Energy-Efficient Hardware Acceleration through Computing in the Memory

Somnath Paul
Intel Corporation
Hillsboro, OR, USA
Email: somnath.paul@intel.com

Robert Karam
Case Western Reserve Univ.
Cleveland, OH, USA
Email: robert.karam@case.edu

Swarup Bhunia
Case Western Reserve Univ.
Cleveland, OH, USA
Email: skb21@case.edu

Ruchir Puri
IBM Watson Research Center
Yorktown Heights, NY, USA
ruchir@us.ibm.com

***Abstract*—Energy-efficiency has emerged as a major barrier to performance scalability for modern processors. We note that significant part of processor's energy requirement is contributed by processor-memory communication. To address the energy issue in processors, we propose a novel hardware accelerator framework that transforms high-density memory array into a configurable computing resource to accelerate variety of tasks - both compute- and data-intensive. It exploits the block-based architecture of nanoscale memory to create a spatially connected array of lightweight processors, each of which uses a memory block as its local memory. The proposed framework provides some unique advantages for hardware acceleration compared to conventional accelerators: 1) memory array provides large set of parallel resources with high bandwidth, which can be configured to perform computing in spatio/temporal manner leading to dramatic reduction in processor-memory traffic; 2) it brings the computing engine close to the data, thus drastically minimizing the von Neumann bottleneck; 3) finally, it exploits the advances in memory technologies and integration approaches e.g. 3D integration to achieve better technology scalability compared to alternative reconfigurable accelerator platforms. Simulation results for several data-intensive applications show that the proposed computing approach provides significant improvement in energy-efficiency compared to software while achieving significantly lower hardware overhead.**

I. Introduction

In the nanometer technology regime, power has emerged as the primary design constraint. Since technology scaling no longer provides cubic reduction in energy following *Dennard's scaling rules* [2], attention has shifted to alternative approaches to improve energy efficiency, namely, new algorithms, highly-optimized accelerators, smarter hardware-software partitioning and sophisticated power management techniques [1]. The demand for improved energy efficiency is imposed by applications spanning diverse areas such as scientific computations, web serving, multimedia storage, searching and characterization. Many of these applications are data-intensive [2], primarily bound by off-chip input/output (IO) performance and energy requirements. In the past decade, integrated graphics processing units (GPU) and other on-chip accelerators have largely addressed the compute energy for these applications and embedded cache hierarchy has partially alleviated their IO bottleneck.

However, as technology scales, scalability in energy-efficiency remains a serious challenge for such applications. There are primarily two major challenges. The first challenge arises from the fact that the power profile for the transistor is not scaling as well as its integration density. It is therefore unlikely that future energy-constrained over-provisioned systems (often referred to as "dark silicon" [3]) will be able to meet the performance requirements for these emerging data-intensive applications. The second challenge is posed by the ever-increasing gap between on-chip memory and processor frequencies and external data rates (refer to Fig. 1). Although large embedded caches have addressed the IO bottleneck in the past, they have already hit limits in hiding the off-chip access latency. In this scenario, 3D integration has emerged as an effective approach to minimize the off-chip communication cost. In such integrated systems, stacked monolithically integrated dies communicate between themselves using through-silicon vias (TSV). The result of this integration is improved power consumption for dynamic random-access memory (DRAM). However, as pointed out in [3], this technology will limit the capacity of DRAM die per stack since each stack will need to provide their own I/O pins. In addition, heat dissipation and thermal-induced reliability concerns remain a major challenge for 3-D IC design. In summary, increasing requirement of back-and-forth data transfer between processor and memory, referred to as *Von Neumann bottleneck*, has emerged not only as a performance bottleneck but also as a limiter to energy-scaling.

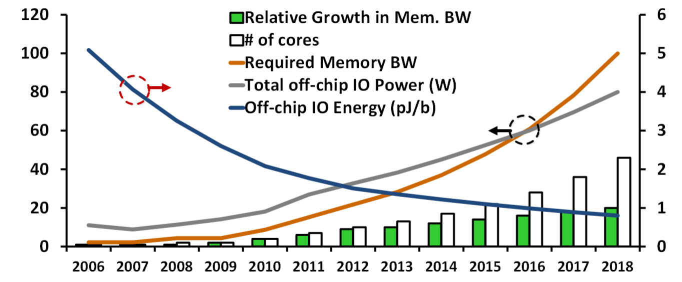

Fig. 1. Scaling trends for off-chip bandwidth and power suggest that a large gap exists between the technology projections and system requirements [4].

Contrary to the grim challenge of energy scaling faced by conventional computing architectures, the field of non-volatile memory research has seen unprecedented developments in the past decade. While NAND Flash trends for increased integration density remains on track, alternate non-CMOS non-volatile memories have emerged and are now even being commercialized. Some of these include phase change memory (PCM) [5], resistive random-access memory (ReRAM) [6] and

978-3-9815370-2-4/DATE14/©2014 EDAA

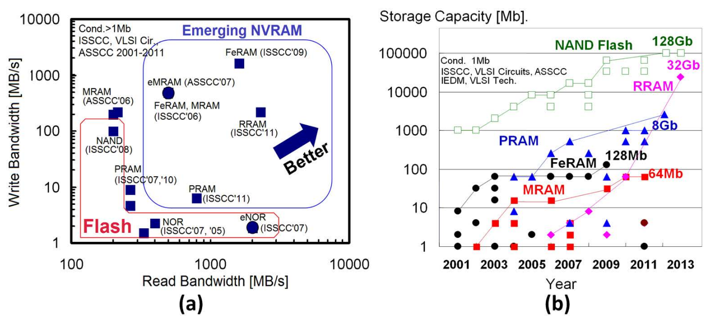
Fig. 2. Trends for a) read/write bandwidth and b) storage capacity scaling for NAND Flash and other emerging non-volatile memory technologies [1].

spin torque transfer random-access memory (STTRAM)[7]. As evident in the trends captured in Fig. 2, these non-volatile memories (NVM) achieve higher read/write bandwidth, increased energy-efficiency and improved endurance compared to NAND Flash at the overhead of reduced storage capacity. As these novel NVM technologies improve over time, they are being considered strong candidates for both stand-alone and embedded applications. The feasibility of integrating permanent storage with computing technologies will undoubtedly bring about a major paradigm shift. As the distinction between compute and storage is blurred, the prospect of computing in memory appears realistic.

The concept of in-memory computing has been proposed in numerous earlier works [8, 9] in the context of computing in the main memory. However, with the emergence of fast and reliable non-volatile storage, recent in-memory compute architectures have targeted these NVM technologies [10]. The benefits of such architectures is particularly noteworthy for pattern recognition, data mining and such analytic applications which traditionally involve moving a large volume of data through the memory hierarchy. In this work we review the previous approaches towards in-memory computing and describe a memory-based *MAlleable Hardware Accelerator* (MAHA) that implements a reconfigurable computing fabric with NAND Flash memory to map data-intensive applications. This novel spatio-temporal framework is implemented by *instrumenting* the last-level non-volatile storage and therefore has minimal impact to its integration density. In particular, the paper has the following major contributions:

1. It presents MAHA, a truly in-NVM reconfigurable framework. It describes in detail the hardware and software implementation of MAHA, which is realized through design changes to non-volatile memory array organization. Modifications for transforming the memory array into a reconfigurable computing resource are critically examined.

2. It presents an emulation framework which validates the functional correctness of the MAHA framework. Results from the emulation setup suggests the significant energy gains that can be achieved through in-memory computing.

II. Background & Motivation

A. Related Work

Overcoming the memory wall has been a major emphasis of alternate architectures that have gained popularity in recent years. For these architectures (example GPU), many-threaded

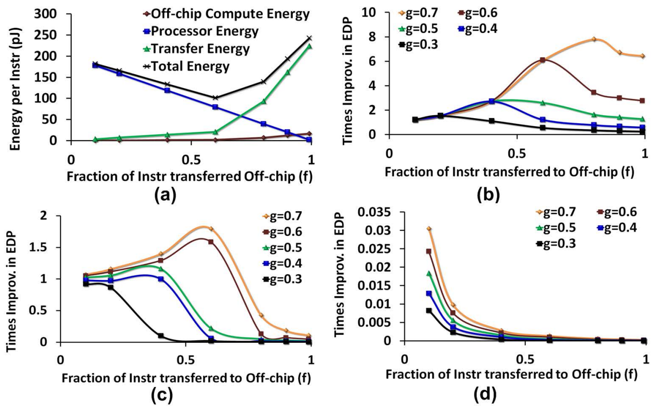
Fig. 3. Energy and performance for a hybrid system with a host processor and off-chip memory based hardware accelerator. a) Effective energy per operation in the hybrid system with *c=o=0.05* and *g=0.7*. Improvement in energy-efficiency (EDP) for the hybrid system with: b) *c=o=0.005*; c) *c=o=0.05*; d) *c=o=0.5*.

execution is a common feature addressing the main memory latency and bandwidth bottleneck. Although these architectures address the performance bottleneck to a large extent, the energy requirement for off-chip access is still taxing. The Intelligent RAM (IRAM) [8] and Processing-in-Memory (PIM) [9] projects have been forerunners for in-memory architectures which attempt to alleviate this energy overhead. The idea is to reduce the latency and energy of DRAM access by placing processor or vector processing unit on the same chip with DRAM. Emerging NVM technologies however hold the promise of low-access times and fine-grained byte-level access, as opposed to high-access latency for fixed-sized blocks with thousands of bytes for today's Flash technology. The proposed MAHA framework is therefore designed to closely integrate with emerging non-volatile storage and thus potentially offers higher energy savings for data-intensive applications.

B. Motivation

Key application and system primitives can be leveraged to determine whether an application will benefit from in-memory acceleration.

A. Key Primitives: In order to compare between a software-only solution and a hybrid system with off-chip in-memory accelerator, we express the application characteristics and the system configuration using a set of primitives as listed below:

1. g - fraction of total instructions with memory reference (loads and stores).

2. f - fraction of total instructions transferred to an off-chip compute engine.

3. c - fraction of instructions translated from host's ISA to the ISA for the off-chip compute framework. Note that loads and stores can be partially removed during such a translation.

4. o - fraction of original instructions which result in an output. A fraction $f \times c \times o$ thus produces outputs which needs to be transferred to the host processor.

5. $e_{offchip}$ - average energy per instruction in the off-chip compute engine.

6. e_{txfer} - energy expended in the transfer of an output from the off-chip framework to the host processor.

7. $t_{offchip}$ - ratio of cycle time for the off-chip compute framework to that for the host processor.

8. n - fraction speedup due to parallelism in the framework

TABLE I. TYPICAL VALUES OF PERFORMANCE AND ENERGY IN AN OFF-CHIP COMPUTE FRAMEWORK

$e_{offchip}$	50pJ (higher energy per instr. considering local memory load and store)
e_{txfer}	10,000pJ (same order of magnitude as the energy for an off-chip non-volatile memory access)
$t_{offchip}$	15 (typical processor (@ 1GHz) to off-chip memory cycle times)
n	0.01 (with the assumption that large # of compute engines can be accommodated into large last level memory)
t_{txfer}	10,000 (same order of magnitude as latency for an off-chip non-volatile memory access)

(ratio of on-chip to off-chip compute engine count).
9. t_{txfer} - time taken in terms of processor clock cycles to transfer an output from the off-chip compute framework to the host processor.
With these primitives, the average time to execute an instruction in a system with a host processor and the off-chip compute framework can be formulated as:

$$T_{sys} = T_{offchip} + T_{proc} + T_{txfer} \tag{1}$$
$$where$$
$$T_{offchip} = t_{offchip} \times (I_{f>g}(f) \times (f - g + f \times c \times n) + I_{f\leq g}(f) \times f \times c \times n)$$
$$T_{proc} = (1 - f) \times t_{proc}$$
$$T_{txfer} = t_{txfer} \times (I_{f>g}(f) \times ((f - g) \times o + f \times c \times o) + I_{f\leq g}(f) \times f \times c \times o)$$
$$where$$
$$I_A(x) = \begin{cases} x \; if \; x \in A \\ 0 \; if \; x \notin A \end{cases}$$

In equation 1, $T_{offchip}, T_{proc}$ and T_{txfer} denote the fraction latencies in the off-chip compute framework, due to processor execution and in the transfer of the resultant output from the off-chip platform to the processor. A similar expression for the energy of the resultant system is given below which shows the transfer energy increasing and the processor energy decreasing with increasing f:

$$E_{sys} = E_{offchip} + E_{proc} + E_{txfer} \tag{2}$$
$$where$$
$$E_{offchip} = e_{offchip} \times (I_{f>g}(f) \times (f - g + f \times c) + I_{f\leq g}(f) \times f \times c)$$
$$E_{proc} = (1 - f) \times e_{proc}$$
$$E_{txfer} = e_{txfer} \times (I_{f>g}(f) \times ((f - g) \times o + f \times c \times o) + I_{f\leq g}(f) \times f \times c \times o)$$

With performance and energy values typical to the operation of an off-chip compute framework (modeled after the *MAHA* platform as listed in Table I), we estimate the system-level improvement in energy-efficiency for applications with varying values of the primitives listed earlier. Fig. 3(a) shows the three components and the total system energy with *g=0.7* and *c=o=0.05*, respectively. As clearly evident from Fig. 3(a), for the values of *c* and *o* selected, minimum energy consumption is achieved for values of *f* close to *g*. For values of *c and o* order of magnitude large or small, the total energy was found to always increase or decrease with *f*, respectively, suggesting that small and large *c and o* value always favor or disfavor off-chip acceleration. A similar dependance on *c, o and f* was observed for total execution latency. Combining the performance and energy trends for this system, we derive the EDP trend of the total system (Fig. 3(b-d)). As expected, the EDP improvement progressively diminishes as the data-intensive nature (value of *g*) of the application is reduced. Finally, from Fig. 3(b) and (c) we note that for EDP improvement > 1, for a given *g*, there exists an *f* for which maximum EDP improvement can be observed. Given the nature of an application (*g* and *o* known), knowledge of its mapping to the off-chip framework (*c* and *n* known), it is possible to determine *f* for which maximum energy savings can be obtained by off-chip computing.
B. Desired traits for off-chip acceleration: From the analysis above, we infer that applications with large number of memory references (high value of 'g') and small output data set (low value of 'o') are particularly amenable to improvement in energy-efficiency through off-chip computing.

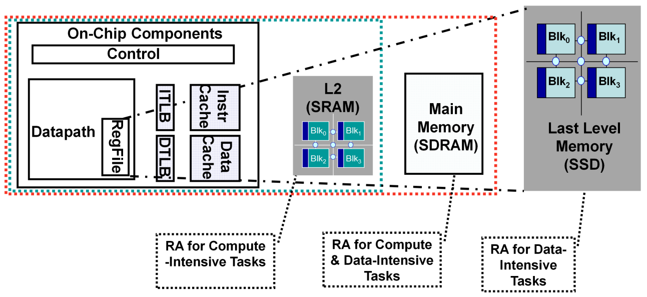

Fig. 4. In-memory acceleration can be employed by integrating reconfigurable arrays (RA) in multiple levels of the memory hierarchy.

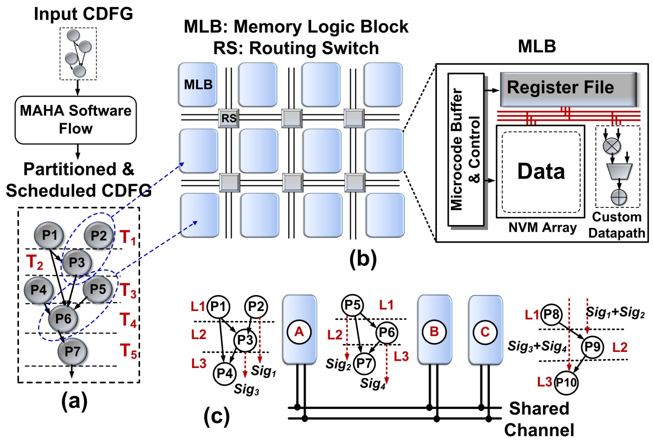

Fig. 5. a) Overview of *MAHA* architecture and application mapping flow; b) μ-arch details of a single computing block (MLB); c) Synchronization among multiple MLBs over shared interconnect.

III. MAHA: COMPUTING IN NON-VOLATILE MEMORY

As illustrated in Fig. 4, the concept of in-memory computing can be applied to each level of the memory hierarchy. Each of these levels bear the common characteristics of i) modular memory design and ii) hierarchical organization of memory blocks. Design-time modifications as proposed here for the MAHA architecture can transform these memory blocks into lightweight processors with each block as its local memory. The trade-offs involved in such modifications would however differ depending on the underlying memory technology.

A. Overview

MAHA is a spatio-temporal mixed-granular hardware reconfigurable framework. It consists of an array of processing

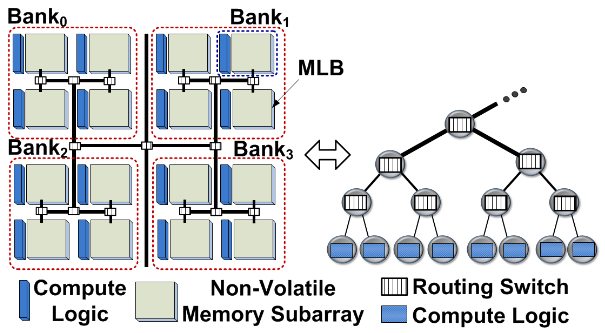

Fig. 6. Conventional hierarchical organization of memory and its modification with additional programmable logic.

elements (PE), communicating using a hierarchical interconnect architecture. The target application to be mapped to *MAHA* is represented as a control and data flow graph (CDFG). The software flow partitions this CDFG into smaller multi-input multi-output tasks and defines the schedule for execution of each task. One or more of these tasks are then mapped to individual PEs. Fig. 5(a) illustrates the application mapping flow for the *MAHA* framework.

1) Compute Logic: Each compute block or PE for the *MAHA* framework is referred to as Memory Logic Block or MLB. The details of a single MLB as illustrated in Fig. 5(b) shows a dense non-volatile memory array which stores data. This is referred to as *function table*. A custom datapath with arithmetic units such as adder and multiplier and permutation unit such as shifter constitute the logic datapath. A local register file is responsible for storing the temporary operands. Sequence of operations inside a MLB is controlled by a μ-code controller referred to as *schedule table*. Tasks mapped to each MLB execute in a topological manner over multiple clock cycles, communicating via local register file.

2) Interconnect Fabric: Tasks mapped to different MLBs communicate via a programmable and hierarchical interconnect. As illustrated in Fig. 5(c), the interconnect is time-multiplexed and shared among multiple MLBs. As shown in Fig. 5(c), Sig_1 and Sig_2 are outputs of MLB A and B at the end of cycle 1, while Sig_3 and Sig_4 are outputs at the end of cycle 2. Signals at the end each cycle are transmitted over the same local/global channel to MLB C. Since an input application is statically scheduled to the *MAHA* hardware, inter-MLB communication is deterministic and obviates the need for complex on-chip networks.

3) NVM Instrumentation: A hardware framework which implements the *MAHA* architecture can be realized through design time modifications (referred to as *instrumentation*) to a hierarchical memory organization which allows the memory to dynamically process during runtime. As illustrated in Fig. 6, on-demand computing requires:
i. Compute Logic: Each memory subarray which sits as a leaf node in a hierarchical memory organization must be augmented with additional compute logic which supports computation on-demand.
ii. Routing Switches: Data movement in a hierarchical interconnect (as an example *Fat tree*) organization can be achieved through *routing switches* present as part of the interconnect (refer to Fig. 6).

4) Key benefits: Key benefits of *MAHA* architecture are:
i) In contrast to large on-chip caches, each MLB incorporates a small local memory which scales better with technology. The data movement inside the MAHA framework is explicitly controlled through software mapping of the target application. This alleviates the requirement for complex control hardware.
ii) *MAHA* is a spatio-temporal framework. In contrast to conventional FPGAs which are fully spatial and are limited by interconnect power and performance, *MAHA* allows the application mapping tool to trade-off distributed vs local execution for maximum energy-efficiency.
iii) *MAHA* is a mixed-granular framework capable of exploiting both data and task parallelism. The application mapping tool receives a high-level application description and efficiently maps it to an underlying hardware framework. In this respect, it is easily programmable like a GPU and flexible like a FPGA.

B. NAND Flash - A Case Study

The benefits of memory-based computing as outlined before has already been validated for the case of on-chip volatile CMOS SRAM and non-volatile STTRAM technologies [10]. In this work, with NAND Flash as a representative non-volatile storage we demonstrate that MAHA can be effective in mitigating the Von-Neumann bottleneck data-intensive applications. Flash memory has seen an astounding increase in integration density, making it attractive for commodity storage systems as well as embedded applications. Recently, 3-D NAND Flash memories have gathered increasing attention as future ultrahigh-density memory technologies [11]. It has the benefits of being a fabrication process that is planar and fully compatible with complementary metaloxide-semiconductor (CMOS) technology. In this work therefore, we describe the off-chip *MAHA* framework based on CMOS-compatible NAND Flash memory array.

1) Overview of current Flash organization: A typical organization of the NAND Flash memory is shown in Fig. 7(a) with Flash memory array and a number of logic structures responsible for controlling the read and write operations to the Flash [12]. The Flash Translation Layer (FTL) converts the logical address of a location to its corresponding physical address. The NAND Flash typically has 8-bit or 16-bit I/O bandwidth. NAND Flash is organized in units of pages and blocks. Typical page size is 2KB [12] and each block can have 64-128 pages. During Flash read, contents of the entire page is first read into the page register and then serially transferred to the Flash external interface. Organization details, read/write performance and energy values for a representative NAND Flash at 45nm are obtained from opensource models [12].

2) Modifications to Flash array organization: We introduce design modifications to the Flash memory so that it is essentially transformed into an array of MLBs, which communicate over a hierarchical interconnect.
i. Compute Logic Modifications: A group of *N* Flash blocks is logically clustered to form a single MLB. MLB control logic (i.e. schedule table) and custom datapath are implemented using static CMOS logic. In light of write endurance problems for NAND Flash, we write to the Flash memory only during the configuration phase (application-mapping phase); a custom *dual ported asynchronous read register file* for storing the temporary/intermediate outputs. A fast intra-MLB multiplexor

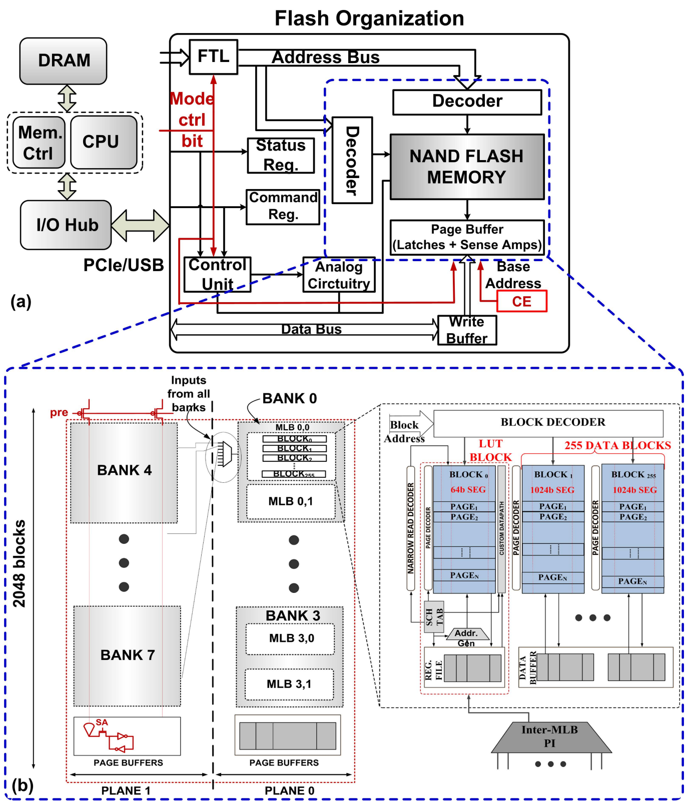

Fig. 7. a) Modifications to Flash memory interface to realize *MAHA* framework. A small control engine (CE) outside the memory array is added to initiate and synchronize parallel operations inside the memory array; b) Modified Flash memory array for on-demand reconfigurable computing. The memory blocks (called MLB) are augmented with local control and compute logic to act as a hardware reconfigurable unit.

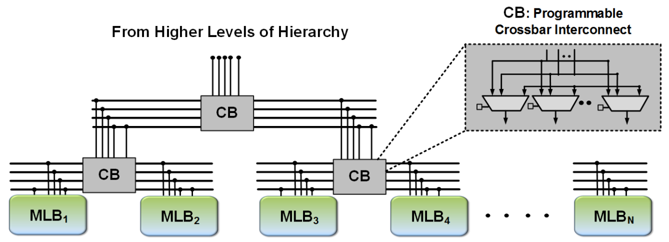
From Higher Levels of Hierarchy
CB: Programmable Crossbar Interconnect
CB
MLB1
MLB2
MLB3
MLB4
MLBN

Fig. 8. Hierarchical programmable interconnect architecture to connect a group of MLBs.

tree consisting of pass gate multiplexors and weak keepers is responsible for selecting appropriate operand(s) for the memory and the datapath. All operations within a given MLB are scheduled beforehand and stored as a μ-code inside the schedule table, implemented using a 2-D flip-flop array. For a normal NAND Flash read, the entire page (typically 2KB) is read at a time. However, to operate on data of smaller sizes, we propose a *narrow-read* scheme for the memory blocks in which a fraction of a page size is read at a time. The proposed scheme incurs hardware overhead due to wordline segmentation, but improves energy-efficiency in scenarios where operands of smaller sizes are stored in contiguous locations of a given page. In this scheme data sizes of 4096 bits being read out from each page and stored inside buffers. The two planes of the Flash array are logically divided into 8 banks each consisting of 2 MLBs each. Each MLB contains 256 blocks of Flash memory.

ii. Routing logic modifications: In order to minimize the inter-MLB PI overhead, we assume a set of hierarchical buses with a crossbar present at each level of hierarchy (Fig. 8). We have assumed a hierarchy with 4 levels similar to banks, subbanks, mats and subarrays present in a typical cache data array.

iii. ECC Computation: Block management and Error Control Coding (ECC) are well-known techniques to ensure reliability for normal Flash read and write operations. Typically ECC in NAND Flash is capable of correcting 1-bit and detecting up to 2-bit errors (SECDED) per 512 bytes (4096 bits) [12] and ECC bits are interleaved with data. In the MAHA framework, ECC check is performed inside each MLB for every read operation. Any error detected by SECDED hardware is indicated to the Flash management layer and execution is stalled.

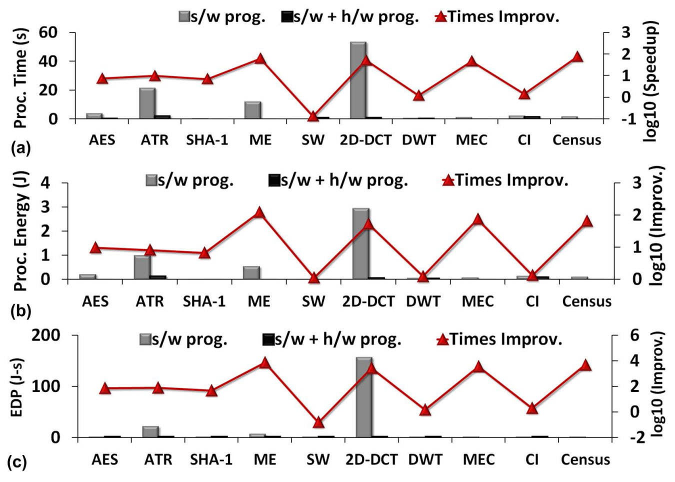

Fig. 9. Comparison of a) total processing time; b) total energy requirement and c) energy-efficiency (measured in terms of EDP) between a CPU only system and a system with CPU and MAHA based hardware accelerator.

IV. Results

A. Simulation Results

After considering the trade-offs associated with the MLB and interconnect design, we arrived at a NAND based MAHA architecture with 16 MLBs arranged in a hierarchy of 8,1,1,2. Since each MLB comprises of 256 blocks, all 4096 blocks of the 1GB Flash are organized into the 8,1,1,2 hierarchy. Compared to the baseline Flash design, the area overhead for this final MAHA configuration is found to be only 5.3%. Energy and performance of a hybrid system with both software execution engine and MAHA hardware accelerator is compared against a software only solution. In the latter, the baseline CPU is a 2-way in-order machine with 32KB 1-cycle I/D L1 and 512KB 4-cycle L2 cache. Typical off-chip access latency and energy estimates were obtained from [2]. Simulation results with the MAHA framework show considerable improvements for data-intensive applications. These findings are presented in Fig. 9 and summarized below.

i. Reduction in off-chip communication: From our simulation, we note that for data-intensive applications such as *Time-Frequency Transformations* where the output data size is of similar order of magnitude to the input data size, execution time and energy is dominated by data transfer from off-chip to on-chip memory. For these applications, a lower improvement

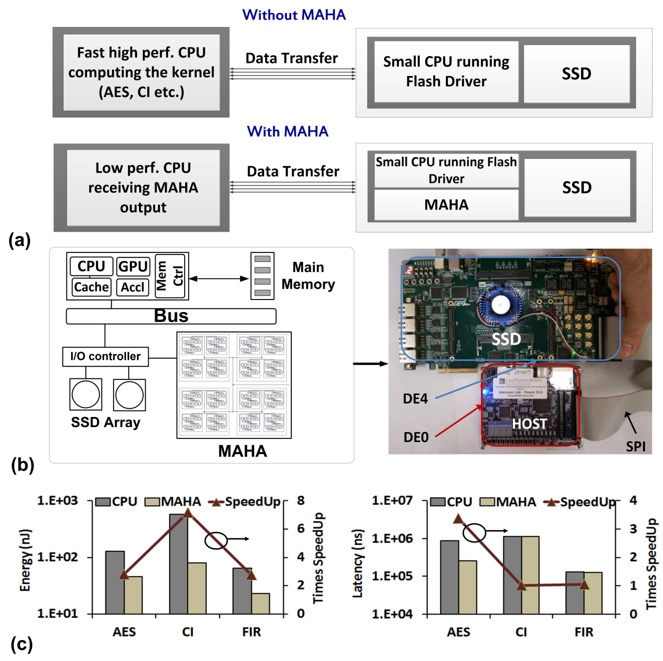

Fig. 10. a) Overview for off-chip acceleration with *MAHA* framework. b) System architecture for FPGA based hardware emulation framework. c) Improvement in latency and energy with *MAHA* based off-chip acceleration.

in off-chip traffic is observed. On an average, we observe *75%* improvement in execution time on the baseline processor, *71%* in on-chip traffic and *45%* improvement in off-chip traffic. Note that maximum savings is for *Mapreduce* class of applications which have large input and small output data set and are not compute-intensive.

ii. Improvement in energy-efficiency: Fig. 9(c) compares the energy efficiency for the applications mapped. On an average *MAHA* improves the energy-efficiency by 91.2×. The times improvement in energy-efficiency compared to the baseline CPU model varies over a large range, from *1.5X* to *4900X* for *Census - a Mapreduce* application. This proves that not all applications are amenable to acceleration through in-memory computing. The applications studied are therefore categorized as:

i. Applications (e.g. *Census*) which are purely data intensive where the output data size is significantly less compared to the input data size benefit most.

ii. Applications (e.g. *2D-DCT*) which are moderately compute intensive, but the output data size is equivalent to the input data size benefit less.

iii. Applications (e.g. *AES*) which are compute-heavy and the output size is same or less than input size benefit still less.

iv. Applications (e.g. *color interpolation*) with low computation requirement and the output size is same as input data size are least likely to benefit.

.

B. Emulation Results

We have developed a FPGA-based emulation framework which validates i) functionality and synchronization of multiple MLBs for several application kernels; and ii) interfacing the MAHA framework with the host processor. In this setup, MAHA behaves as an energy-efficient loosely-couple off-chip accelerator to which data-intensive kernels can be off-loaded. The overview of the scheme is illustrated in Fig. 10(a) with architecture and system details provided in Fig. 10(b). The hardware emulation framework was developed in the Altera Stratix IV FPGA environment. Energy values were obtained using DE4's onboard current sensor and latency from SignalTap Logic Analyzer. Based on cycle-accurate experimental data (Fig. 10(c)), we observed average improvement of 1.3X in latency and 4.3X in energy for the three benchmarks. EDP improvements in the order of 10X, 7X and 3X was achieved on the emulation platform.

V. Conclusion

The advent of non-volatile random-access memories with fast and energy-efficient read/write capabilities has opened new avenues in computer architecture research. In this work, we have presented *MAHA*, a spatio-temporal reconfigurable hardware which can greatly improve the energy-efficiency for data-intensive applications by computing in close proximity to the non-volatile storage. Design-time modifications to the NVM which allows the memory array to dynamically transform into a reconfigurable hardware on-demand has been described in detail. Our investigation demonstrates MAHA improves energy-efficiency for data-intensive tasks and that key application and system primitives can be leveraged to apriori identify the applications which can benefit from the proposed in-memory computing approach.

References

[1] Intl. Solid State Circuits Conference 2013 Tech Trends [Online] http://isscc.org/trends/

[2] P. Kogge et al., "ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems", *Exascale Computing Study Report* [Online] http://users.ece.gatech.edu/mrichard/ExascaleComputingStudyReports/exascale_final_report_100208.pdf

[3] Nicholas. P. Carter et al., "Runnemede: An Architecture for Ubiquitous High-Performance Computing", *Intl. Symposium on High-Performance Computer Architecture*, 2013.

[4] S. Pawlowski, "Architectural Considerations For Todays Technology Trends", [Online] http://eecs.oregonstate.edu/research/vlsi/teaching/ECE570_WIN13/Pawlowski\%20February\%2022\%202013\%20OSU.pdf

[5] C. Villa et al., "A 45nm 1Gb 1.8V phase-change memory", *Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 2010 IEEE International , vol., no., pp.270,271, 7-11 Feb. 2010.

[6] A. Kawahara et al., "An 8Mb multi-layered cross-point ReRAM macro with 443MB/s write throughput", *Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 2012 IEEE International , vol., no., pp.432,434, 19-23 Feb. 2012.

[7] T. Kawahara, "Challenges toward gigabit-scale spin-transfer torque random access memory and beyond for normally off, green information technology infrastructure", *Jnl. of Applied Phys.*, Vol. 109, No. 7, 2011.

[8] C.E. Kozyrakis et al., "Scalable processors in the billion-transistor era: IRAM", *Computer*, Vol. 30, No. 9, 1997.

[9] Final Report: Processor-In-Memory (PIM) based Architectures for PetaFlops Potential Massively Parallel Processing [Online] http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.g

[10] S. Paul et al., "Energy-Efficient Reconfigurable Computing Using a Circuit-Architecture-Software Co-Design Approach", *IEEE Journal on Emerging and Selected Topics in Ckts. and Sys.*, Vol. 1, No. 3, 2011.

[11] W. Kim et al., "Multi-layered vertical gate NAND flash overcoming stacking limit for terabit density storage", *Proc. VLSI Tech.*, June, 2009.

[12] Micron 1Gb NAND Flash Data Sheet [Online] http://download.micron.com/pdf/datasheets/flash/nand/1gb_nand_m48a.pdf