Improving Performance of SystemC Based Cycle-Level Simulations Using Multicore Platforms

Xiongfei Liao and Thambipillai Srikanthan
School of Computer Engineering, Nanyang Technological University, Singapore
{liao0016, astsrikan}@ntu.edu.sg

1. Problem and Motivation
SystemC cycle-level simulations can be very slow as SystemC engines, including [1], are single-threaded and can not utilize multicore computers to accelerate simulation performance. Thus, we are motivated to explore techniques to accelerate above simulations on multicore computers.

2. The Proposed Methodology
Our methodology maps simulation of SystemC modules into groups such that each group is executed by a dedicated thread and multiple such threads realize concurrent computations. As the amount of computation per clock phase of individual modules in a typical modeling style is usually too small to run a thread efficiently, we organize multiple modules into a group. Simulation of a group provides its thread a relatively independent workload which can be adjusted by adding (removing) modules into (from) it.

To support multi-threaded simulations without loss of SystemC simulation semantics [2], a single-threaded engine can be transformed into a parallelized engine which keeps the synchronization costs low and maintains thread safety. This engine relieves designers from threads programming and automatically generates threads which are crucial for accelerating simulations. Each clock phase of a multi-threaded simulation is divided into alternate sequential and parallel sections. In parallel sections, threads execute simulations of their groups concurrently. In sequential sections, threads are synchronized to update shared variables or to exchange data. Acceleration is obtained when savings from parallel execution overcome overheads of synchronization aggregately.

Information of groups can be embedded in names of modules. The engine analyzes these names to group modules and other SystemC objects. When multicore or manycore systems are simulated, modules can be manipulated with granularities coarser than module, such as core and tile, since modules in a core or tile are closely coupled.

3. Experiments and Results
The engine of UNISIM [3] [4] is parallelized to simulate a 64-tile NoC based MPSoC whose architecture is similar to the one in [5]. A program for calculating \( \pi \) is run on all tiles for \( 64 \times 1024 \) steps where calculation is distributed evenly to 64 MPI processes. When tiles are grouped with different options (Figure 1), workloads of groups are adjusted. Speedup is defined by \( S_p = \frac{T_1}{T_p} \), where \( T_1 \) is the time of simulation under the single-threaded engine and \( T_p \) is the simulation time under option \( p \). Simulations are completed on an 8-core computer. The highest speedup of up to 2.305X is achieved using 8 cores under option (4) where each group has 8 tiles and all tiles form 8 contiguous areas for groups.

![Figure 1. Options for grouping objects (at tile level)](image-url)

4. Novelty, Contributions and Future Work
Different from previous works [2] [6], our work handles SystemC objects in executables. Thus, it does not modify source code and is suitable for general systems.

Our methodology relies on the formation of groups based on SystemC modules to generate multiple threads which paves the way for accelerating simulations. It is highly scalable due to its being capable of generating the required number of threads depending on numbers of SystemC modules and cores on a simulation host.

We will continue to automate our methodology to obtain optimum speedups by establishing an adaptive technique to minimize workload imbalances among threads.

References