Chapter 12
Programmability and Scalability on Multi-Core Architectures

Jaeyoung Yi
Yonsei University, Seoul, Korea

Yong J. Jang
Yonsei University, Seoul, Korea

Doohwan Oh
Yonsei University, Seoul, Korea

Won W. Ro
Yonsei University, Seoul, Korea

ABSTRACT
In this chapter, we will describe today’s technological trends on building a multi-core based microprocessor and its programmability and scalability issues. Ever since multi-core processors have been commercialized, we have seen many different multi-core processors. However, the issues related to how to utilize the physical parallelism of cores for software execution have not been suitably addressed so far. Compared to implementing multiple identical cores on a single chip, separating an original sequential program into multiple running threads has been an even more challenging task. In this chapter, we introduce several different software programs which can be successfully ported on the future multi-core based processors and describe how they could benefit from the multi-core systems. Towards the end, the future trends in the multi-core systems are overviewed.

INTRODUCTION
Intel has shipped the first dual-core processor as early as 2005 and many major processor vendors have developed dual-core or quad-core processors since then. We are now entering the new era of the multi-core processors, and practically every field in computer science or computer engineering will be affected by this strong movement. Though computing power has improved dramatically with higher clock frequencies and techniques such as superscalar, superpipelining, and VLIW (Very Large Instruc-
tion Word), it seems that this progress will see a significant slowdown and we will have to come up with other solutions to maintain the speed of improvement we are enjoying now.

There are three main reasons for the slowdown in single core performance improvement. First of all, we cannot increase the clock frequency following the Moore’s Law due to the power dissipation concerns and thermal problems. As millions of transistors are integrated onto one chip and as the clock speed goes up, the heat becomes too much to handle with current affordable cooling solutions. Secondly, the latency of processor-memory requests becomes a limiting factor, caused by the gap of speed advancements between the processor and the memory. Indeed, this becomes a major bottleneck for overall computing performance. Lastly, it is known that ILP (instruction-level parallelism) from a single thread almost reaches its limit in the current microprocessor architectures and compiler techniques. Therefore, we can see that the next path to take is the multi-core approach; instead of trying to improve the performance of single thread execution, we should partition applications into multiple threads that can run in parallel on prevailing multi-core systems.

In this chapter, we are going to look at recent researches on multi-core architectures and how to fully utilize the multi-core systems. In the next section, we look at hardware designs and characteristics of the multi-core architectures. In Section 3, we explain software programming skills to exploit parallelism in two specific applications which are network coding and Intrusion Detection Systems (IDS). In Section 4, we touch on the programmability and scalability issues to use multi-core systems for video applications, and in Section 5, we conclude the chapter with forecasting on future multi-core systems.

BACKGROUND STUDY: HARDWARE DESIGNS OF MULTI-CORE ARCHITECTURE

In this section, the general hardware architecture of current multi-core processors is surveyed as background study. We first describe the basic method to build multi-core processors and then describe the memory hierarchy design issues for multi-core system.

Multi-Core Processor Architecture: Homogeneous or Heterogeneous

In a simplest way to design multi-core processors, we can just imagine to arrange multiple processing units (so called cores) on a single chip. In theory, this might be a good way to boost performance by providing parallelism through more processors. However, in reality, multi-cores do not always promise the performance enhancement in software execution (Hill & Marty, 2008). In fact, it could cause opposite results due to some communication restrictions and memory sharing problems. Therefore, hardware designers of multi-core based processors have to research various methods of multi-core structure and simultaneously find effective algorithm of data sharing to boost performance and efficiency of processing.

According to an architect for a specific design, the internal architecture of each core can vary. Actually, the core can be either heterogeneous or homogeneous. In Figure 1, we show the two ways to build a multi-core processor. The left-side two diagrams show multi-core processor from Intel and AMD which can be classified as “homogeneous multi-core processors”. As the name implies, these models integrate identical cores on to a single chip. Yorkfield from Intel is a quad-core CPU that integrates two dual-core CPUs onto a single die. Phenome from AMD consists of four cores that have identical architecture and shared L3 cache on a single chip.