Heterogeneous Computing Meets Near-Memory Acceleration and High-Level Synthesis in the Post-Moore Era

Nam Sung Kim[1], Deming Chen[1], Jinjun Xiong[2], and Wen-mei Hwu[1]

[1]University of Illinois, Urbana-Champaign, IL, USA

[2]IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA

{nskim, dchen, w-hwu}@illinois.edu; jinjun@us.ibm.com

Abstract

As the trends driven by Moore's Law come to an end, increased heterogeneity at all levels of computing is required to deliver computing performance needed for emerging applications, leading to the proliferation of various application- or domain-specific accelerators. This in turn demands more memory bandwidth, as heterogeneous computing with accelerators consumes data at a much higher rate than traditional homogeneous computing, limiting the computing performance. To tackle this challenge, this article presents a conceptual near-memory acceleration architecture; demonstrates its practicality and plausibility using a recent experimental platform from IBM, as well as its potential impact on performance and energy efficiency; and discusses the need for adopting a high-level synthesis approach for such a near-memory acceleration architecture. Subsequently, this article concludes with future research directions for broad adoption of near-memory acceleration.

Keywords: heterogeneous computing, near-DRAM acceleration, DRAM, FPGA, high-level synthesis, compiler technology.

1. Introduction

Advances in computing have significantly relied on Moore's Law that kept the promise of providing improved performance, energy efficiency, and cost objectives with the ever-increasing density of transistors. Thanks to Moore's Law, the traditional computing system designs primarily depended on homogeneous computing resources (or general-purpose processors) to manage the complexity and manufacturing cost and to facilitate programming productivity and portability. Homogeneous computing in the post-Moore era will, however, not be able to deliver the performance demanded by emerging applications because of their inherent inefficiency (i.e., executing diverse applications with different characteristics using homogeneous general-purpose processors) and stringent power and thermal constraints imposed by the fabrication technologies.

Faced with such a challenge, heterogeneous computing with accelerators such as GPU (Graphics Processing Unit), FPGA (Field Programmable Gate Array), and ASIC (Application-Specific Integrated Circuits), optimized for a domain of applications or a specific application has emerged and proliferated to deliver desired performance for current and future computing, and it has proven to be very effective in dramatically improving performance of diverse applications through many academic studies and industry demonstrations. This in turn demands more memory bandwidth than traditional homogeneous computing, because such accelerators consume data at a much higher rate proportional to their increased processing capability. However, the memory bandwidth based on the traditional processor-memory interface technology will practically stop scaling in the foreseeable future of the post-Moore era, thus limiting the performance of heterogeneous computing with accelerators.

To further improve computing performance under the memory bandwidth constraint, researchers have proposed various near-memory acceleration architectures, exploiting emerging integration technologies to provide more bandwidth for accelerators near memory. However, the high cost and some limitations of such integration technologies can prevent broad adoption of near-memory acceleration architectures especially for low-cost commodity computing systems.

This article first presents a conceptual near-memory acceleration architecture, as well as its potential for improving performance and energy efficiency. The primary benefit of the proposed near-memory acceleration architecture is that it is less expensive and more practical than other near-memory acceleration architectures relying on the emerging integration technologies, because it can be built with commodity memory devices and modules. Second, this article demonstrates the practicality and plausibility of the presented architecture using a recent experimental platform that is designed to allow the proprietary memory interface of IBM POWER8 systems to support various commodity and specialized memory modules such as DDR DIMMs and NVDIMMs. Third, the article argues for the need for adopting a high-level synthesis (HLS) approach to exploit such a platform, as well as demonstrates the performance and energy efficiency improvement obtained by using the HLS approach. Lastly, this article concludes by identifying future research directions to broaden the adoption of such an architecture for more diverse applications especially by reducing the complexity and effort associated with identifying application code segments and mapping them to accelerators.

2. Background

Near-DRAM Acceleration

Among the various memory technologies, DRAM maintains the best balance in capacity, bandwidth, and cost and thus remains as the most dominant and competitive technology for the main memory of general-purpose processors and accelerators. Meanwhile, high-valued applications increasingly demand higher DRAM bandwidth when they run on heterogeneous computing systems because accelerators consume data much faster than conventional processors. However, the bandwidth of the conventional DRAM technology has not significantly increased due to various technological constraints and super-linear cost increase in order to overcome these constraints. Furthermore, the capacitance of on- and off-chip interconnects has scaled at a much slower rate than that of logic with technology scaling. This significantly increases the fraction of data transfer energy in the total system energy, and thus fundamentally limits the energy efficiency obtained by accelerators. These aspects have motivated researchers to leverage emerging DRAM modules such as HMC (Hybrid Memory Cube) and HBM (High-Bandwidth Memory) that facilitate tight 3D-integration of DRAM with accelerators and explore various near-DRAM acceleration (NDA) architectures to reap the performance and energy-efficiency benefits of both accelerators and processing near DRAM (e.g., [1]).

High-Level Synthesis (HLS)

HLS can transform application code based on high-level languages such as C and C++ to RTL (Register Transfer-Level) code automatically. Part of the HLS flow also optimizes the generation of loop and tiling structures, function interfaces, pipelining and in-lining, and various resource instantiations. Figure 1 illustrates some typical steps for HLS. In general, HLS provides effective design complexity management. For example, the code base for HLS designs is 5-10× smaller than that for equivalent RTL designs, and HLS simulations can be up to 1000× faster than equivalent RTL simulations [2]. Thus, HLS is able to explore a large design space efficiently and enables fast iteration on complex designs.

C is the traditional input language for HLS. Other high-level language programs can be translated into C for input to HLS. The FCUDA framework is such an example, where CUDA code is first transformed into C code through source-to-source transformations [3] before given to an HLS tool. Today, any language that can be compiled to a Control Data Flow Graph (CDFG), e.g., C++, OpenCL, MATLAB, can be translated to a hardware description through HLS. A key enabler of this language flexibility is advances

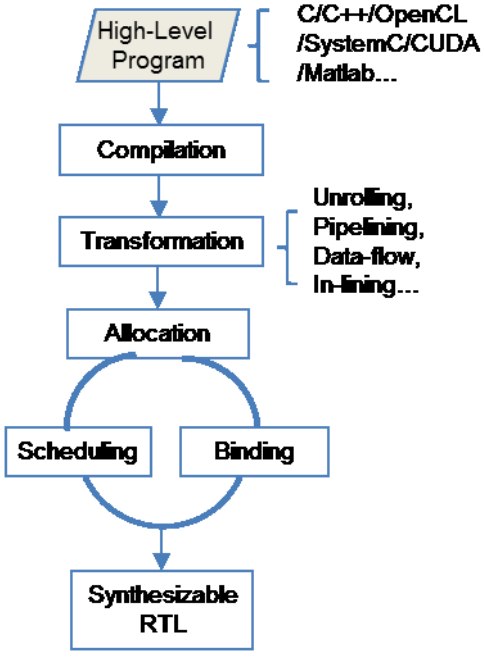

Figure 1: Typical HLS flow.

in common software intermediate representations such as LLVM-IR [4].

Previous HLS solutions have limitations, and studies have shown that the design quality of HLS could be inferior to that of a manual RTL design. New techniques developed recently have, however, led to drastically improved HLS solutions [5], which improved not only the traditional design metrics such as circuit performance and energy efficiency but also emerging metrics such as circuit reliability, robustness and security.

3. Near-DRAM Acceleration Architecture

This section presents an NDA architecture that can be built with standard DRAM DIMMs (Dual-Inline Memory Modules) [6]. Subsequently, it describes a recent experimental buffered DRAM module dubbed ConTutto from IBM [7], and shows that ConTutto can serve as a platform for NDA architectures built with standard DRAM DIMMs. The presented NDA architecture may benefit a narrower range of applications and/or provide lower improvement in performance and energy efficiency than NDA architecture based on 3D and 2.5D integration of DRAM with accelerators. Nonetheless, the presented NDA architecture can serve as an intermediate, low-cost solution for a wide adoption of NDA architecture in the future by bridging the significant cost and capacity gap between standard DRAM DIMMs for traditional computing systems and emerging DRAM modules such as HMC and HBM for emerging computing systems.

Accelerators in Memory Buffers

In pursuit of practical and inexpensive NDA architecture, we exploit some unique aspects of standard DDR4 LRDIMM (Load-Reduced DIMM) architecture. That is, a DDR4 LRDIMM is composed of (1) a repeater device, called RCD (Registering Clock Driver), for command/address (C/A) signals from a memory controller to all of its DRAM devices, and (2) DB (Data Buffer) devices for data signals, one per group of vertically aligned standard DDR4 DRAM chips. LRDIMMs are commonly used to provide up to 8× more main memory capacity than UDIMMs[1] (Unbuffered DIMMs) without sacrificing the maximum bandwidth of commodity DDR4 devices.

Especially, it is recognized that DB devices can be an attractive substrate to integrate accelerators near DRAM [6]. Figure 2(a) depicts how an accelerator can be integrated into the traditional buffering circuitry of a DB device, where a pair of (de)multiplexers in a DB device can provide a path between an accelerator (or buffering circuitry) and a DRAM in NDA (or normal) mode. Figure 2(a) also illustrates how the limitations of DB device as a substrate for integrating an accelerator can be overcome. That is, the DQ[2] I/O circuitry of commodity DRAM devices needs to be slightly modified such that bidirectional DQ pins not only transfer data between a DB device and its coupled DRAM devices but also dispatch C/A pairs from the DB device to the coupled DRAM devices in NDA mode. This small change can be easily accommodated since it does not require us to change either the existing DQ pins anord their layout of packages or DRAM bank architectures[3].

This NDA architecture can improve performance and energy efficiency of memory-intensive applications with limited temporal locality of data but simple operations repetitively applied in parallel to different data. To demonstrate the improvement in performance and energy efficiency, we take a set of benchmarks from widely used benchmark suites such as San Diego Vision, Parboil, CORAL, SPLASH-2 and Rodinia to evaluate the performance and energy consumption of the presented NDA architecture in Figure 2(b). These benchmarks exhibit high parallelism and they can distribute their input sets across accelerators to exploit concurrency and localize most of memory accesses. This in turn reduces inter-accelerator communications, providing high performance and energy efficiency.

Figure 2(b) plots the simulated performance and energy consumption of a computer system with this NDA architecture, normalized to a computer system based on an Intel Haswell-like processor as a baseline. Although any programmable compute units such as SIMD (Single-Instruction Multiple-Data) engines, GPU cores and FPGA devices can be accelerators for this NDA architecture [6], a CGRA (Coarse-Grained Reconfigurable Accelerator) is

---

[1] UDIMMs require a trade-off between capacity and bandwidth (e.g., 1600, 1866, and 2133MT/s for 3, 2, and 1 DIMMs per channel) due to signal integrity challenges.

[2] DQ denotes data in DRAM.

[3] Sharing DQ pins with C/A pins was implemented in DDR2 DRAM for Fully Buffered DIMM.

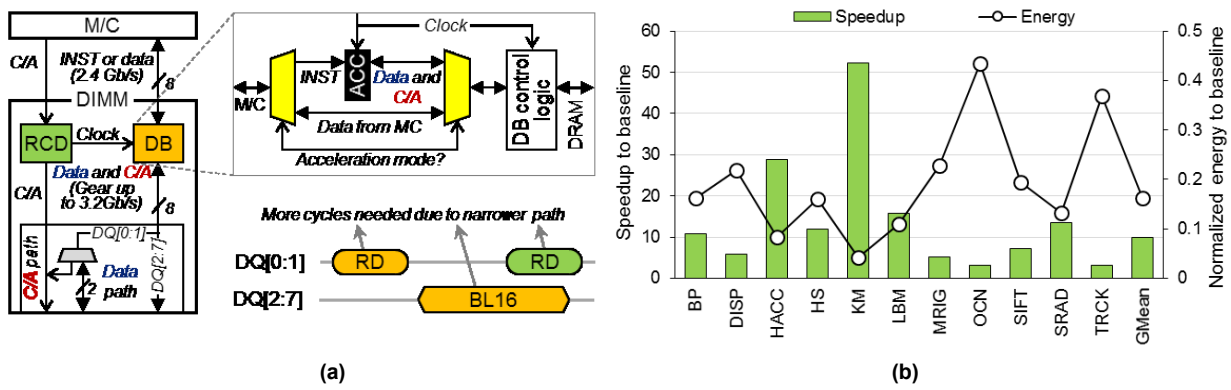

Figure 2: (a) Near-DRAM accelerator architecture where the accelerators are integrated within buffer devices. (b) Simulated performance and energy of near-DRAM accelerator illustrated in (a). BP, DISP, HACC, HS, KM, LBM, MGRIG, OCN, SIFT SRAD, and TRCK denote Back Propagation, DISParity map, Hardware ACCelerated cosmology code, HotSpot, K-Means, Lattice-Bolzmann Method fluid dynamics, Magnetic Resonance Imaging Gridding, OCeaN movements, Scale-Invariant Feature Transform, Speckle Reducing Anisotropic Diffusion, and feature TRaCKing.

chosen because it can provide lower control overhead and higher energy efficiency than SIMD engines and GPU cores, and better programmability than FPGA devices in this experiment. In the later part of this article, we will consider FPGA-based accelerators as well since the programmability challenge in using FPGA devices can be significantly mitigated by using the HLS approach.

Performance evaluation is performed by *gem5* integrated with an in-house CGRA simulator. To model the memory sub-system, we take DDR4-2400 with tRC of 45.32ns and tRCD of 13.32ns. To evaluate the energy consumption of processors and DDR4-2400 DRAM, we use McPAT and DRAMPower [8]. To model the timing, energy consumption and area of a CGRA with 64 functional units, we develop Verilog models that are synthesized with Synopsys DesignWare IPs, a TSMC 40nm standard cell library, and Synopsys Design Compiler to operate at 800MHz. The synthesized CGRA consumes 0.832 mm^2 with power density of 4 – 21mW/mm^2, demonstrating that a CGRA can be integrated into a package for a DB device. The results show that this NDA architecture can offer 9.89× higher performance and 83% less energy than the baseline.

Note that this NDA architecture can simply replace conventional DB devices with accelerator-integrated DB devices on standard LRDIMMs to enable NDA. As such, it is significantly cheaper than NDA architecture relying on 3D or 2.5D-stacking accelerators atop HBM/HMC-like memory at the moment[4].

Performance Scaling with More NDA-enabled Memory Modules

As this NDA architecture is built with commodity DRAM devices and standard LRDIMMs, one critical question arises regarding how to provide sufficiently high DRAM bandwidth for the accelerators. To tackle this challenge, the following observations can be exploited.

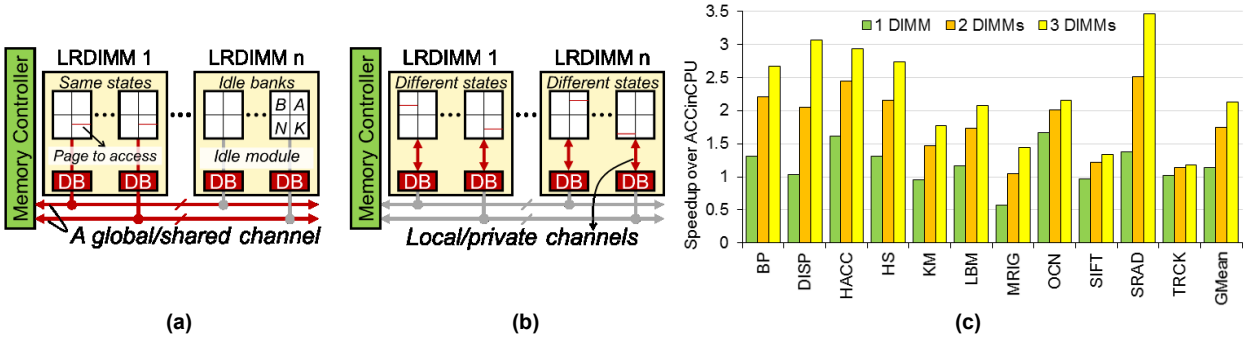

Figure 3: Bandwidth exposed to accelerators in (a) the host processor and (b) buffer devices. (b) Performance improvement with more NDA-enabled memory modules, relative to accelerators in the host processor ("ACCin CPU").

[4] This statement is based on conversations with major DRAM manufacturers.

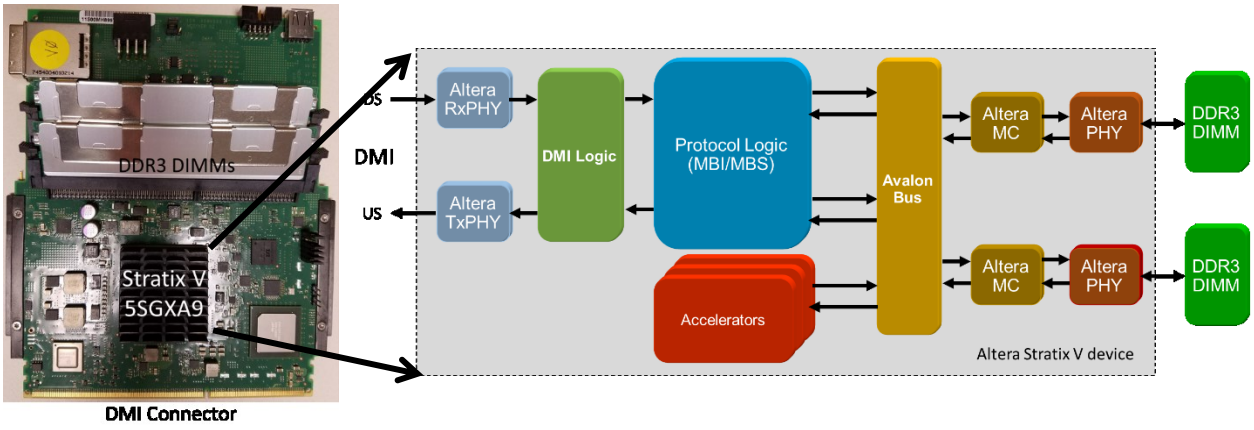

Figure 4: (a) IBM ConTutto prototype designed to be plugged into a main-memory slot of a IBM POWER8 system; "DMI" is the IBM proprietary memory interface equivalent to DDR used for x86-based systems. **(b) Hardware components mapped onto the FPGA device of an IBM ConTutto board.**

Increasing the number of memory modules per channel (or controller) does not increase the bandwidth between the host processor (and accelerators in the processor) and the memory modules, because the channel is shared by all the memory modules in a traditional main-memory system illustrated in Figure 3(a). In the presented NDA architecture, however, accelerators in each memory module do not use the shared channel but the private channels between memory buffer devices and their coupled memory devices. Hence, the accelerators are isolated from the shared channel by the buffer devices in NDA mode [6,9]. Consequently, the aggregate bandwidth that can be utilized by the accelerators is multiplied by the number of NDA-enabled memory modules per channel as depicted in Figure 3(b). In other words, the bandwidth per accelerator remains constant in this NDA architecture, whereas it would have been divided by the number of accelerators in a traditional architecture that integrates the accelerators within the host processor itself (denoted by "ACCinCPU" in this article).

Industry Proprietary, Experimental Buffered Memory Module as an NDA Platform

Figure 4(a) depicts a ConTutto prototype, a recently announced buffered DRAM module from IBM [7]. It is designed to be plugged into a memory slot of an IBM POWER8 system and allows an IBM POWER8 system to use commodity DDR DIMMs or NVDIMMs instead of DRAM modules based on DMI (Differential Memory Interface), an IBM proprietary memory interface. Specifically, two primary components of a ConTutto board are (1) DDR DRAM DIMMs (or NVDIMMs) and (2) an FPGA device implementing DDR and DMI interfaces with IP blocks such as PHY and DDR memory controllers provided by the FPGA device and some custom logic designed by IBM as illustrated in Figure 4(b). The FPGA device in a ConTutto board is a high-end Stratix V, and a large fraction of (programmable) resources is unused after implementing the primary function as a buffered memory interface between DMI and DDR. Exploiting the unused FPGA resources, we can implement accelerators in the FPGA device. That is, a ConTutto board can be programmed into a platform for NDA architectures similar to the presented NDA architecture in this section, demonstrating the practicality and plausibility of the presented NDA architecture. Lastly, as multiple ConTutto boards can be plugged into the memory slots of an IBM POWER8 system and each ConTutto board provides two DDR DRAM channels (i.e., doubling the DRAM bandwidth per channel), we can also scale the aggregate DRAM bandwidth exposed to the accelerators in ConTutto boards, as described earlier in this section.

4. A High-Level Synthesis Approach for FPGA-based NDA Platforms

The FPGA device in an NDA platform as illustrated in the previous section can serve as a programmable accelerator for diverse applications. It has been reported that FPGA-based acceleration can provide orders of magnitude of improvement in performance and energy efficiency when compared with execution based on general-purpose processors. Nonetheless, it is non-trivial to utilize FPGA as an accelerator, because application code segments, which need to be accelerated in FPGA, must be designed in hardware (i.e., synthesizable HDL code at the RTL). In contrast, traditional programmable general-purpose accelerators such as CGRA or GPU do not impose such a

burden as they execute the application code segments in software albeit transformed. Faced with such a challenge especially in FPGA-based NDA, we propose to leverage recent advances in HLS to program the FPGA device. Moreover, adopting HLS for NDA will allow us to rapidly (re)program the FPGA-based NDA in datacenter (cloud) server environments where we need to satisfy demands for accelerating various applications from many users.

Recent rapid advances in machine-learning algorithms such as DNN (deep neural network) have enabled countless emerging recognition applications that are particularly suitable for cloud deployment. Since such machine-learning algorithms are memory-intensive, the performance of running them on traditional general-purpose processors with their memory systems is often limited by the available main-memory bandwidth [10]. Therefore, such applications are excellent candidates for NDA. In this section, we take an HLS approach to efficiently implement a convolutional neural network (CNN)[5] on the FPGA device. More specifically, we first restructure the original CNN C++ source code to make it synthesizable, meaning that it does not contain code that cannot be translated into customized hardware. We then apply various techniques such as data sharing, data-flow optimization, batch processing, layer merging and combination, and task-level pipelining together with the right set of compiler pragmas to guide the compilation of Xilinx Vivado HLS. Then, we generate the corresponding RTL code, and map it onto a Xilinx FPGA device with two DDR3 DRAM channels (Xilinx ZYNQ-7000 SOC ZC706). To demonstrate the potential impact of this HLS approach on performance, power, and energy improvements for FPGA-based NDA, we also run the original CNN code in C++ on a computer with an Intel Xeon E3-1240 processor operating at the maximum frequency of 3.4GHz and 24GB RAM as a baseline. We estimate the power consumption of accelerators implemented in the FPGA device using power simulation tools from the Xilinx design suite and measure that of a computer based on the Intel processor using a power meter. Our experiment shows that the FPGA-based acceleration of CNN provides 268× improvement in performance while consuming only 10% power with negligible energy compared with the baseline.

	Intel Xeon	FPGA with HLS	FPGA/Xeon
Performance	0.525 GOPs	140.6 GOPs	268×
Power	95W	9.6W	10.1%
Energy	1168.5J	0.44J	0.038%

Table 1: Performance, power and energy of FPGA-based acceleration using HLS compared to an Intel Xeon-based system.

5. Future Directions to Improve Performance and Programming Portability of NDA

Performance Portability

In accelerating diverse applications with a NDA platform, even adopting an HLS approach requires a substantial level of programmer's effort to identify and optimize application code segments to be accelerated before the synthesizable RTL code is mapped to an FPGA device. Albeit demanding less effort than FPGA-based acceleration, CGRA- or GPU-based acceleration must go through similar steps by matching the application code segments to the parallelism and memory architecture of the accelerator. Otherwise, high performance improvement cannot be accomplished regardless of the type of accelerators deployed especially for NDA. This challenge demands NDA to adopt a practical program synthesis system such as Tangram [11], which takes architecture-neutral program expression and automatically performs accelerator and memory architecture-specific optimization/tuning for performance, and adapt it for NDA with some enhancements. Specifically, we envision that a new compiler flow that can map Tangram code fragments to FPGA can naturally support NDA targeting IBM POWER8 system with ConTutto boards.

Programming Portability

[5] A CNN design typically consists of a number of convolution layer groups and some fully-connected layers. Each convolution layer group often contains one convolution layer, one rectified linear unit (ReLU) layer and one max-pooling layer. These groups are instantiated as computation modules. Apart from computation module, the CNN design also contains memory module, which stores the weight and input data.

The use of traditional accelerators connected with system interconnects such as PCIe (and memory channels for NDA) often requires invasive changes in existing application source codes, because APIs specific to an accelerator architecture need to be incorporated into the original application codes to orchestrate coherent communications between the host processor and accelerators. This can be a considerable barrier to adoption because NDA is yet another acceleration architecture that will require some unique APIs considering those distinctive aspects of NDA. Such APIs and their usage will be different from those for traditional accelerators connected to PCIe, hence requiring a fresh learning curve for programmers, hurting productivity and thus adoptability of NDA. To this end, we expect that emerging interface technologies such as OpenCAPI (Open Coherent Accelerator-Processor Interface) [12] will replace the traditional interface technologies such as DDR and PCIe and be used as a unified interface technology for both accelerators and memories. With such interface technologies, it only requires one time effort to be proficient at using common/unified APIs for such interface technology, and it will then allow programmers to exploit the performance improvement by any future accelerator architecture including NDA architecture.

6. Conclusions

Faced with various challenges to improve performance and energy-efficiency of computing systems in the post-Moore era, this article presented a conceptual near-DRAM acceleration (NDA) architecture, a heterogeneous computing architecture to further improve the acceleration performance that is currently limited by the available memory bandwidth between accelerators and memory. This article demonstrated ~10× and ~5× improvement in performance and energy efficiency, respectively, when compared with a conventional computer system based on a high-performance general-purpose processor. Furthermore, this article demonstrated the practicality and plausibility of the NDA architecture with a recent experimental IBM platform, as well as 268× improvement in performance using a fast and convenient HLS approach for FPGA-based acceleration. Lastly, this article discussed future directions to improve performance and programming portability of NDA with an emerging program synthesis system and accelerator-processor interface technology.

References

[1] M Gao and C. Kozyrakis, “HRL: Efficient and Flexible Reconfigurable Logic for Near-Data Processing,” IEEE International Symposium on High-Performance Computer Architecture (HPCA), 2016.

[2] K. Wakabayashi, “C-based Behavioral Synthesis and Verification Analysis on Industrial Design Examples,” IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC), 2004.

[3] K. Papakonstantinou, et al, “FCUDA: Enabling Efficient Compilation of CUDA Kernels onto FPGAs,” International Symposium on Application Specific Processors (SASP), 2009.

[4] C. Lattner and V. Adve, “LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation,” IEEE International Symposium on Code Generation and Optimization (CGO), 2004.

[5] K. Campbell, W. Zuo, and D. Chen, “New Advances of High-Level Synthesis for Efficient and Reliable Hardware Design”, Integration, the VLSI Journal. To appear.

[6] H. Asghari-Moghaddam, et al., “Chameleon: Versatile and Practical Near-DRAM Acceleration Architecture for Large Memory Systems,” IEEE/ACM International Symposium on Microarchitecture (MICRO), 2016.

[7] Thomas Roewer, “ConTutto – A flexible memory interface in the OpenPOWER ecosystem,” [Online]. Available: https://openpowerfoundation.org/wp-content/uploads/2016/11/Thomas-Roewer-ConTutto-A-flexible-memory-interface-in-the-OpenPOWER-ecosystem.pdf

[8] K. Chandrasekar et al., “DRAMPower: Open-source DRAM power & energy estimation tool.” [Online]. Available: http://www.drampower.info

[9] S. H. Pugsley, et al., “Comparing Implementations of Near Data Computing with In-Memory MapReduce Workloads,” IEEE Micro, 34(4), 2014.

[10] Chen Zhang and et. al. “Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks,” in IEEE International Symposium on Field-Programmable Gate Arrays (FPGA), 2015.

[11] L.-W. Chang, et al, “Efficient Kernel Synthesis for Performance Portable Programming,” in ACM/IEEE International Symposium on Microarchitecture (MICRO), 2016.

[12] “OpenCAPI,” [Online] Available: http://opencapi.org

Acknowledgement

This work is supported in part by IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR) - a research collaboration as part of the IBM Cognitive Horizon Network, and grants from Samsung, DARPA and NSF.

Nam Sung Kim (Fellow `16) is an Associate Professor in Electrical and Computer Engineering of the University of Illinois at Urbana-Champaign. His interdisciplinary research incorporates device, circuit, architecture, and software for power-efficient computing. He is a recipient of the IEEE Design Automation Conference (DAC) Student Design Contest Award in 2001, Intel Fellowship in 2002, IEEE International Symposium on Microarchitecture (MICRO) Best Paper Award in 2003, NSF CAREER Award in 2010, IBM Faculty Award in 2011 and 2012, and University of Wisconsin Villas Associates Award in 2015. He is a member of IEEE International Symposium on High-Performance Computer Architecture (HPCA) Hall of Fame and IEEE/ACM International Symposium on Microarchitecture (MICRO) Hall of Fame. He received his Ph.D. degree from the University of Michigan, Ann Arbor.

Deming Chen is a Professor in Electrical and Computer Engineering of the University of Illinois at Urbana-Champaign. He obtained his BS in computer science from University of Pittsburgh, Pennsylvania in 1995, and his MS and PhD in computer science from University of California at Los Angeles in 2001 and 2005 respectively. His current research interests include system-level and high-level synthesis, computational genomics, GPU and reconfigurable computing, and hardware security. He received the Arnold O. Beckman Research Award from UIUC in 2007, the NSF CAREER Award in 2008, and six Best Paper Awards. He also received the ACM SIGDA Outstanding New Faculty Award in 2010, and IBM Faculty Award in 2014 and 2015. He is a senior member of IEEE and the Donald Biggar Willett Faculty Scholar of College of Engineering.

Jinjun Xiong is a Program Director for Cognitive Computing Systems Research at IBM Thomas J. Watson Research Center. Prior to that, he was a Manager responsible for IBM Research’s Big Bet Program on Smarter Energy Research. Xiong pioneered the statistical data-driven methodology for improving VLSI chip yields, and developed techniques that were used by IBM's high performance ASIC and server designs. For his contribution, Xiong was recognized as an IBM Master Inventor, and a recipient of various IBM technical achievement awards and the IEEE Region One Outstanding Technical Contribution Award. Xiong received his Ph.D. degree in Electrical Engineering from the University of California, Los Angeles in 2006.

Wen-mei W. Hwu (Fellow ’98) is a Professor and holds the Walter J. (‘‘Jerry’’) Sanders III-Advanced Micro Devices Endowed Chair in Electrical and Computer Engineering of the University of Illinois at Urbana-Champaign. His research interests are in the area of architecture, algorithms, and programming models for parallel computer systems. He directs the IMPACT research group (www.crhc.uiuc.edu/Impact). For his contributions, he received the ACM SigArch Maurice Wilkes Award, the ACM Grace Murray Hopper Award, the IEEE Computer Society Charles Babbage Award, the IEEE Computer Society B. R. Rau Award, and the ISCA Most Influential Paper Award. He is a fellow of IEEE and ACM. Hwu received his Ph.D. degree in Computer Science from the University of California, Berkeley in 1987.