, etc. Graphlab [56] uses the GAS model to separate the application-specific operations from the low-level common operations. This allows domain experts to specify their programs without worrying about the implementation details related to distributed computing.

A similar approach can be used for hardware accelerator design of graph applications. We are currently investigating a customizable hardware template that allows generating hardware for different graph applications easily. While this work is still in progress, we expect it to be very beneficial for graph parallel applications.

VI. Conclusion

In this paper, we have surveyed various aspects of hardware accelerators in the context of data centers. We have also focused on graph applications as an example, and discussed different modeling options in the context of big data processing.

References

[1] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *Proceedings of the 38th Annual International Symposium on Computer Architecture*, ISCA '11, (New York, NY, USA), pp. 365–376, ACM, 2011.

[2] "Big graph." https://www.nsa.gov/research/tnw/tnw204/article3.shtml. Accessed: August 14, 2015.

[3] "Genome sequencing costs." http://www.genome.gov/images/content/costpergenome_apr2015.jpg. Accessed: August 14, 2015.

[4] "Data center efficiency assessment: Scaling up energy efficiency across the data center industry: Evaluating key drivers and barriers." https://www.nrdc.org/energy/files/data-center-efficiency-assessment-IP.pdf. Accessed: August 14, 2015.

[5] M. Taylor, "Is dark silicon useful? harnessing the four horsemen of the coming dark silicon apocalypse," in *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, pp. 1131–1136, June 2012.

[6] "International technology roadmap for semiconductors 2007 edition system drivers." http://www.itrs.net/. Accessed: August 14, 2015.

[7] "Intel takes the flexibility of ibm db2 with blu acceleration even further." http://www.intel.com/content/dam/www/public/us/en/documents/solution-briefs/xeon-e5-v3-ibm-db2-blu-solution-brief.pdf. Accessed: August 14, 2015.

[8] "How intel and ibm did big data 148x better." https://communities.intel.com/community/itpeernetwork/datastack/blog/2014/03/10/how-intel-and-ibm-did-big-data-148x-better. Accessed: August 14, 2015.

[9] "Top500 june 2015 list." http://www.top500.org/list/2015/06/. Accessed: August 14, 2015.

[10] "ibm, nvidia, u.s. dept. of energy outline supercomputing centers, collaboration." http://www.zdnet.com/article/ibm-nvidia-u-s-dept-of-energy-outline-supercomputing-centers-collaboration/. Accessed: August 14, 2015.

[11] B. He, W. Fang, Q. Luo, N. K. Govindaraju, and T. Wang, "Mars: A mapreduce framework on graphics processors," in *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, PACT '08, (New York, NY, USA), pp. 260–269, ACM, 2008.

[12] "Top500 june 2015 list." http://www.top500.org/list/2015/06/. Accessed: August 14, 2015.

[13] W. Fang, K. K. Lau, M. Lu, X. Xiao, P. Y. Y. Chi Kit Lam, B. He1, Q. Luo, P. V. Sander, and K. Yang, "Parallel data mining on graphics processors," Tech. Rep. CS08-07, HKUST, Oct. 2008.

[14] "Caffe deep learning framework." http://caffe.berkeleyvision.org/. Accessed: August 14, 2015.

[15] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, and Y. Bengio, "Theano: new features and speed improvements," *CoRR*, vol. abs/1211.5590, 2012.

[16] S. Che, M. Boyer, J. Meng, D. Tarjan, J. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on*, pp. 44–54, Oct 2009.

[17] J. A. Stratton, C. Rodrigues, I.-J. Sung, N. Obeid, L.-W. Chang, N. Anssari, G. D. Liu, and W.-m. Hwu, "Parboil: A revised benchmark suite for scientific and commercial throughput computing," *Center for Reliable and High-Performance Computing*, 2012.

[18] S. Hong, S. K. Kim, T. Oguntebi, and K. Olukotun, "Accelerating cuda graph algorithms at maximum warp," *SIGPLAN Not.*, vol. 46, pp. 267–276, Feb. 2011.

[19] J. Zhong and B. He, "Medusa: A parallel graph processing system on graphics processors," *SIGMOD Rec.*, vol. 43, pp. 35–40, Dec. 2014.

[20] M. Burtscher, R. Nasre, and K. Pingali, "A quantitative study of irregular programs on gpus," in *Workload Characterization (IISWC), 2012 IEEE International Symposium on*, pp. 141–151, Nov 2012.

[21] J. Y. Kim and C. Batten, "Accelerating irregular algorithms on gpgpus using fine-grain hardware worklists," in *Microarchitecture (MICRO), 2014 47th Annual IEEE/ACM International Symposium on*, pp. 75–87, Dec 2014.

[22] A. Putnam, A. M. Caulfield, E. S. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G. P. Gopal, J. Gray, M. Haselman, S. Hauck, S. Heil, A. Hormati, J.-Y. Kim, S. Lanka, J. Larus, E. Peterson, S. Pope, A. Smith, J. Thong, P. Y. Xiao, and D. Burger, "A reconfigurable fabric for accelerating large-scale datacenter services," in *Proceeding of the 41st Annual International Symposium on Computer Architecuture*, ISCA '14, (Piscataway, NJ, USA), pp. 13–24, IEEE Press, 2014.

[23] F. Liu, S. Ghosh, N. P. Johnson, and D. I. August, "Cgpa: Coarse-grained pipelined accelerators," in *Proceedings of the 51st Annual Design Automation Conference*, DAC '14, (New York, NY, USA), pp. 78:1–78:6, ACM, 2014.

[24] M. Yoshimi, Y. Nishikawa, M. Miki, T. Hiroyasu, H. Amano, and O. Mencer, "A performance evaluation of cube: One-dimensional 512 fpga cluster," in *Reconfigurable Computing: Architectures, Tools and Applications* (P. Sirisuk, F. Morgan, T. El-Ghazawi, and H. Amano, eds.), vol. 5992 of *Lecture Notes in Computer Science*, pp. 372–381, Springer Berlin Heidelberg, 2010.

[25] K. H. Tsoi and W. Luk, "Axel: A heterogeneous cluster with fpgas and gpus," in *Proceedings of the 18th Annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '10, (New York, NY, USA), pp. 115–124, ACM, 2010.

[26] "Src computers launches saturn 1 server, the first reconfigurable hyperscale server." http://srccomputers.com/wp-content/uploads/2015/05/SRC-Press-Release-Saturn-1-Server-052815.pdf. Accessed: August 14, 2015.

[27] M. Yoshimi, R. Kudo, Y. Oge, Y. Terada, H. Irie, and T. Yoshinaga, "An fpga-based tightly coupled accelerator for data-intensive applications," in *Embedded Multicore/Manycore SoCs (MCSoc), 2014 IEEE 8th International Symposium on*, pp. 289–296, Sept 2014.

[28] Y. Shan, B. Wang, J. Yan, Y. Wang, N. Xu, and H. Yang, "Fpmr: Mapreduce framework on fpga," in *Proceedings of the 18th Annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '10, (New York, NY, USA), pp. 93–102, ACM, 2010.

[29] Z. Lin and P. Chow, "Zcluster: A zynq-based hadoop cluster," in *Field-Programmable Technology (FPT), 2013 International Conference on*, pp. 450–453, Dec 2013.

[30] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, pp. 107–113, Jan. 2008.

[31] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, MSST '10, (Washington, DC, USA), pp. 1–10, IEEE Computer Society, 2010.

[32] V. Govindaraju, C.-H. Ho, and K. Sankaralingam, "Dynamically specialized datapaths for energy efficient computing," in *High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on*, pp. 503–514, Feb 2011.

[33] N. Clark, A. Hormati, and S. Mahlke, "Veal: Virtualized execution accelerator for loops," in *Computer Architecture, 2008. ISCA '08. 35th International Symposium on*, pp. 389–400, June 2008.

[34] H. Park, Y. Park, and S. Mahlke, "Polymorphic pipeline array: A flexible multicore accelerator with virtualized execution for mobile multimedia applications," in *Proceedings of the 42Nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, (New York, NY, USA), pp. 370–380, ACM, 2009.

[35] J. Cong, M. A. Ghodrat, M. Gill, B. Grigorian, and G. Reinman, "Charm: A composable heterogeneous accelerator-rich microprocessor," in *Proceedings of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design*, ISLPED '12, (New York, NY, USA), pp. 379–384, ACM, 2012.

[36] S. McGettrick, D. Geraghty, and C. McElroy, "An fpga architecture for the pagerank eigenvector problem," in *Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on*, pp. 523–526, Sept 2008.

[37] J. Perez, P. Sanchez, and M. Martinez, "High memory throughput fpga architecture for high-definition belief-propagation stereo matching," in *Signals, Circuits and Systems (SCS), 2009 3rd International Conference on*, pp. 1–6, Nov 2009.

[38] J. Choi and R. Rutenbar, "Fpga acceleration of markov random field trw-s inference for stereo matching," in *Formal Methods and Models for Codesign (MEMOCODE), 2013 Eleventh IEEE/ACM International Conference on*, pp. 139–142, Oct 2013.

[39] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam, "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," *SIGARCH Comput. Archit. News*, vol. 42, pp. 269–284, Feb. 2014.

[40] T. Chen, Y. Chen, M. Duranton, Q. Guo, A. Hashmi, M. Lipasti, A. Nere, S. Qiu, M. Sebag, and O. Temam, "Benchnn: On the broad potential application scope of hardware neural network accelerators," in *Workload Characterization (IISWC), 2012 IEEE International Symposium on*, pp. 36–45, Nov 2012.

[41] J. Ahn, S. Hong, S. Yoo, O. Mutlu, and K. Choi, "A scalable processing-in-memory accelerator for parallel graph processing," in *Proceedings of the 42Nd Annual International Symposium on Computer Architecture*, ISCA '15, (New York, NY, USA), pp. 105–117, ACM, 2015.

[42] E. Nurvitadhi, G. Weisz, Y. Wang, S. Hurkat, M. Nguyen, J. C. Hoe, J. F. Martnez, and C. Guestrin, "Graphgen: An fpga framework for vertex-centric graph computation," in *Proceedings of the 2014 IEEE 22Nd International Symposium on Field-Programmable Custom Computing Machines*, FCCM '14, (Washington, DC, USA), pp. 25–28, IEEE Computer Society, 2014.

[43] M. deLorimier, N. Kapre, N. Mehta, D. Rizzo, I. Eslick, R. Rubin, T. Uribe, J. Knight, T.F., and A. DeHon, "Graphstep: A system architecture for sparse-graph algorithms," in *Field-Programmable Custom Computing Machines, 2006. FCCM '06. 14th Annual IEEE Symposium on*, pp. 143–151, April 2006.

[44] J. Cong and B. Xiao, "Optimization of interconnects between accelerators and shared memories in dark silicon," in *Computer-Aided Design (ICCAD), 2013 IEEE/ACM International Conference on*, pp. 630–637, Nov 2013.

[45] E. G. Cota, P. Mantovani, G. D. Guglielmo, and L. P. Carloni, "An analysis of accelerator coupling in heterogeneous architectures," in *Design Automation Conference*, June 2015.

[46] "Power8 coherent accelerator processor interface." http://www-304.ibm.com/webapp/set2/sas/f/capi/home.html. Accessed: August 14, 2015.

[47] "Intel quickassist technology." http://www.intel.com/content/www/us/en/embedded/technology/quickassist/overview.html. Accessed: August 14, 2015.

[48] L. Daoud, D. Zydek, and H. Selvaraj, "A survey of high level synthesis languages, tools, and compilers for reconfigurable high performance

computing," in *Advances in Systems Science* (J. Switek, A. Grzech, P. Switek, and J. M. Tomczak, eds.), vol. 240 of *Advances in Intelligent Systems and Computing*, pp. 483–492, Springer International Publishing, 2014.

[49] Y. S. Shao, B. Reagen, G.-Y. Wei, and D. Brooks, "Aladdin: A pre-rtl, power-performance accelerator simulator enabling large design space exploration of customized architectures," *SIGARCH Comput. Archit. News*, vol. 42, pp. 97–108, June 2014.

[50] "Graph computing and linked big data." http://ieee-icsc.org/icsc2014/GraphComputing_IBM_Lin.pdf. Accessed: August 14, 2015.

[51] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: A system for large-scale graph processing," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, (New York, NY, USA), pp. 135–146, ACM, 2010.

[52] M. S. Lam, S. Guo, and J. Seo, "Socialite: Datalog extensions for efficient social network analysis," in *Proceedings of the 2013 IEEE International Conference on Data Engineering (ICDE 2013)*, ICDE '13, (Washington, DC, USA), pp. 278–289, IEEE Computer Society, 2013.

[53] "The combinatorial blas: Design, implementation, and applications." http://gauss.cs.ucsb.edu/ aydin/combblas-r2.pdf. Accessed: August 14, 2015.

[54] K. Pingali, D. Nguyen, M. Kulkarni, M. Burtscher, M. A. Hassaan, R. Kaleem, T.-H. Lee, A. Lenharth, R. Manevich, M. Méndez-Lojo, D. Prountzos, and X. Sui, "The tao of parallelism in algorithms," *SIGPLAN Not.*, vol. 46, pp. 12–25, June 2011.

[55] A. Arasu, J. Novak, J. Tomlin, and J. Tomlin, "Pagerank computation and the structure of the web: Experiments and algorithms," 2002.

[56] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein, "Distributed graphlab: A framework for machine learning and data mining in the cloud," *Proc. VLDB Endow.*, vol. 5, pp. 716–727, Apr. 2012.