Efficient Co-Simulation of Multicore Systems

Brock-Nannestad, Laust; Karlsson, Sven

Published in:
Proceedings of the Fourth Swedish Workshop on Multicore Computing

Publication date:
2011

Efficient Co-Simulation of Multicore Systems

Laust Brock-Nannestad
DTU Informatics
Technical University of Denmark
s060339@student.dtu.dk

Sven Karlsson
DTU Informatics
Technical University of Denmark
ska@imm.dtu.dk

ABSTRACT

Simulation is an indispensable tool for debugging and verification of multicore systems. However simulation is slow. For complex multicore systems a simulation model will execute several orders of magnitude slower than the actual hardware implementation. We propose a method for capturing the hardware state of a multicore design while it is running on an FPGA. With minimal changes to the design and using only the built-in JTAG programming and debugging facilities, we describe how to transfer the state from an FPGA to a simulator. We also show how the state can be transferred back from the simulator to FPGA. Given that the design runs in real-time on the FPGA, the end result is speed improvements of orders of magnitude over traditional pure software simulation.

1. APPROACH

We propose a method which complements a normal FPGA design work flow. The design on the FPGA is halted and its state transferred to a PC using a so-called *Readback*. We correlate the captured state of memory blocks and registers with the equivalent variables in the HDL model. We then update the HDL model so that it may be started in a simulator at the point in time where the hardware design was halted. Likewise we perform the opposite process, capturing values from the simulator and generating a new *bitstream* for the FPGA so it can be started where the simulator was halted. A set of tools help automate these steps. This iterative process can be used to speed up execution by only employing the slow but detailed simulator when the design is executing at a point of interest.

Similar approaches can be found in other projects. gNOSIS [2] focuses on verification by running a design on an FPGA and in a simulator in parallel, the goal being to find simulation mismatches by comparing their state. ProtoFlex [1] is a hybrid simulation/emulation platform which partially models I/O devices or processors in FPGAs. The FPGA only implements common operations and is backed by a full software model, which requires that the state be transferred between the two models. Where ProtoFlex uses FPGAs to speed up full system simulation, we want to improve the simulation time for generic hardware designs.

2. EXPERIMENTAL RESULTS

We employ two circuits: a small circuit which is used for validation and a larger circuit to estimate real world impact. The larger circuit is roughly equal in size and complexity to a single node in an FPGA based multicore system and is the one being presented here. The Xilinx FPGA ran at 50 MHz.

The execution time of various tasks can be seen in table 1. As one can see, there is a high initial cost for execution on the FPGA, taken up by initial bitstream generation, see *Synthesis, Place & Route*. If the state of the FPGA is captured after less than 50,000,000 cycles, corresponding to *one* second of execution on the FPGA, simulation in ModelSim is faster. For longer simulation times, executing on the FPGA is clearly beneficial. Once the initial bitstream has been generated, subsequent modifications are considerably faster as they only modify register and memory values and not the layout of the design.

Task	Duration (sec.)
Synthesis, Place & Route	309
Run 50,000,000 cycles on FPGA	1
Readback and Parse captured state	37
Simulate 50,000,000 cycles	355
Modification of bitstream	33

Table 1: Execution times for steps in the workflow.

3. CONCLUSION AND FUTURE WORK

We have found our approach useful when a long simulation time is required. The proposed workflow incurs a high initial time penalty, especially due to synthesis and mapping, but these only need to execute as part of the first iteration and subsequently performance improves significantly. For our test circuit, execution on the FPGA was found to be over 300 times faster than in the simulator. While we have shown the our approach is feasible, there are still issues to tackle. Certain HDL constructs leave the concrete implementation up to the synthesizer, making it difficult to automatically map their state back into the simulation model. In addition, we only consider state inside the FPGA and not devices attached to it, e.g. external memories.

4. REFERENCES

[1] E. S. Chung, J. C. Hoe, and B. Falsafi. Protoflex: Co-simulation for component-wise fpga emulator development. In *2nd Workshop on Architecture Research using FPGA Platforms*, 2006.

[2] A. Khan, R. N. Pittman, and A. Forin. gnosis: A board-level debugging and verification tool. In *International Conference on Reconfigurable Computing and FPGAs*, 2010.