Chapter 4
Exploiting Parallelism: the 2D-Wave

Abstract  In the previous chapter we have analyzed various parallelization approaches for H.264 decoding and concluded that in order to scale to a large number of cores, macroblock-level parallelism needs to be exploited. The next question is how to efficiently exploit this parallelism. In other words, how to map this parallelism onto a multi-/many-core architecture. To answer this question, in this chapter we present two implementations of the 2D-Wave approach. The first implementation maintains a centralized pool of macroblocks that are ready to be decoded and cores retrieve tasks from this Task Pool. In the second approach, called Ring-Line, full lines of macroblocks are statically assigned to cores and the cores synchronize and communicate point-to-point. Both approaches have been implemented and are evaluated on a dual-chip Cell BE system with 18 cores in total.

4.1 Introduction

In Chapter 3 we have described the 2D-Wave algorithm and evaluated its scalability using an analytical approach, where we assumed that it takes constant time to decode a macroblock (MB) and that there is no communication and synchronization overhead. In reality, however, the MB decoding time varies significantly and communication and synchronization overhead needs to be minimized in order to obtain an efficient implementation. Thus the next question, which is addressed in this chapter, is how to efficiently implement the 2D-Wave algorithm.

Two important issues need to be considered when developing parallel applications: load balancing and communication/synchronization overhead. The load needs to be distributed equally over the cores so that cores are not idle waiting for results from other cores. For example, a static scheduling where first the first diagonal of parallel MBs is processed, then the second diagonal, and so on, cannot lead to an efficient implementation because the time to process a diagonal will be determined by the longest MB decoding task. Furthermore, the communication/synchronization overhead needs to be minimized, since when it is significant, the overall speedup...
will be limited. One technique to reduce communication overhead is by overlapping communication with computation.

This chapter presents two implementations of the 2D-Wave algorithm on a system consisting of two Cell BE processors. The first implementation, referred to as the Task Pool (TP) [2], is based on the master-slave programming paradigm. Slaves request work (in this case corresponding to MBs) from a master, which keeps track of the dependencies between the MBs. This implementation can achieve (in theory) perfect load balancing, since MBs are submitted to the task pool as soon as they can be decoded. In the second implementation [4], referred to as Ring-Line (RL), the cores process entire lines of MBs rather than single MBs. This approach does not require a centralized master to keep track of dependencies. Furthermore, its static mapping of MBs to cores allows overlapping communication with computation, since it is known a priori which core will decode which MB. A potential disadvantage of the RL implementation is, however, that it might cause load imbalance if MB lines have different processing times. These trade-offs are evaluated on an 18-core Cell BE system.

This chapter is organized as follows. In Section 4.2 a brief overview of the Cell architecture is presented. The TP implementation is described in Section 4.3 while Section 4.4 presents the RL approach. Experimental results are presented in Section 4.5, and conclusions are drawn in Section 4.6.

4.2 Cell Architecture Overview

The Cell Broadband Engine [7] (Cell BE) is a heterogeneous multi-core consisting of one PowerPC Element (PPE) and eight Synergistic Processing Elements (SPEs). The PPE is a dual-threaded general purpose PowerPC core with a 512 kB L2 cache. Its envisioned purpose is to act as the control/OS processor, while the eight SPEs should provide the computational power. Figure 4.1 shows a schematic overview of the Cell processor. The processing elements, memory controller, and external bus are connected to an Element Interconnect Bus (EIB). The EIB is a bi-directional ring interconnect with a peak bandwidth of 204.8 GB/s [3]. The XDR memory can deliver a sustained bandwidth of 25.6 GB/s.

Two features make the Cell processor an innovative design. First, it is a *functionally heterogeneous multi-core*, meaning that the cores are optimized for different types of code. In the Cell, the PPE is optimized for control/OS code, while the SPEs are targeted at throughput computing kernels. Second, it has a *scalable memory hierarchy*. In conventional homogeneous multi-core processors, each core has several levels of cache. The caches reduce the average latency and bandwidth requirements to the external (off-chip) memory. With multiple cores there are multiple caches and cache coherence is required to make sure that the data in the private cache of each core uses is up-to-date. The complexity of cache coherence increases with the number of cores, however. In the Cell architecture, the SPEs do not feature a cache but rely on a software-managed *local store* and a *Direct Memory Access* (DMA).