Phase-Change Memory Controller Architecture for Low-Latency Access in OpenPOWER Systems

A. Prodromakis*, N. Papandreou†, E. Bougioukou*, U. Egger†, N. Toulgaridis*, Th. Antonakopoulos*,
H. Pozidis†, E. Eleftheriou†
†IBM Research - Zurich, Säumerstrasse 4, 8803 Rüschlikon, Switzerland
*University of Patras, 26500, Rio-Patras, Greece

***Abstract*— Novel forms of nonvolatile memory, such as phase-change memory (PCM), promise low latency and small granularity of read and write access at high storage density. They also feature very high endurance. These characteristics make them highly desirable for emerging high-capacity (hybrid) memory applications such as in-memory databases and in-memory processing. In this work we present the architecture, implementation and experimental performance results of an FPGA-based PCM memory controller for OpenPOWER servers. The memory controller leverages the Coherent Accelerator Processor Interface (CAPI) of the POWER processor in order to offer low-latency access to the CPU memory space. In addition, the memory controller implements an efficient management protocol that supports a dynamic size of pending read and write requests in order to offer high bandwidth under mixed-type workloads. We describe the architecture and implementation details of the memory controller and we demonstrate its performance using a prototype platform based on different types of OpenPOWER servers equipped with CAPI-enabled FPGA cards. The developed PCM controller is evaluated in terms of sustained data rates (MBps) and access latency (us). Experimental results are based on legacy commercial 90nm PCM chips as well as on accurate HW emulation of next generation PCM chips.**

I. Introduction

In the last few years we are witnessing the emergence of new forms of nonvolatile memory (NVM), such as phase-change memory (PCM), magnetic random access memory and resistive random access memory. These new memory technologies exhibit a combination of characteristics that make them at least partially suitable for both main memory and storage applications. In particular, they have fast read and write access time with respective latencies ranging between 50ns to 1us, they have good endurance that typically exceeds 1 million write cycles, they can be written in place and the allow byte-level access granularity. In addition, they are nonvolatile, have moderate-to-long data retention, and exhibit excellent scalability, making them amenable to integration at very high storage densities. As a result, these new NVM technologies appear ready to establish a new tier in the memory hierarchy in between today's DRAM and NAND Flash technologies. At the same time, they promise to enable new applications such as hybrid memory, i.e., a combination of DRAM as a small main memory and NVM as the large far memory, or fast-durable storage where the NVM is used as a cache for hot data in front of a Flash storage pool.

Together with the big promises often come significant system-level integration challenges [1]. One such challenge stems from the low read/write (R/W) access latency offered by the new NVM technologies, in particular for storage-type applications. Specifically, attaching NVM to the host through the I/O bus and accessing it via a conventional storage protocol will likely add substantial access time overhead and thus the low-latency potential of the NVM will be unrealized. This is because legacy storage protocols have been designed to interface with hard disk drives originally and lately also with Flash, however new NVM is 2-3 orders of magnitude faster than Flash. Therefore, there is a need to re-design storage access protocols in a way that unleash the NVM low-latency potential.

In this work we present an approach that facilitates fast access to PCM. PCM has developed into a mature technology and is considered the top contender for realizing storage-class memory [2], [3]. The paper presents the architecture of an FPGA-based PCM memory controller for OpenPOWER servers. The memory controller leverages the Coherent Accelerator Processor Interface (CAPI) of the POWER processor in order to offer low-latency access to the CPU memory space. We describe the implementation details of the PCM controller and we demonstrate its performance using a prototype platform that consists of different servers equipped with CAPI-enabled FPGA cards. We showcase the system performance using custom DIMMs designed with legacy 90nm PCM chips as well as based on accurate HW emulation of next generation PCM technology. We discuss the architecture of the two controller design and evaluate their performance in terms of sustained data rates and access latency.

II. FPGA Architecture

Fig. 1 illustrates the general architecture of the system that consists of the POWER8 server equipped with a PCIe-Gen3 FPGA card that interfaces to custom designed DIMM modules populated with PCM chips. Fig. 1 presents the CAPI-based PCM controller. The FPGA implements the Power Service Layer (PSL) along with a custom Accelerator Functional Unit (AFU) to enable the Host application running on the POWER8 server to access the PCM memory over the CAPI interface. The basic memory controller implements a seamless CAPI adaptation layer (CAL) to the AFU and in addition integrates the data and address management as well as the PHY operations to the PCM DIMM modules. The next sections describes the details of the FPGA architecture.

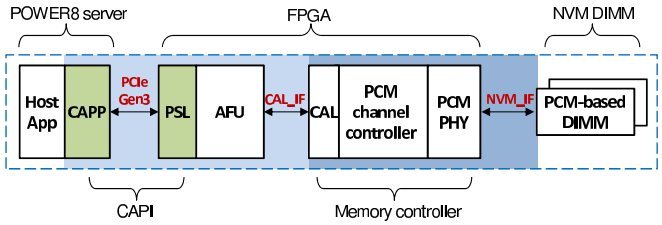

Fig. 1. General system architecture comprising of a POWER8 server equipped with a PCIe-Gen3 FPGA card that connects to custom PCM DIMMs.

A. Accelerator Functional Unit

The FPGA architecture of the PCM controller design consists of three main parts: the PSL which enables the FPGA device to communicate with a CAPI-enabled POWER8 server over the PCIe bus, a custom AFU that implements the management of the read and write requests from the Host over the CAPI protocol and the basic PCM memory controller that implements the low-level memory managements and PHY operations.

In particular, PSL allows the FPGA device to communicate with a CAPI-enabled POWER8 server [4]. PSL provides memory address translation via a memory management unit, which allows an AFU to use effective addresses in order to reference data structures in the same manner that they are used by software applications running on the Host CPU. The PSL to AFU interface consists of five independent interfaces that allow the AFU to access the system memory through load and store requests which are cache-line oriented (128 bytes). The MMIO and Control interfaces allow the software application to read and control the status of the AFU, while the Command, Buffer and Response interfaces allow the AFU to access data in the system memory.

The task that the AFU has been assigned to execute is defined by a Work Element Descriptor (WED). A WED is generated by the Host application and has a fixed size of 128 bytes. In our application a WED supports a varying number of one to eight commands. Each command has a length of 16 bytes and contains a Command Field that specifies the command type (read or write) and the block size (in multiples of 128 bytes), a Device Cache-line Offset Field that refers to the block offset in the device memory, and a Host Base Address Field that refers to the effective base address in the Host memory. This structure allows full utilization of the WED size and enables multiple threads running on the Host to form a single WED with multiple commands. The Host activates an AFU by sending a WED pointer.

Fig. 2 describes the architecture of our custom AFU that consist of two main parts. The first part implements the logic to meet the timing and synchronization requirements of the PSL interface as well as a buffering stage where the 64-bit effective base addresses of incoming WED pointers are stored. The second part of the AFU core consists of the master FSM and four special engines. The master FSM checks the status of the WED pointer FIFO and if there are pending request it activates the RWED engine. The latter is responsible to request

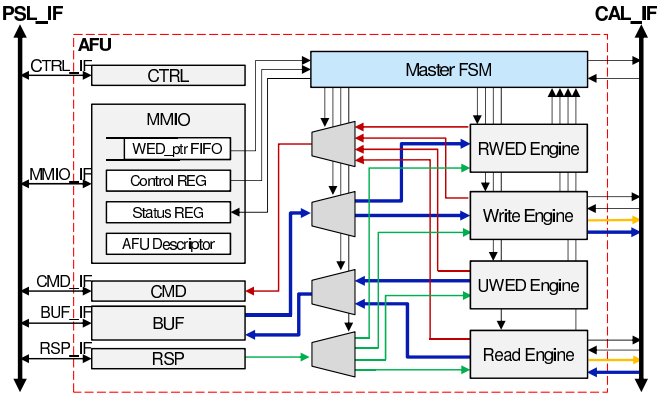

Fig. 2. Architecture of the CAPI-based accelerator functional unit (AFU).

and decode the WED content from the Host. Depending on the type of the incoming request, i.e. read or write, the master FSM activates the Read or Write Engine, respectively. PSL supports a maximum of 64 outstanding credit-based load and store requests. The Write and Read Engines implement the logic to utilize the maximum throughput by supplying the PSL with the maximum number of outstanding requests continuously. Moreover, they support re-ordering of command and data from PSL to further utilize performance under sequential workloads. When a command is completed, the UWED updates the WED structure in the system memory to acknowledge the completion of the particular WED. For write commands, completion is acknowledged when all data requests have been forwarded to the PCM controller. For read commands, completion is acknowledged when the AFU has sent all the data to the PSL.

The AFU interfaces with the PCM controller via the CAL interface which was designed in order to decouple the memory controller from the complexity of decoding, managing and servicing the WEDs. The CAL interface runs at 250 MHz clock frequency with two independent write and read paths, with 1024 bits and 64 bits data and address buses, respectively. For a write request, AFU sends the address, data and a write enable signal to CAL in one clock cycle, while CAL responds with a *write ready* signal if it can accept new requests. Similarly, for a read request, if the *read ready* signal is active, the AFU issues a read request by sending the read address and a read enable signal. CAL return the read data along with a valid signal after a number of clock cycles that depend on the latencies of the memory controller and of the memory chip.

B. Phase-Change Memory Controller

Two different PCM controllers have been designed and implemented. The first controller is based on legacy commercial 90 nm SLC PCM chips and the second one is based on next generation SLC and MLC PCM chip specifications. Fig. 3 illustrates the memory controller architecture for the legacy 90nm PCM chips. These particular chips use a serial interface of 66 MHz that accommodates data and address bytes on the same signals and supports 64 byte R/W accesses. The PCM channel is organized in a novel 2D architecture that allows

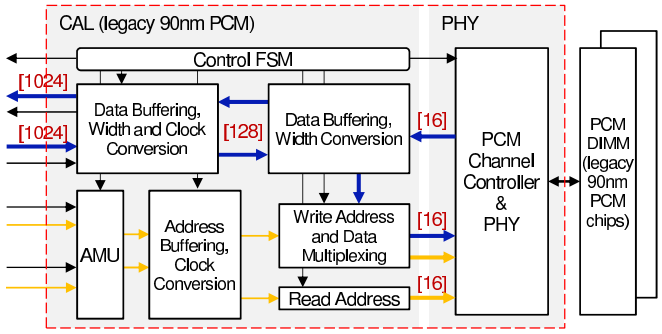

Fig. 3. Memory controller architecture for the legacy SLC PCM chip.

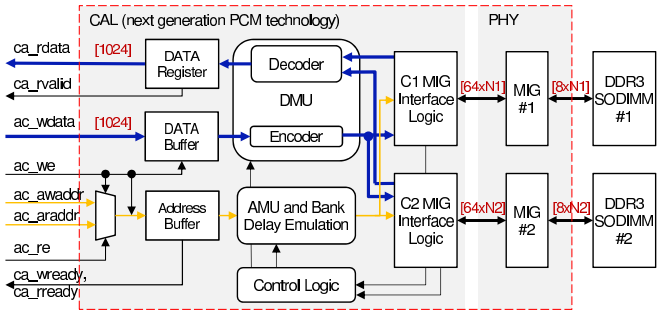

Fig. 4. Memory controller architecture for next generation PCM technology.

multiple chips to be organized into stripes and accessed in parallel. The channel controller operates at 125 MHz clock frequency and requires 16-bit wide address-bus and data-bus, therefore CAL implements the appropriate data width and clock domain conversion circuits. The Address Management Unit (AMU) is responsible for mapping the user addresses into channel and stripe addresses. In the write path, data and address are multiplexed following the chip serial specifications and sent to the channel controller. The PCM channel controller and PHY circuit implement the physical layer interface to the PCM chips. CAL supports R/E request from the AFU with a granularity of 128 bytes which are serviced by 2 PCM chips that belong to the same stripe. These chips have a typical page program cycle time of 120 us while read requests can be serviced continuously. The current CAL architecture supports up to eight PCM channels.

Fig. 4 illustrates the memory controller architecture for next generation PCM technology. Due to lack of commercial chips and in order to be able to test the system we have assumed a DDR3-compatible PCM chip interface and used DRAM modules to perform real time R/W access. However, in order to expose the Host application to the latency characteristics of the PCM technology, the memory controller implements the read and write access latency and active bank constraints of the PCM chip. In particular, we have assumed the specifications from state of the art 25nm PCM chips [5]. The AMU unit maps the received flat addresses into bank, row and column addresses to be written on the two SODIMM channels available in the FPGA card. The memory controller supports multiple configurations of the two SODIMM channels, i.e.

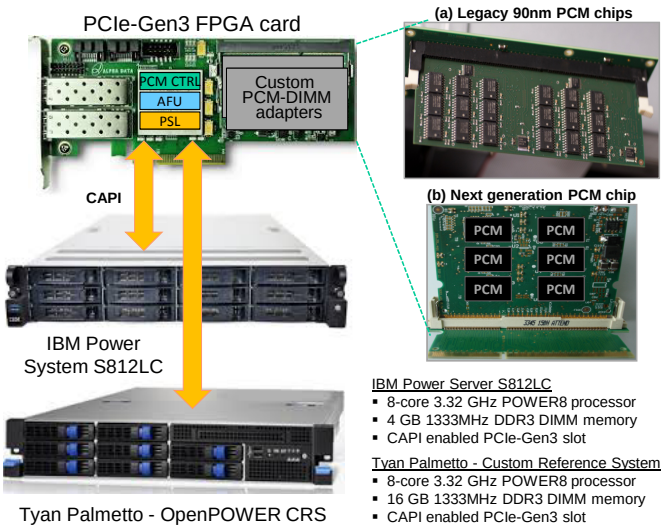

Fig. 5. Demonstration platform that comprises of two OpenPOWER servers each equipped with a Gen3-PCIe FPGA and custom DIMMs with PCM chips.

the controller can access less than 8 chips per SODIMM, in order to address the different requirements of extra parity chips for SLC or MLC PCM technology. The Data Management Unit (DMU) implements a BCH interleaved error correction encoder and decoder. In the current implementation every write request of 128 bytes is converted into four 256 bit codewords. The encoded data are sent to the two channels using dedicated PHY based on the DDR3 Memory Interface Generator (MIG) IP cores. User data are send to the first channel and parity data to the second one. A Control Logic unit is utilized to synchronize the two MIG interfaces. The proposed architecture can emulate different parameters for the PCM technology in terms of R/W access latency and thus it provides a flexible platform to study the performance of the new non-volatile memory in the system.

III. Performance Measurements

In order to evaluate the performance and end-to-end latency of the system we developed a benchmark tool based on the CAPI library. The tool runs on the POWER server and integrates special API functions to open, enable, communicate and close the AFU device in the FPGA card. In the current version, the tool generates a constant flow of random or sequential read or write traffic. The requests to the AFU are generated by sending the corresponding WED pointers. The data size associated with each request can vary from a minimum of 128 bytes to a few megabytes. In addition, the tool collects and reports statistics regarding throughput and latency. For throughput measurements, the tool generates a number of read and write requests of fixed data size, it records the total elapsed time up to the completion of the last request and finally calculates the performance. For latency measurements, the tool generates a number of read and write requests and in this case it waits for each request to be completed before sending the next one. For each issued request it records the elapsed time.

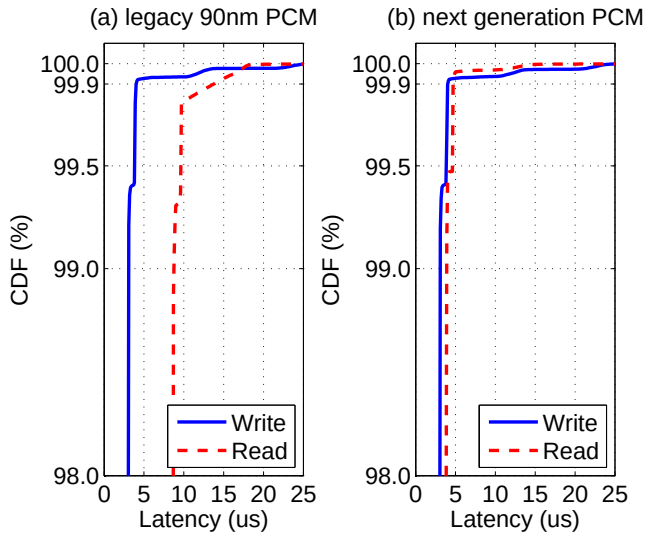

Fig. 6. Cumulative distribution function of end-to-end Write and Read access latency of 128 bytes using (a) the legacy 90-nm PCM chip and (b) the HW emulation of next generation PCM technology.

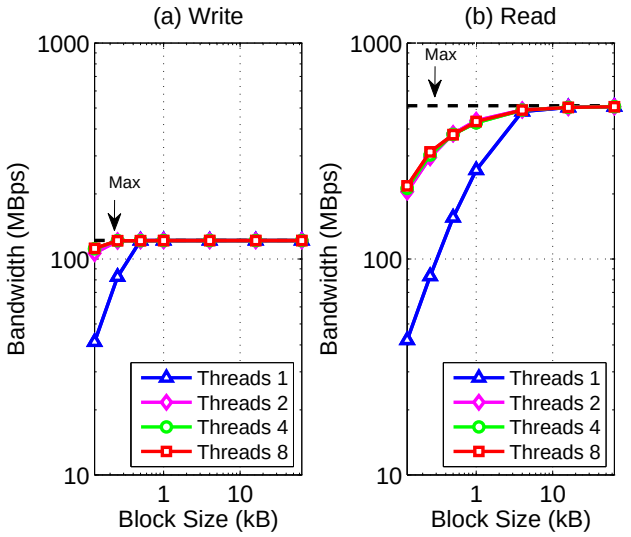

Fig. 7. Throughput performance of (a) write and (b) read measured for the next generation PCM chip under different number of threads running in the Host CPU.

TABLE I
LATENCY OF 128B WRITE-READ ACCESS

Legacy 90nm PCM chip	50%	99%	99.9%
Write	2.9 us	3.1 us	4.1 us
Read	8.6 us	8.8 us	13.8 us
Next generation PCM chip	50%	99%	99.9%
Write	2.9 us	3.1 us	4.1 us
Read	3.7 us	3.9 us	4.7 us

A. Demonstration platform

Fig. 5 illustrates the demonstration platform that consist of an IBM POWER8 server and a Tyan Palmetto P8 development platform both equipped with an ADM-PCIE-7V3 FPGA card with a Xilinx FPGA. Special DIMMs have been used for both the legacy as well as for next generation PCM chips. The DIMMs are directly connected to each FPGA card via custom SODIMM adapters. Both servers are CAPI-enabled and access the PCM chips via the CAPI-based memory controllers.

B. Results

Fig. 6 shows the cumulative distribution function (CDF) of the latency of each controller for 128 byte write and read access while Table I reports the 50%, 90% and 99.9% values. For both controllers, 99% of the writes complete within 3.1 us. For the next generation chip, 99% of the reads complete within 3.9 us while for the legacy PCM the latency increases to 8.8 us, however half of this value is due to the slow serial command/data interface of the PCM chip. We have measured very similar results for both servers shown in Fig. 5.

Fig. 7 shows the throughput performance of the next generation PCM controller for sequential read and write workloads when multiple threads are running in the Host CPU. Thanks to the WED management protocol that supports a dynamic size of pending read and write requests described in Section II-A, the system performance increases with increasing number of threads and approaches the maximum throughput determined by the PHY specifications of the PCM chip technology.

IV. CONCLUSION

In this paper, we presented the FPGA architecture and implementation details of a PCM memory controller that leverages the CAPI interface in order to offer low-latency access to a POWER8 server. We demonstrated performance results using specially designed PCM-based DIMM modules using legacy 90nm SLC chip and via HW emulation of next generation PCM technology. A special WED management scheme allows full utilization of the available bandwidth of the PCM chips.

ACKNOWLEDGMENTS

We gratefully acknowledge our colleagues at IBM Research - Zurich, R. Polig, H. Giefers and C. Hagleitner, for their help in bringing up the CAPI functionality.

REFERENCES

[1] H. Hunter, L. A. Lastras-Montao, and B. Bhattacharjee, "Adapting server systems for new memory technologies," *Computer*, vol. 47, no. 9, pp. 78–84, Sept. 2014.

[2] R. Freitas and W. Wilcke, "Storage-class memory: The next storage system technology," *IBM J. Res. Develop.*, vol. 52, no. 4.5, pp. 439–447, July 2008.

[3] R. Haas, X.-Y. Hu, I. Koltsidas, and R. Pletka, "Subsystem and system-level implications of pcm," in *European Phase Change and Ovonics Symposium (EPCOS)*, 2011.

[4] "Keynote talk by dr. jeff stuecheli: Open innovation with power8," in *IEEE 25th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, June 2014, pp. 1–1.

[5] J. Cheon, I. Lee, C. Ahn, M. Stanisavljevic, A. Athmanathan, N. Papandreou, H. Pozidis, E. Eleftheriou, M. Shin, T. Kim, J. H. Kang, and J. H. Chun, "Non-resistance metric based read scheme for multi-level pcram in 25 nm technology," in *IEEE Custom Integrated Circuits Conference (CICC)*, Sept 2015, pp. 1–4.