Towards Certifiable Loop Pipelining Algorithms in Behavioral Synthesis

Disha Puri

21st Oct 2013
Computer Science Department
Portland State University
Behavioral Synthesis

High level specification of hardware design (in C, C++, System C, etc.)

Compiler Transformations
(e.g., dead code elimination)

Scheduling Transformations
(e.g., Loop pipelining)

Resource Allocation and Control Synthesis

Behavioral synthesis tool
(e.g., AutoESL, CtoSilicon, Cynthesizer)

Implementation in hardware description language (in VHDL, Verilog, etc.)
Motivation behind Loop Pipelining

Execution order before pipelining

Execution order after pipelining

"Pipeline interval is 1"
Goal: A Certifiable Loop Pipelining Algo

Sequential Execution

Certifiable Pipelining Algo

Pipeline Reference Model
Challenges in Loop Pipelining \# 1

Problem: \( a \) will be overwritten by \( X_2 \) before being read by \( Z_1 \)
Challenges in Loop Pipelining #2

Problem: Attempt to read \( i \) before it has been written
Challenges in Loop Pipelining # 3

Problem: If the cond is true, $Z_1$ is never executed
Approach

Algorithm:
• Define a generic framework of pipelining primitives to handle these challenges
• Develop a simple pipelining algorithm using a combination of these primitives to generate reference loop pipelines

Certification:
• Prove using ACL2 theorem prover that these primitives are correct
• Prove using ACL2 theorem prover that the algorithm correctly applies these primitives.
Thanks!