Chapter 2.7
Parallel Programming and Its Architectures Based on Data Access Separated Algorithm Kernels

Dake Liu
Linköping University, Sweden

Joar Sohl
Linköping University, Sweden

Jian Wang
Linköping University, Sweden

ABSTRACT

A novel master-multi-SIMD architecture and its kernel (template) based parallel programming flow is introduced as a parallel signal processing platform. The name of the platform is ePUMA (embedded Parallel DSP processor architecture with Unique Memory Access). The essential technology is to separate data accessing kernels from arithmetic computing kernels so that the run-time cost of data access can be minimized by running it in parallel with algorithm computing. The SIMD memory subsystem architecture based on the proposed flow dramatically improves the total computing performance. The hardware system and programming flow introduced in this article will primarily aim at low-power high-performance embedded parallel computing with low silicon cost for communications and similar real-time signal processing.

1. INTRODUCTION

The programming and the architecture of real-time parallel computing for on-chip multicore computers are based on either general computing solutions or custom solutions. General solutions, usually based on a cache-coherent programming model, are not low cost solution for real-time applications (Hennessey & Patterson, 2003). Custom solutions are application-specific and suitable only for a selection of applications, such as LeoCore

DOI: 10.4018/978-1-61350-456-7.ch2.7

of Coresonic (Nilsson, Tell & Liu, 2008). Parallel programming based on architectures with local scratchpad memories associated with ultra large register files was proposed by Flachs et al., 2006, Khailany et al., 2008. A large register file supports flexible parallel programming and consumes much power. Parallel computing based on a VLIW DSP processor has been well used in industry (Tretter, 2003). However, VLIW based DSP processors cannot offer silicon efficiency and low power.

Currently master (host)-multi-SIMD based architecture is the main driver of embedded DSP computing. Several hundreds GOPS computing performance offers great opportunities for computationally demanding applications, yet some applications cannot be supported because of the high power consumption. A majority share of power is consumed during data access for parallel computing. Excessive and redundant parallel data access drives the clock frequency to a very high rate, so that the power consumption cannot be reduced by lowering the supply voltage.

1.1. Essential Glossary

OpenCL

OpenCL (**Open C**omputing **L**anguage) (Khronos, 2008) is a framework for writing programs that execute across heterogeneous platforms consisting of CPUs, GPUs, and other processors.

Kernel

The definition of a kernel by OpenCL: A *kernel* is a function declared in a *program* and executed on an OpenCL *device*. A *kernel* is identified by the **__kernel** qualifier.

From control complexity: A kernel is a subroutine executed independently in a SIMD or in an accelerator without interwork to its host machine or other SIMD.

From data complexity: Kernel is a computation that uses single the regular memory access pattern for each operand array (using only one addressing kernel / template).

From algorithm complexity: A kernel shall handle only one algorithm or part of an algorithm which can be implemented using only one loop.

Cluster

A cluster here consists of one master (host) machine and several SIMD machines.

Total data access cost in SIMD:

The run time cost of (1) loading data from the main memory to the SIMD local vector memory, (2) loading data from SIMD local vector memory to the vector register file, and (3) storing results from SIMD local vector memory to the main memory.

Data permutation:

The data permutation here in this article is used to select each piece of data in a vector and to store it in a memory block of the vector memory. It can be conducted during the data loading from the main memory to the local vector memory. The purpose of data permutation is to distribute data to different memory blocks in a vector memory so that multiple data values can be used in parallel simultaneously.

Conflict free memory access:

Based on data permutation, data is selected to be stored in different memory blocks. Multiple data can be accessed in parallel without conflict, facilitating parallel computing.

Separated data access kernel:

The data access kernel is separated from its original algorithm kernel. A kernel carries the data location information in the main memory and in the local vector memory. It also specifies the way that the data in the main memory is collected and merged into one DMA transaction, and the way that the data shall be distributed to each block of the vector parallel memory.

Prolog and Epilog in host:

It is a part of a context; a **prolog** is used to introduce a kernel to be executed in a SIMD ma-