Chair of Electronic Design Automation
Department of Computer Engineering
School of Computation, Information and Technology
Technical University of Munich

Flexible Generation of Fast and Accurate Software Performance Simulators From Compact Processor Descriptions

Conrad Foik, Robert Kunzelmann*, Daniel Mueller-Gritschneder** and Ulf Schlichtmann

Design Space Exploration

Simulation-based design space exploration to find **optimal** application-specific solutions.

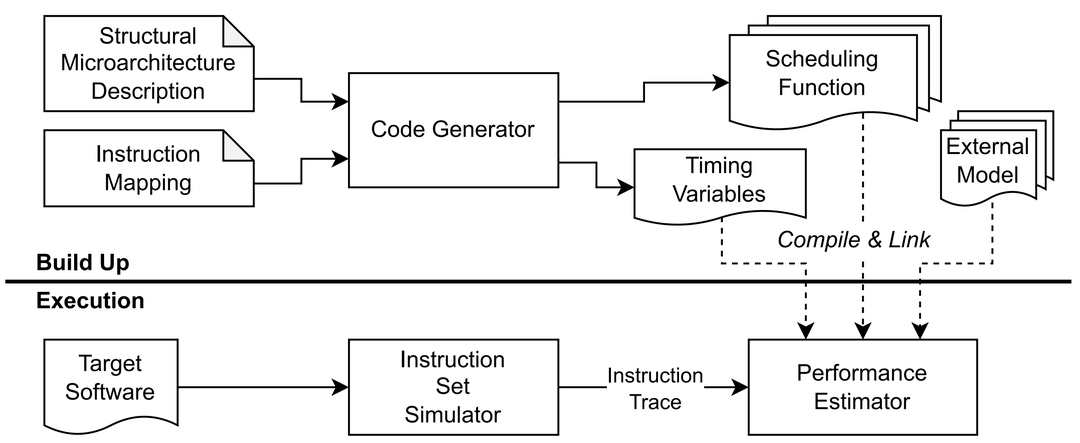
Requirements:

- **Accurate** performance estimates to evaluate solutions
- High simulation **speed** to run exhaustive simulations
- **Flexibility** to adapt and evaluate new microarchitecture variants

Current Approaches:

- RTL simulation: Too slow for exhaustive simulations
- Instruction Set Simulators (ISS): Inaccurate, no performance variations visible
- Performance Simulators: Typically inflexible, focus on single target processor

Methodology



Code Generator:

- Compact processor description as input (**Flexibility**)
- Constraint-based intermediate representation supporting single- & multi-issue concepts (**Accuracy**)

Generated Items:

- *Scheduling functions*: Model instruction type specific timing behavior
- *Timing variables*: Represent availability of processor's components

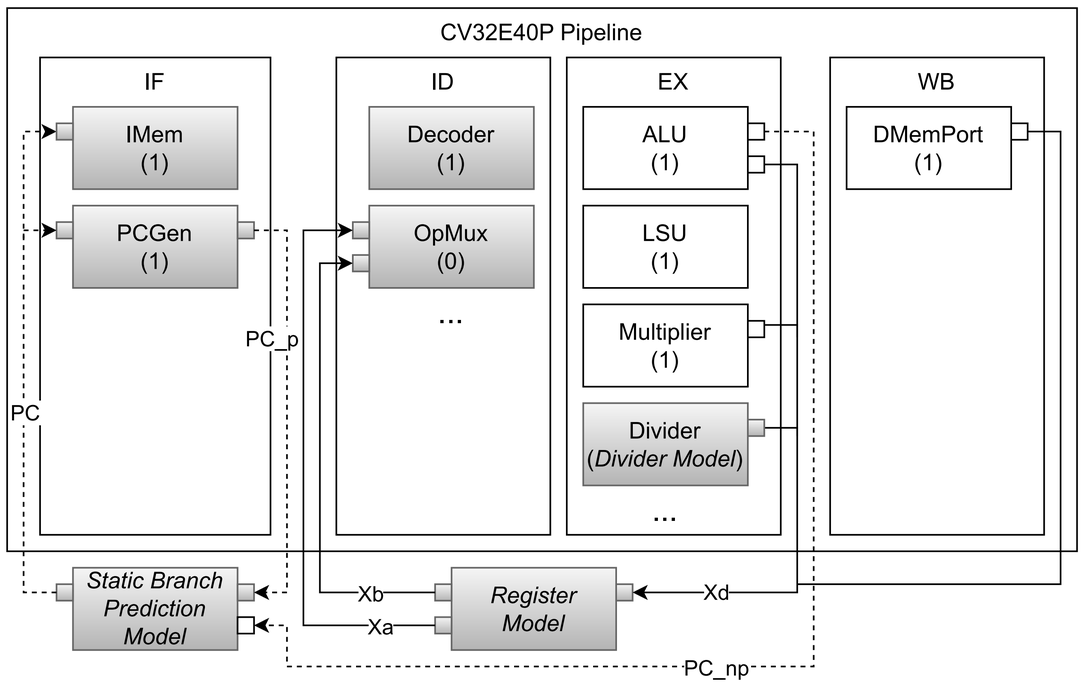
Performance Estimator:

- Based on ISS instruction trace (**Portability**)
- Once per instruction: Updated by pre-compiled scheduling function (**Speed**)

Input Description

Compact Processor Description:

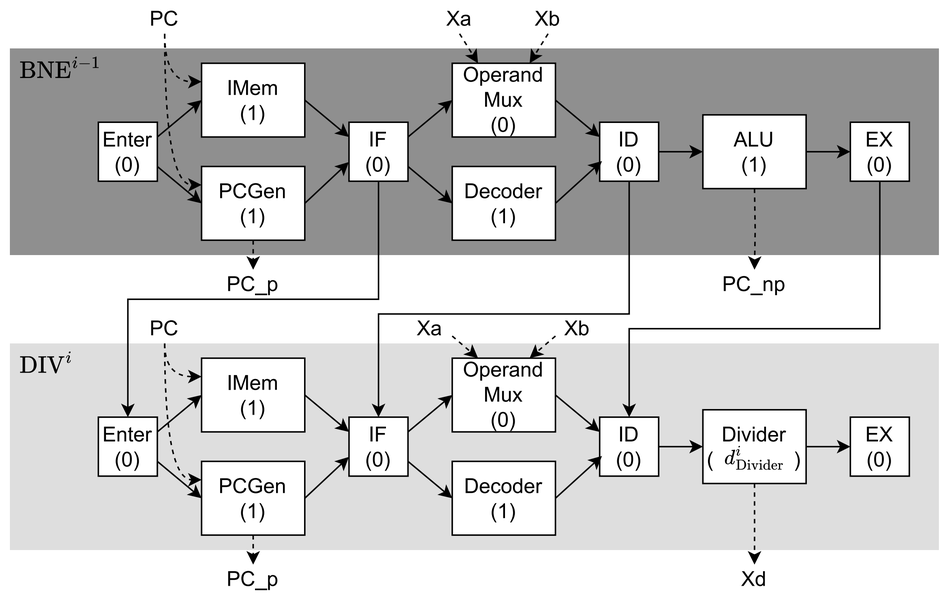
- **Structural** (non-functional) description of microarchitecture
- **Instruction mapping** assigning instructions to resources



Code Generation

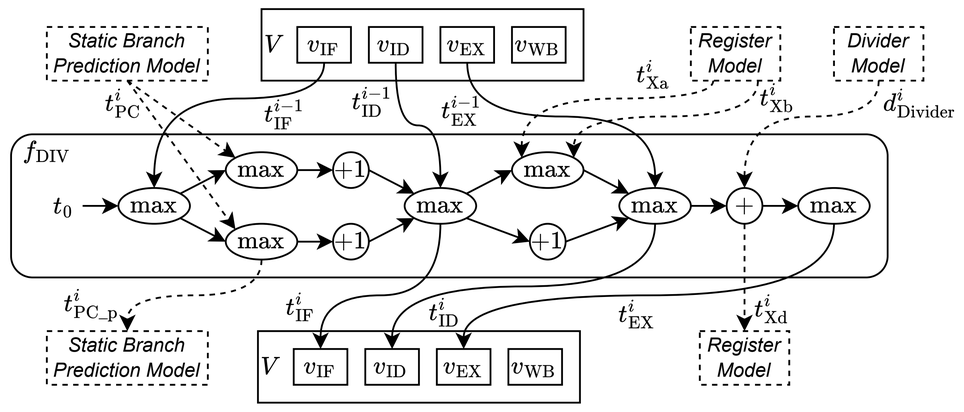
Timing Constraints: Automatically derived from input description

- **Instruction-internal** dependencies (e.g. $t^i_{IF} \geq t^i_{IMem}$, $t^i_{ALU} \geq t^i_{ID} + 1$)
- **Cross-instruction** dependencies (e.g. $t^i_{IF} \geq t^{i-1}_{ID}$)
- **Dynamic** dependencies (e.g. $t^i_{PCGen} \geq t^i_{PC} + 1$)



ASAP Scheduling:

- Simple algebraic *scheduling functions* (max, +)
- *Timing variables* hold timing state and "decouple" calculations



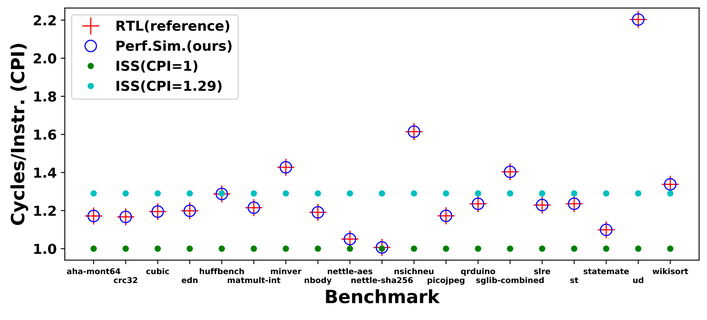
Multi-Issue Concepts:

- Constraints: Expand concept to cover advanced features
  - Stages may hold more than one instruction at a time (e.g. $t^i_{IS} \geq t^{i-8}_{EX}$)
  - Resources in stages may be organized into parallel *sub-pipelines*
- Scheduling: Use queue-like timing variables to hold values provided by multiple past instructions

Experimental Results

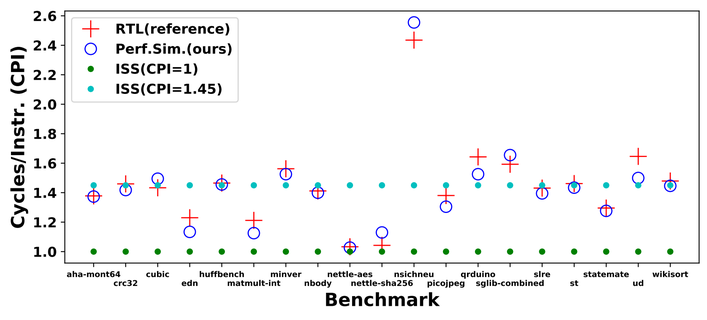
CV32E40P [1]:

- Avg. overall error: **0.00148%**
- Avg. error per instr.: **0.00013 cycles**
- Avg. simulation speed: **23.8 million instr./s**



CVA6 [2]:

- Avg. overall error: **3.88%**
- Avg. error per instr.: **0.13 cycles**
- Avg. simulation speed: **15.5 million instr./s**



[1] OpenHW Group. "CV32E40P user manual" 2023 Available: https://docs.openhwgroup.org/projects/cv32e40p-user-manual/
[2] OpenHW Group. "CVA6 user manual" 2023 Available: https://docs.openhwgroup.org/projects/cva6-user-manual/

* Mr. Kunzelmann is now with Infineon Technologies
** Prof. Mueller-Gritschneder is now with the Institute of Computer Engineering, TU Wien

This work was partly supported by the German Federal Ministry of Education and Research (BMBF) and ITEA within the project GenerIoT under contract no. 01IS22084G.

SPONSORED BY THE
Federal Ministry of Education and Research

Contact:
conrad.foik@tum.de
daniel.mueller-gritschneder@tuwien.ac.at

Open Source:
https://github.com/tum-ei-eda/PerformanceSimulation_workspace