# SPM-Aware Scheduling for Nested Loops in CMP Systems

Zhi Chen, Meikang Qiu*
Dept. of Elec. and Comp. Engr., Univ. of Kentucky, Lexington, KY 40506, USA *

## 1. PROBLEM AND MOTIVATION

*Chip multiprocessors* (CMP) computing systems are usually employed to facilitate many specific applications including medical image processing, computer vision, and aerospace. In these computation-intensive applications, nested loops take the most significant section of computation cost and greatly affect system performance in terms of latency due to the frequent memory accesses. In order to enhance the parallelism of a nested loop, a critical work is to strategically map the iterative loops to processors so that we can exploit good parallelization of these loops and reduce the execution latency of the whole application. One of the most widely used method to do the iteration-to-processor mapping is pipelining, which enables each processor to perform the operations for the iteration mapped to it.

## 2. BACKGROUND AND RELATED WORK

Most prior work on nested loops either focuses on loop partitioning such as tiling, collapsing, and transformation, etc, or focuses on scheduling loops on different processors with hardware caches. However, hardware caches have shortcomings in size, power, and predictability and incur high penalties in cache misses. Statistically, caches are replacing the processors becoming the main energy consumer of computing systems. SPM, a software-controllable on-chip memory, has been widely utilized by many key manufacturers, due to two major advantages over a cache memory.

First, SPM does not have the comparator and tag SRAM, since it is accessed by direct addressing. Therefore, it does not perform the complex decode operations to support the runtime address mapping for references. This property of SPM can save a large amount of energy. Second, SPM generally guarantees single-cycle access latency, while accesses to cache may suffer capacity, compulsory, and conflict misses that incur very long latency [2]. Given these advantages, caches are widely used in CMP systems including Motorola M-core MMC221, IBM CELL [1], TI TMS370CX7X, and NVIDIA G80. Therefore, it is important to study the loop scheduling with SPM support to reduce the latency and energy consumption of nested loops on CMP system.

Although the obvious advantages SPM offers, two major challenges remain: 1) what is the basic granularity of the loop scheduling? 2) how to schedule nested loops on different SPMs of each processor in a CMP systems? Since there are a large amount of previous work targets nested loop partitioning, we mainly focus on the iteration-to-processor mapping by using the partitioning results from the existing techniques. Also, we are targeting the type of applications that intensively consist of loops manipulating arrays.

## 3. APPROACH AND UNIQUENESS

Depending on the partitioning technique such as cyclic partitioning and block partitioning, the granularity of partitioned loops varies. We focus on the block-based loops which can be obtained by using tiling techniques. Based on the obtained blocks, we construct a directed graph to represent the dependencies between them. Communication overhead between dependent blocks and the number of accesses to each processor can be learned with the help of profiling tools. Then, based on profiling information, a communication matrix and a processor access matrix can be built. We also define an assignment matrix to represent the mapping of each loop block to the SPM of each processor. The dimension of the assignment matrix is $N \times M$, where $N$ and $M$ represent the number of partitioned loop blocks and the number of processors on the target system, respectively.

In this paper, we will consider three types of on-chip memory access for loop scheduling: local access, remote access, and main memory access. Different memory access has different energy consumption and latency. In addition, the communication cost between different blocks will also be factored into the total execution cost. Since the target system is a CMP system, where each processor is attached with an on-chip SPM and all processors share an off-chip memory, there exist totally four kinds of communication: intra-memory communication, inter-memory communication, on-chip/off-chip memory communication, and off-chip memory communication. The overhead of each type of communication varies significantly. Our goal is to map all loop blocks in an applications to the on-chip SPMs and the off-chip main memory so that the total cost (either energy or latency) can be minimized.

A naive method is to distribute the loop blocks across available on-chip SPM evenly. However, this is not the best to implement the iteration-to-processor mapping for most of applications because different iteration blocks usually take different number of execution cycles and different amount of power, due to the existence of branches in loops and locality of on-chip SPMs. This paper carefully considers the characteristics of each loop block and proposes a dynamic programming algorithm to optimally perform the iteration-to-processor mapping. We will formally define the cost of memory access and communication, and use the availability of on-chip SPM resources to constrain the allocation.

## 4. RESULTS

Extensive experiments will be conducted to verify our algorithms on target a CMP system with 2, 4, and 8 cores. A host of benchmarks, such as MiBench and SPEC2006, will be used to evaluate the proposed strategies. We will also compare the efficiency of our dynamic programming algorithm with several other methods proposed in the literature. All these algorithms will be implemented as stand-alone programs, which take memory traces as inputs. Memory parameters are obtained from CACTI tools. Since the objective for this paper is to reduce the total memory access cost with respect to energy and latency, an array of results will be achieved to show how much on-chip memory energy and time (average and WCET) our dynamic algorithm can save.

## 5. REFERENCES

[1] C. R. Johns, et al. Introduction to the cell broadband engine architecture. *IBM Journal of Research and Development*, 51(5):503–520, 2007.

[2] P. R. Panda, et al. Efficient utilization of scratch-pad memory in embedded processor applications. In *IEEE/ACM DATE*, pages 7–11, 1997.

*Meikang Qiu (mqiu@engr.uky.edu) is the corresponding author, supported by NSF CNS-1249223.