High-Level Synthesis of Multithreaded Accelerators for Irregular Applications

Stefano Devecchi, Nicola Saporetti, Marco Minutoli, Vito Giovanni Castellana, Marco Lattuada, Pietro Fezzardi, Antonino Tumeo and Fabrizio Ferrandi

Summary

Irregular Applications exhibit:
- Unpredictable, fine-grained data accesses
- Pointer or linked list-based data structures (e.g., graphs, unbalanced trees, unstructured grids, sparse matrices), difficult to partition in a balanced way
- Task-level parallelism
- High synchronization intensity
- Memory-bound (exploiting available bandwidth is non-trivial due to high memory reference rates)

Conventional High-Level Synthesis flows address:
- Dense, regular data structures
- Simple memory models
- Instruction-level parallelism
- Compute-bound kernels (Digital Signal Processing-like)
- OpenCL works well for regular, compute-bound workloads

Our contributions:
- Parallel distributed Controller (PC) for complex loops nests
- Hierarchical Memory Interface (HMI), supporting multi-banked/multi-ported memory and atomic memory operations
- Dynamic Task Scheduler (DTS) for unbalanced loop iterations
- New architecture template significantly extends the DTS design
- New architecture template significantly extends the DTS design
- Tolerates memory latency while computation continues
- Task status stored in Context Queues
- A task becomes ready again when memory operation completes
- Memory operations include atomics
- Whole template integrated in an open-source High-Level Synthesis tool
- Tool synthesizes starting from C sources annotated with OpenMP and explores parameters, such as the number of contexts, accelerators, and memory channels

Experimental Evaluation

Synthesis of graph walks (graph pattern matching routines) to solve Queries 1-7 of the Lehigh University Benchmark (LUBM) for the semantic web. Code is a set of nested loops that perform graph walks, look up labels, and count results with atomic memory operations in C. Latest version of Xilinx Vivado, Virtex-7 xc7vx690t.

Plots (only report query Q2), single context, and 32 contexts per accelerator:
- 2-32 accelerators (workers), 4-16 memory channels (CH)
- Demonstrate problem is memory bound
- Two accelerators and 32 contexts already maximize performance (memory throughput)
- Area for 2 and 32 workers when varying number of contexts (overheads due to muxes)

Tables, all queries:
- Parallel Controller (PC) only, Dynamic Scheduler (DT), and multithreading (Svelto)
- Four accelerators (PC and DT) vs. one accelerator with eight contexts (four memory channels)

Previous Architectural Templates

We have developed a set of solutions to address HLS for irregular applications:
- **Parallel distributed Controller (PC)** allows controlling pools of parallel accelerators (i.e., “hardware tasks”) with token passing mechanisms
- **Hierarchical Memory Interface (HMI)** allows supporting multi-ported shared memory with dynamic address resolution and atomic memory operations
- **Dynamic Task Scheduler (DTS)** allows launching a task as soon as an accelerator (kernel) in a pool is available (PC alone supports only block-based fork-joins)

References:

Temporaly Multithreaded Architecture Template and High-Level Synthesis Flow

4:00 PM – Opening Remarks, Pitch Party format

& Other Details

https://panda.dei.polimi.it