J. Parallel Distrib. Comput. 106 (2017) 31–49

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

A novel hardware support for heterogeneous multi-core memory system

Tassadaq Hussain *

UCERD: Unal Color of Education Research and Development, Islamabad, Pakistan
Riphah International University, Islamabad, Pakistan
Microsoft Research Center and Barcelona Supercomputing Center, Spain

HIGHLIGHTS

- Support for both static and dynamic data-structures and memory access patterns.
- Specialized scratchpad memory is integrated to map complex access patterns.
- The data manager accesses, reuses and feeds complex patterns to the processing core.
- Complex patterns are managed at run-time, without the support of a master core.
- Can be integrated with soft/hard soft processor core.
- Support trace driven simulation and FPGA based real prototyping environment.

ARTICLE INFO

Article history:
Received 22 July 2015
Received in revised form
20 January 2017
Accepted 21 February 2017
Available online 16 March 2017

Keywords:
Heterogeneous
Controller
HPC

ABSTRACT

Memory technology is one of the cornerstones of heterogeneous multi-core system efficiency. Many memory techniques are developed to give good performance within the lowest possible energy budget. These technologies open new opportunities for the memory system architecture that serves as the primary means for data storage and data sharing between multiple heterogeneous cores. In this paper, we study existing state of the art memories, discuss a conventional memory system and propose a novel hardware mechanism for heterogeneous multi-core memory system called Pattern Aware Memory System (*PAMS*). The PAMS supports static and dynamic data structures using descriptors and specialized scratchpad memory. In order to prove the proposed memory system, we implemented and tested it on real-prototype and simulator environments. The benchmarking results on real-prototype hardware show that PAMS achieves a maximum speedup of 12.83x and 1.23x for static and dynamic data structures respectively. When compared to the *Baseline System*, the PAMS consumes up to 4.8 times less program memory for static and dynamic data structures respectively. The PAMS consumes 4.6% and 1.6 times less dynamic power and energy respectively. The results of simulator environment show that the PAMS transfers data-structures up to 5.12x faster than the baseline system.

1. Introduction

For a long era, speed was one and only source which was used for estimating the performance of any High Performance Computing (HPC) systems. Both, raw computing performance and performance per watt are equally important. Nowadays, power consumption and power density determine the system performance [18]. Besides, supercomputers, servers, and data centers also have the power constraints. To achieve the performance, the computer architectures use heterogeneous accelerator cores. The system architects are targeting low power and high-performance HPC systems having general purpose processing cores [49] and application specific accelerators [61].

As the amount of on-chip gates increases, there is a dramatic increase in size and architecture of local memories such as caches. The concept of scratchpad memory [3] is an important architectural consideration in modern HPC systems, where advanced technologies have made it possible to combine with DRAM. With the unveiling of on-chip memories such as memristors [25] and embedded DRAMs [47] the size of on-chip memory is getting a dramatic increase. Embedded DRAM (eDRAM) has the advantage of having much higher density and lower power consump-

* Correspondence to: UCERD: Unal Color of Education Research and Development, Islamabad, Pakistan.
E-mail address: tassadaq@ucerd.com.

http://dx.doi.org/10.1016/j.jpdc.2017.02.008
0743-7315/

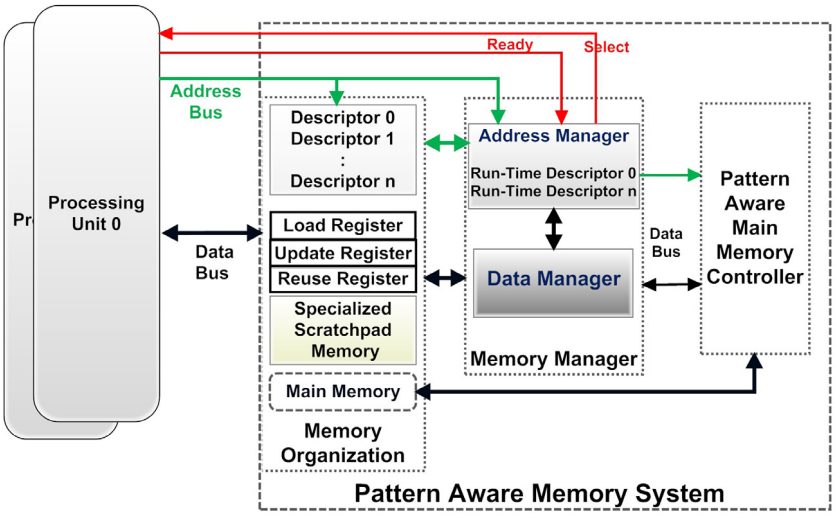

Fig. 1. Architecture of Pattern Aware Memory System.

tion than the traditional SRAM technology used for local memory. Integrating eDRAM on the same silicon chip with the processing logic holds the promise of greater bandwidth access than through external interfaces. Although the data read and write latency remains to be a bottleneck for new memory technologies. Having huge *Local Memory* as shared memory still requires a memory system in hardware and/or software that hides the on-chip communication mechanism between the applications.

Integrating intelligent registers [69,42] inside application specific processors improves the performance of the memory hierarchy. The register memory enhances the performance for applications having data locality but does not support applications with large and dynamic data-structures. Since different applications have different memory access patterns and data-structures, finding one topology that fits well for all applications is difficult. Integrating more memory controllers on system platform can increase bandwidth, but require many Input/output pins that consume power. Therefore, a memory system requires an intelligent memory controller that manages and schedules the data accesses.

Increasing the number of memories and placing them near the processing core is not the only way to address and improve the performance of applications. We present a technique to minimize the on-chip data transfer execution time of shared/distributed systems by a careful partitioning of the local memory, access patterns, and schedule and manage them in hardware. In this work, we propose the Pattern Aware Memory System (PAMS), a memory system for heterogeneous multi-core architectures. PAMS accelerates both static and dynamic data structures and their access patterns by arranging memory accesses to minimize access latency based on the information provided by pattern descriptors. PAMS operates independently from the master core at run-time. PAMS keeps data-structures and access pattern descriptors in a separate memory and prefetches the entire data structure into a special scratchpad memory. Data-structures and memory accesses are arranged in the pattern descriptors at program time and PAMS manages multiple patterns at run-time to reduce access latency. PAMS controls data movement between the *Main Memory* and the specialized scratchpad memory; data present in the *Local Memory* is reused and/or updated when accessed by several patterns. The significant contributions of the proposed PAMS architecture are:

- Support for both static and dynamic data structures and memory access patterns using the memory pattern descriptors thus reducing the impact of memory latency.
- A specialized scratchpad memory that is tailored for the *Local Memory* organization and maps complex access patterns.
- A data manager that efficiently accesses, reuses and feeds data to the computing unit.
- Data management and handling of complex memory access at run-time, without the support of a processor or the operating system.
- When compared to the *ARM based Multi-core System*, PAMS achieves between 1.35× to 12.83× and 1.12× to 1.23× of speedup for applications having static and dynamic data structures respectively. The PAMS consumes 4.6% and 1.6 times less dynamic power and energy respectively.
- In a many-core trace-driven simulation environment, the PAMS transfers data-structures up to 5.12× faster than the baseline system.

The rest of this paper is organized as follows: Sections 2 and 3 describe PAMS and Programming Model respectively. Sections 4 and 5 present the evaluation architectures and results respectively. Finally, Section 6 discusses the related work and Section 7 provides the conclusions.

2. Pattern Aware Memory System (PAMS)

The proposed PAMS (shown in Fig. 1) performs isolation and management of data-structures using *Specialized Scratchpad Memory* and improves data transfers by arranging access requests using *Descriptor Memory*. The *Descriptor Memory* manages data structures using single or multiple descriptor blocks and organizes data access requests in the form of patterns that reduce the run-time address generation and management overhead and avoids request grant time. The *Descriptor Memory* manages compile-time as well as run-time generated memory accesses of the applications. The PAMS *Memory Manager* handles *Specialized Scratchpad Memory* data and performs load, reuse and update data operations that avoid accessing the same data multiple times. At run-time, the PAMS schedules data accesses according to the application needs and applies fair data transfer scheme. The PAMS *Pattern Aware Main Memory Controller* transfers data between the *Main Memory* and *Local Memory* by maximum utilizing multiple DRAM banks. This section is further divided into following subsections, the *Memory Organization*, the *Data-Structures and Access Description*, the *Memory Manager* and the *Pattern Aware Main Memory Controller*.

2.1. Memory organization

To provide isolation and improve data locality the PAMS memory is subdivided into four sections that are: the *Descriptor Memory*, *Buffer Memory*, *Specialized Scratchpad Memory* and *Main Memory*.