C-based Synthesis of Area-Efficient Accelerators for OpenMP Worksharing Loops

Lukas Sommer Julian Oppermann Andreas Koch
Embedded Systems and Applications Group (ESA)
Technische Universität Darmstadt
{sommer, oppermann, koch}@esa.tu-darmstadt.de

Keywords

Heterogeneous computing; High-level synthesis; OpenMP; FPGA

1. INTRODUCTION

Heterogeneous computing is a promising approach to fill the gap between CPU performance and the increasing demand for computer performance. Next to GPUs, FPGA-based application-specific hardware accelerators have been established as an energy-efficient alternative.

To make FPGA-based accelerators available to a broad range of application developers, high-level synthesis tools are used to map software algorithm descriptions to hardware designs. Recently, many high-level synthesis tools started to exploit thread-level parallelism to further speedup computation on the accelerator, and to better utilize the increasing amount of hardware resources available on today's FPGAs.

For this work we have extended the Nymble hardware/software co-compilation framework [1] to automatically generate highly area-efficient FPGA-based multi-threaded hardware accelerators from OpenMP worksharing loops.

2. APPROACH

Starting from an OpenMP-annotated ANSI C input file, our tool Nymble-OMP maps the body of a worksharing loop to a hardware accelerator. At the top level, the generated accelerator consists of multiple instances of each hardware computing unit (analogous to a "core" in a CPU). Each computing unit comprises a datapath and the associated hardware controller.

In contrast to our prior work, the datapaths generated by Nymble-OMP provide access to external memories using a cache- and memory-architecture which is fully compliant with the OpenMP relaxed-consistency, shared memory model. Unlike our previous work, execution on the computing units is subject to a novel, multithreaded execution model, featuring dynamic thread interleaving.

The interfacing between the software running on the CPU and the hardware accelerator is automatically created by Nymble-OMP.

3. EXECUTION MODEL

Key to a high area-efficiency is a high utilization of hardware resources employed in the datapath, and an efficient handling of stalls encountered during the execution of variable latency operations (VLO), e.g., cached memory accesses.

In order to hide the latencies of these VLOs, our execution model temporally interleaves the execution of multiple independent threads on the same hardware, similar to a classic barrel processor. In case of a stall, a computing unit switches to execute another available hardware thread, while the processing of the VLO for the stalled thread continues in the background. Instead of idling, as in a single-threaded scenario, the datapath is now kept busy.

After the stall has been resolved, the stalled thread becomes available again and resumes execution at the next thread switch. The thread scheduler and context switching mechanisms themselves do not cause any additional latency for the executed thread.

4. RESULTS

We evaluate our approach using a benchmark comprising a set of OpenMP-parallelized matrix algorithms. The resulting hardware accelerators contain operators from a custom module library to perform memory accesses and typical arithmetic operations on integer and floating-point values.

Synthesis results show that the area overhead incurred for the generation of 4-way multithreaded compute units (e.g., for thread context storage in the datapath and thread switching logic in the controller), is as low as 5% per computing unit compared to single-threaded accelerators. This is much less than prior work, which often instantiates n separate compute units for n-threaded execution, requiring (at least) n-times the hardware area.

Comparing our implementation to single-threaded execution on a single computing unit, our experiments have at least doubled the efficiency (as throughput-per-hardware area). The best case we have observed with the current prototype increases throughput by a factor of 6 when interleaving the execution of 4 threads each on two computing units ("2C/8T", in CPU terms), with a total throughput-per-area efficiency of 2.8.

As the multithreading area overhead is only dependent on the *number* of operations in a datapath, but not on their individual size, we expect area efficiency to improve even further for datapaths with larger operators (e.g., floating-point).

5. REFERENCES

[1] J. Huthmann, B. Liebig, J. Oppermann, and A. Koch. Hardware/software co-compilation with the Nymble system. In *2013 8th International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*, pages 1–8, July 2013.