Computers and Electrical Engineering 51 (2016) 184–194

Contents lists available at ScienceDirect

Computers and Electrical Engineering

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

A novel memory management method for multi-core processors ☆

Jih-Fu Tu*

Department of Electronic Engineering, St. John's University, 499, Sec. 4, Tamking Road, Tamsui District, New Taipei City, Taiwan

ARTICLE INFO

Article history:
Received 31 May 2015
Revised 21 October 2015
Accepted 26 October 2015
Available online 14 December 2015

Keywords:
Multicore
system-on-chip
hardwired

ABSTRACT

This study examines a multicore processor based on a system-on-chip (SoC) and configured by a Tensilica Xtensa® LX2. The multicore processor is a heterogeneous, configurable dual-core processor. In this study, one core was used as the host to control the processor chip, and the other was used as a slave to extend digital signal processing applications. Each core not only owned its local memory, but also shared common data memory. In addition, the proposed multicore processors had a virtual memory. This additional memory supported the processor by enabling it to easily manage complex programs; it also allowed the two cores to access data from the unified data memory of different tasks. For bus management, a bus arbitration mechanism was added to handle the cores and to distribute the priority of asynchronous access requests. The benefits of the proposed structure include avoiding hardwired memory and reducing interface handshaking. To verify the proposed processor, it was simulated on the model level using a Petri net graph, and on the system level using ARM SoC designer tools. In the performance simulation, we found that the lowest latency-to-cost ratios were achieved using a 32-bit bus interface and a 4-entry data queue.

1. Introduction

In terms of embedded systems, the multiply function in handheld devices often encounters issues related to power consumption, speed, and heat. The performance of these devices is mainly dependent on processor clock frequency; however, to increase performance, a higher current is required to drive the hardwired logic. With the decreasing sizes of their silicon areas and their need to conserve power, portable devices are often unable to achieve faster frequencies and higher performance. Contrary to expectations, more circuits have been added, which increases power consumption. To meet new requirements, we need new solutions to these problems. Many multimedia handheld devices now contain multicore processors, and execute system control and multimedia computing from separate cores.

In 2014, [1] described a quasi-partitioning scheme for last-level caches that combined memory-level parallelism, cache friendliness, and the interference sensitivity of competing applications, to efficiently manage the shared cache capacity. Although a heterogeneous core may reduce burdens by executing predefined tasks in advance, maintaining these two sets of development environments often requires significant costs and labor. However, future processor systems will increasingly utilize parallel processing; multithreaded dispatchers will manage performance. This allows the operating system (OS) to access available processors to manage the OS, communications, and multimedia, and reduces the need to increase CPU clock speed. In the past, it was

☆ Reviews processed and recommended for publication to the Editor-in-Chief by Guest Editor Dr. T-H Meen.

* Tel.: +886 930038679.
E-mail address: tu@mail.sju.edu.tw

http://dx.doi.org/10.1016/j.compeleceng.2015.10.009
0045-7906/

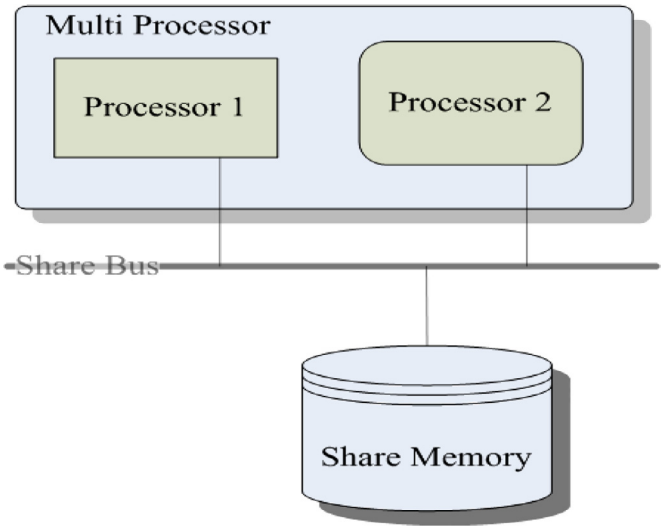

Fig. 1. Basic structure of the multicore design.

difficult for multicore architectures to fully satisfy these application demands simultaneously. We design a multiprocessor model that supports embedded system applications. It requires that an instruction set architecture be integrated into a multiprocessor system-on-chip (MPSoC) based on configurable cores. This paper describes a configurable multicore structure.

In this study, we implemented a dual-core system and a shared-memory bus arbitration method that handles shared-memory accesses; the basic structure is shown in Fig. 1. Other components (such as hardware accelerators or other cores) can be added to this simulator system for co-simulation, provided they contain a shared-memory bus interface.

We exploited the parallelism in random-access schemes to implement a multiple-core processor (shown in Fig. 1) that contains multiple function units. These include the microprocessor, switch engineer for arbitrating the use of the assigned bus, and the shared-memory module, which is globally addressable and can be distributed among the microprocessors or centralized in one place.

We used a Tensilica Xtensa® LX2 [10] to design the multicore processors. When collocated with Diamond standard processors, which can provide superior processing capability compared to traditional designs, the LX2 benefits in terms of area and power characteristics. This implementation can be configured in a multiprocessor architecture, and verified by a provided simulation program. The main work of systems integration includes the multicores, local memory, cache, shared memory, and shared bus.

The remainder of this paper is organized as follows. Section 2 surveys previous related studies. The design methodology is introduced in Section 3. Section 4 presents the results of the proposed processor. Finally, we present the conclusions in Section 5.

2. Background

There is increasing demand for embedded multimedia communication systems in mobile and portable device applications. For multimedia communications, the implementation of audio and video compression standards is essential. In addition, a system demanding improved performance requires a higher clock frequency. As such, multiple-function handheld devices are often challenged by problems related to power consumption, clock speed, and heat dissipation.

2.1. Related works

In many multicore in-order processing systems, only one core can be used when the instruction at the head of the queue produces data input for the next instruction in the queue [2]. To achieve higher performance and flexibility, hybrid architecture has been proposed. In this architecture, operation-intensive functions are implemented with hardwired blocks, and other less complex functions are implemented with software executed by an application-specific instruction processor. Current multimedia handheld devices often use built-in multicore approaches [3]. System control and multimedia computations are executed using separated cores. Although heterogeneous cores reduce burdens from predefined tasks [4], maintaining two sets of development environments requires significant costs and labor.

For example, Bernabe et al. [5] used the compute unified device architecture (CUDA) and the CUDA basic linear algebra subroutines library, which were tested on two different graphics processor unit architectures, to design a hyper spectral data cube. Shnaiderman and Shmueli [6] presented parallel path stack (PPS) and parallel twig stack (PTS) algorithms. The PPS and PTS algorithms are novel and efficient for matching extensible markup language query twig patterns in a parallel, multithreaded computing platform [6]. Chang et al. [7] explored the joint considerations of memory management and real-time task