View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Online Research Database In Technology

Technical University of Denmark

Effcient Co-Simulation of Multicore Systems

Brock-Nannestad, Laust; Passas, Stavros; Karlsson, Sven

Publication date:
2011

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):
Brock-Nannestad, L., Passas, S., & Karlsson, S. (2011). Effcient Co-Simulation of Multicore Systems. Poster session presented at 4th Swedish Workshop on Multicore Computing, Linköping, Sweden.

DTU Library
Technical Information Center of Denmark

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

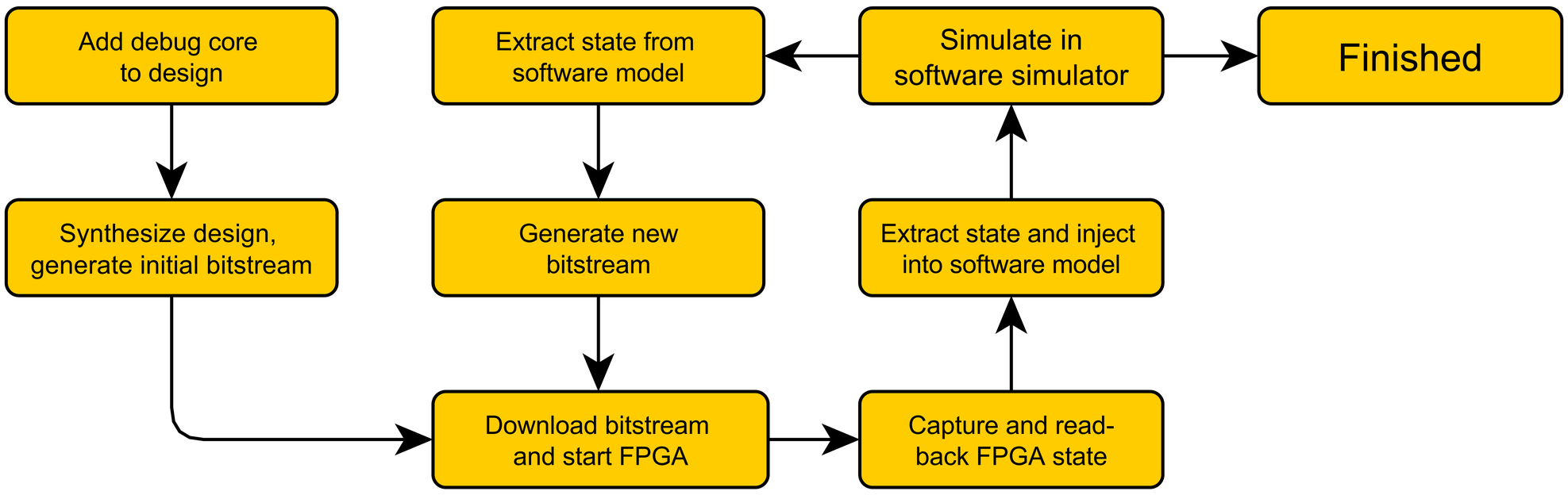
- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Efficient Co-Simulation of Multicore Systems

Laust Brock-Nannestad, Stavros Passas and Sven Karlsson
Technical University of Denmark

Workflow



Motivation

- Simulation of hardware helps debugging by exposing a hardware design's inner operation
- Simulation speed significantly slower than the actual hardware implementation
- Can we simulate at hardware speeds?
  - by default use FPGA to simulate at high speed
  - switch to a software simulator where visibility is required

Contributions

- A method for debugging and co-simulation of hardware designs at close to real-time hardware speeds
- A mechanism for transparently capturing and moving state between FPGA and software simulator
- Performance evaluation and validation using two circuits

Approach

- We capture the state of the FPGA and transfer it to our software
- The state is injected into a software simulation model and simulation is resumed from the point of capture
- We perform the reverse process; transfer state from software simulation model back into the FPGA
- The hardware design is augmented with a small "capture" core which initiates a capture
- We rely on capture and readback functionality implemented by FPGA

Conclusions

- We implemented a method for co-simulation of hardware designs using software simulator and FPGA
- We validated our approach using a small circuit in a single FPGA
- Our evaluation of performance shows co-simulation is beneficial for long simulation times

Future Work

- Consider state of components external to the FPGA – e.g. memories
- Handle all complex HDL constructs such as state encodings chosen by the synthesis tool
- Allow for multiple FPGAs

Experimental setup

- Experiments were carried out on a Xilinx Spartan-3AN FPGA @ 50 MHz
- ModelSim 6.5 was used for simulations
- We used generic low-cost JTAG dongle for all communication with FPGA
- Two hardware designs were implemented using VHDL
  - A small design used to validate correctness
  - A large design to evaluate performance

Resource	HDL Design	
	Validation	Evaluation
Slice Flip-flops	64	381
LUTs	42	1691
Block RAMs	1	16

Table: Resource utilization for the two FPGA designs.

Results

- To evaluate performance we compare execution time of 50,000,000 cycles in the software simulator and on FPGA
  - We see that hardware is more than 300 times faster than simulation
- We also measure the time from hardware description to running simulation and from hardware description to running FPGA
- Finally, we measure the time to transfer the state between hardware and software, when executing our workflow

Task	Execution time (seconds)	
	Validation	Evaluation
Simulate 50,000,000 cycles	104	355
Run 50,000,000 cycles on FPGA	1	1
Generate initial bitstream	58	309
Download of bitstream	1	1
Readback and parse	33	37
Update simulation model	4	N/A
Compile in ModelSim	1	4
Regeneration of bitstream	13	33

Table: Execution times for baseline and steps in the workflow.

- Generating initial bitstream is very slow due to synthesis and mapping
  - It only has to take place **once**
- Subsequent modifications to bitstream are fast as we only alter initial states of flip-flops and block memories
- For short simulation times, the faster compile time of the software simulator is beneficial, despite slow simulation speed
- Regular software simulation is 300 times slower than execution on the FPGA — one second on FPGA takes 355 seconds to simulate