Piparazzi: A Test Program Generator for Micro-architecture Flow Verification

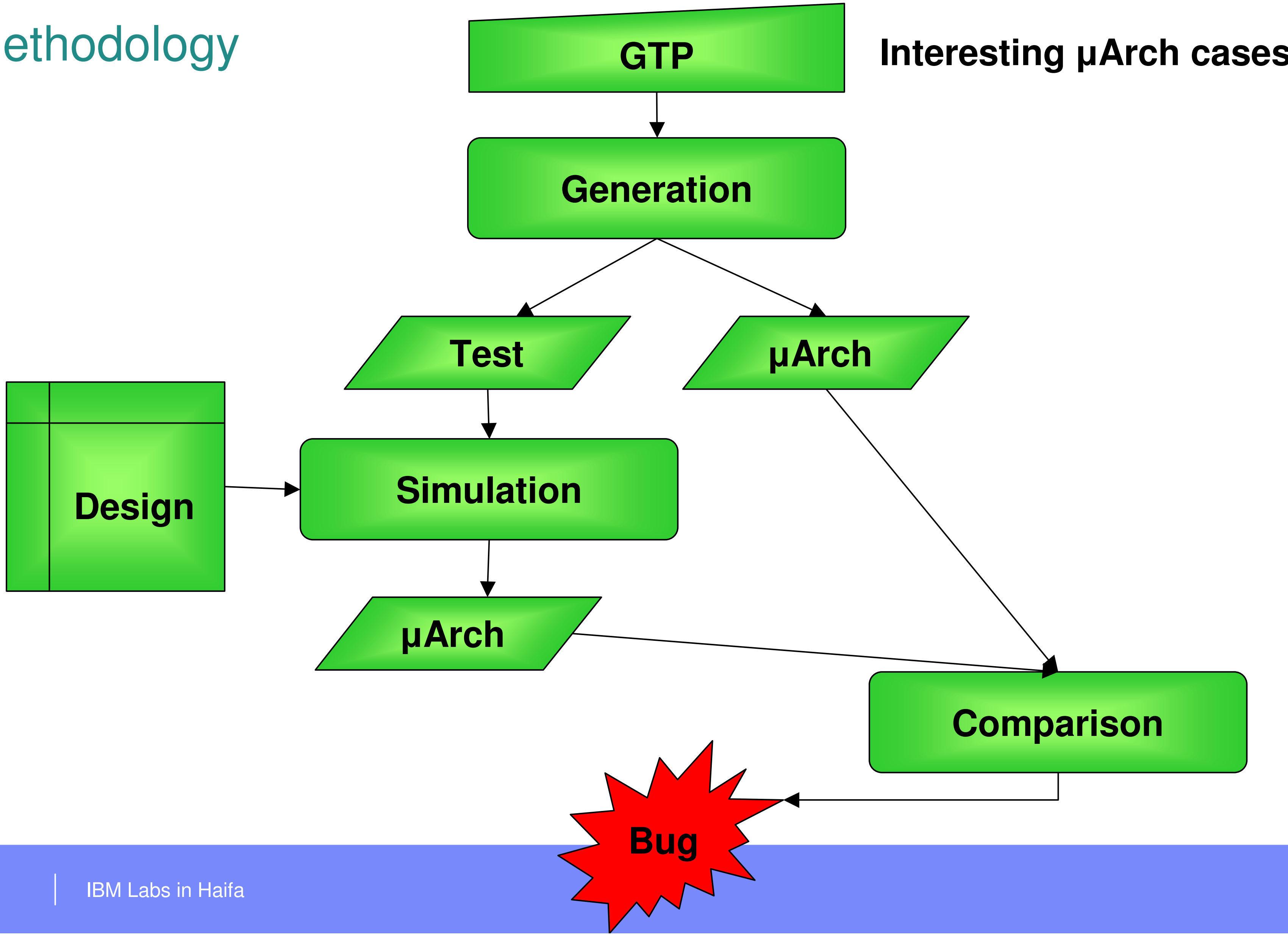
Eyal Bin

IBM Labs in Haifa

Outline

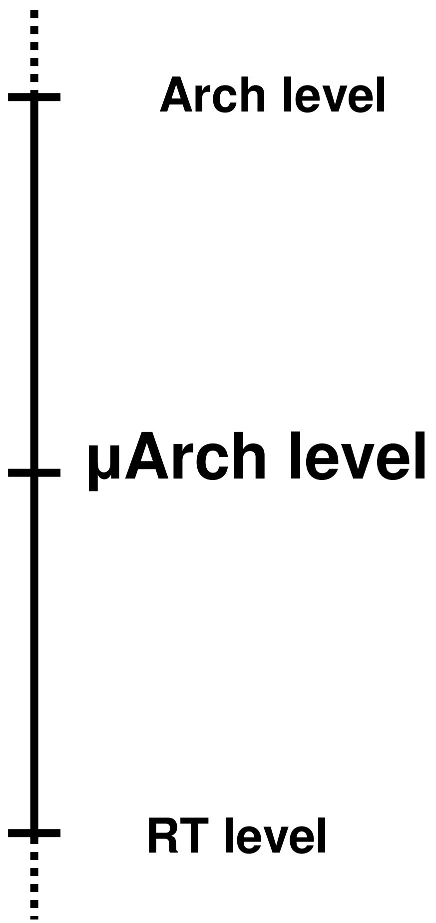
- Methodology
- Types of bugs
- Generic Test Plan (GTP)
- Generation
- µArch comparison
- Results and summary

Methodology



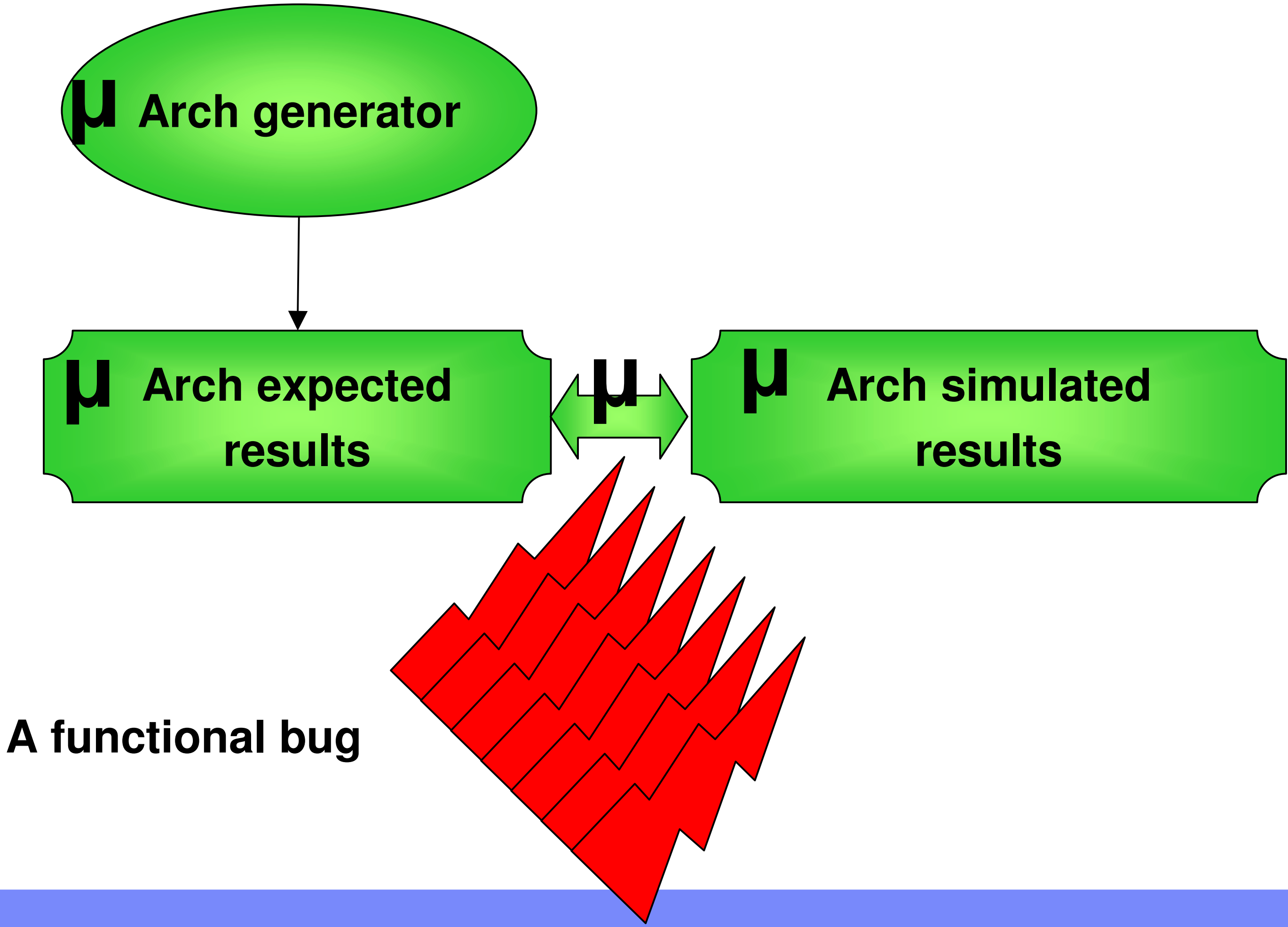
Micro-architecture verification

- Microprocessors
- Core level verification



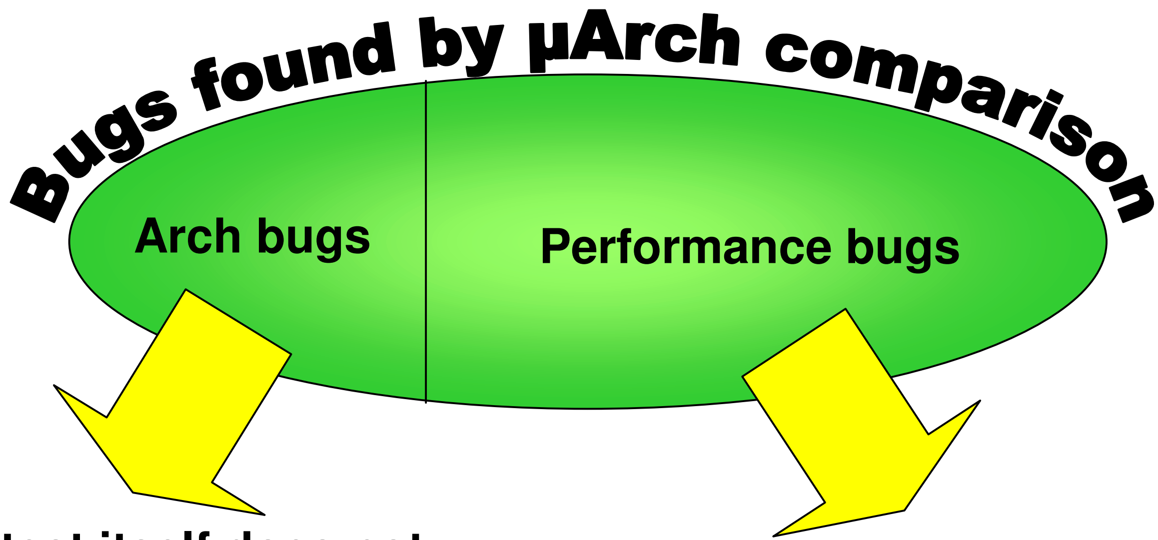
There are more bugs in the µArch level than in the Arch level

Different abstraction levels to find a **functional** bug using expected results



Classification of bugs found by **µArch** comparison

- **Bugs found by µArch** discrepancy: timing, mechanism, location



While the test itself does not show a functional problem, it can be inferred from an analysis of the faulty µArch behavior

Cannot be found by arch comparison

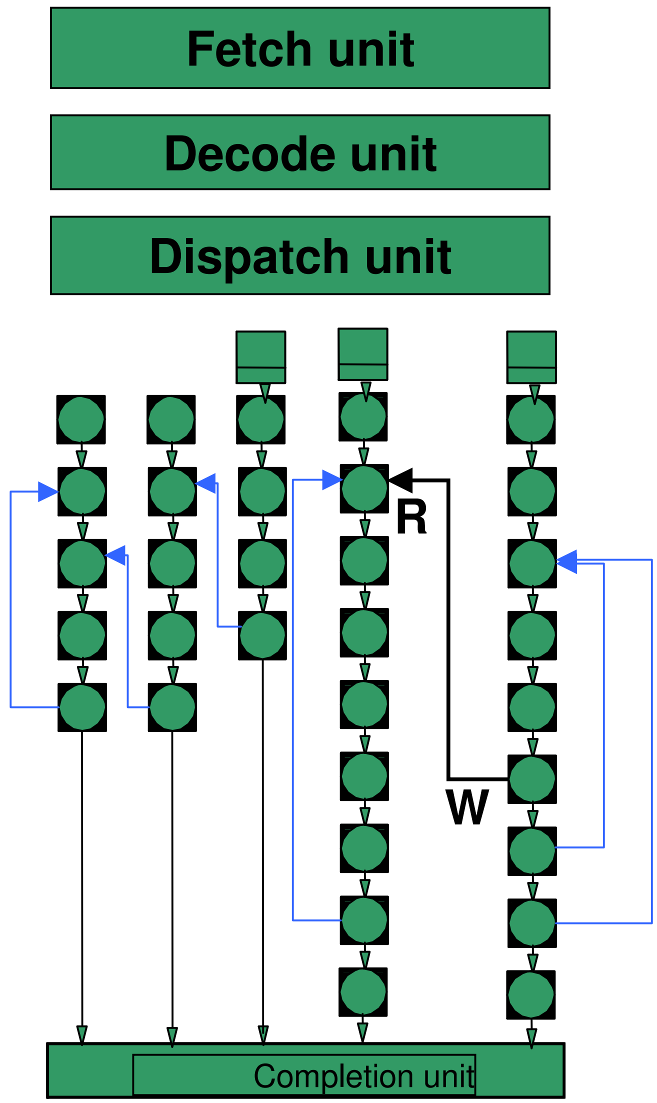
Generic Test Plan – processor independent test plan

- Coverage models for different units in the processor
- Cross-product coverage models
- Coverage models for resources and instructions
- Knowledge reuse

Even though microprocessors have a different µArch , we define a generic µArch test plan covering **specific** µArch behavior

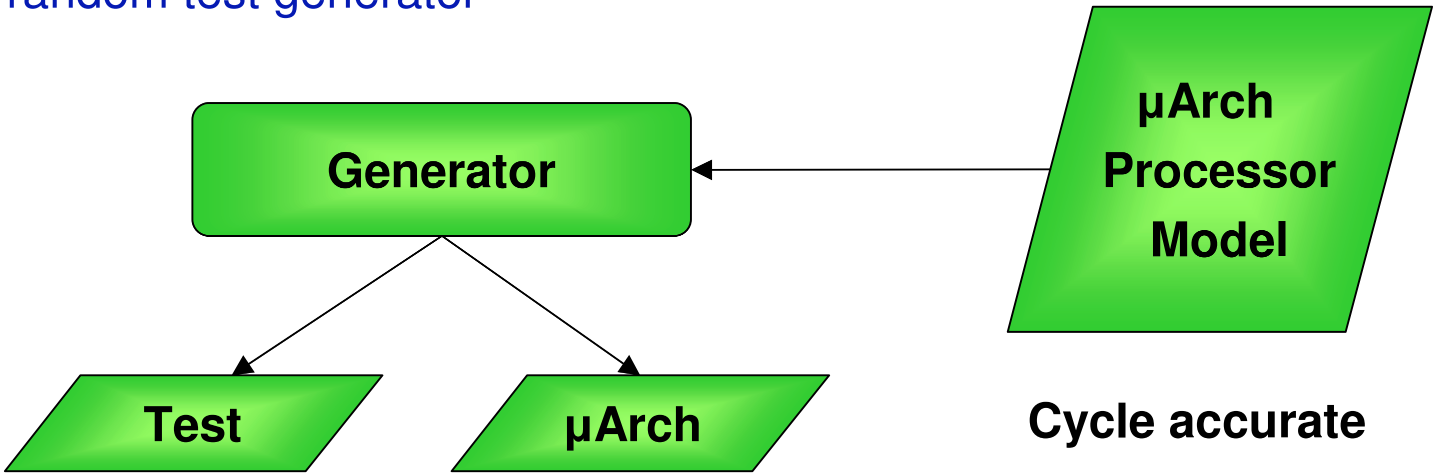
An example

For each pipeline in the processor
  For each stage in the pipeline
    For each stall reason in the stage
      Generate a test that creates the stall



Generator

- Model-based (processor independent)
- Based on CSP
- Coverage by generation
- Two main outputs
- Slower than random test generator



μArch model

- Enables the definition of different microprocessors (Out of order, Pipelines, Super scalar, Multithreading)
- A cycle accurate model

Modeling ≡ Configuration of predefined building blocks

Types of building blocks

Hardware

Fetch
Dispatcher
Pipeline
Cache
Branch prediction
Queue

Mechanisms

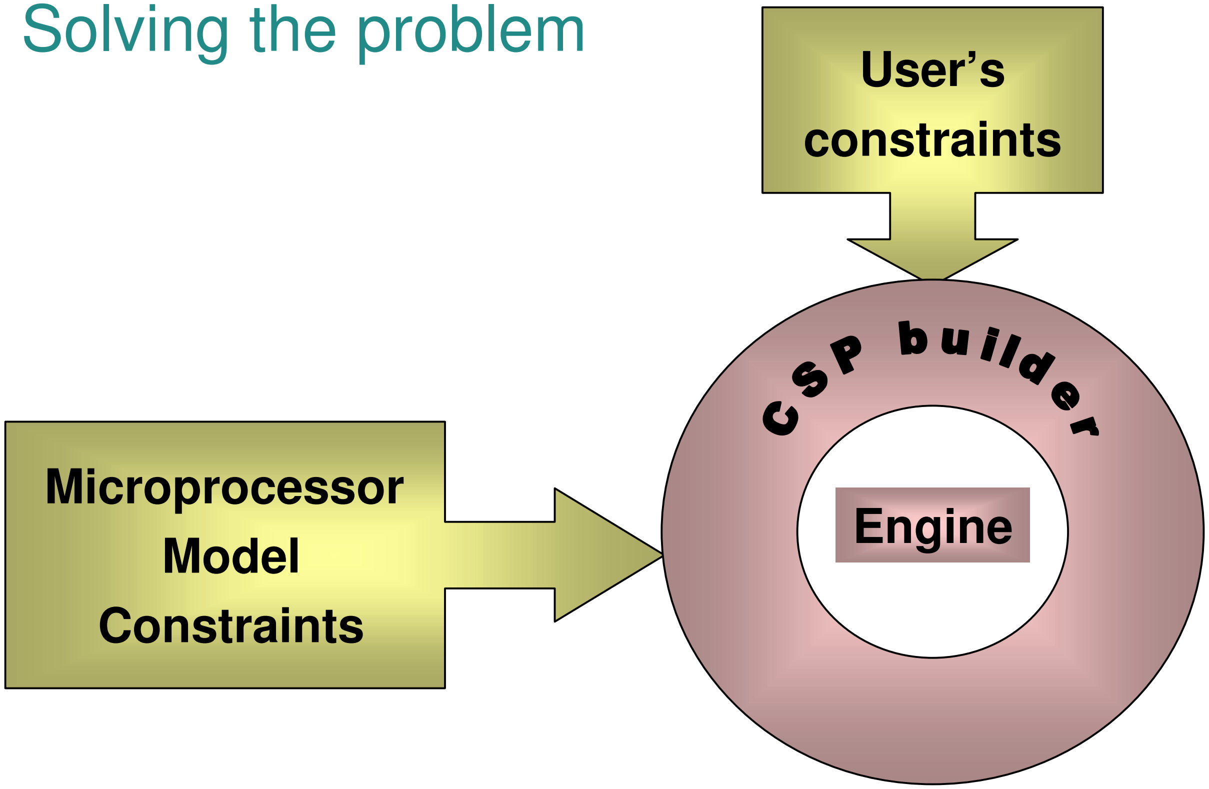
Flush
Forwarding
Interrupt
Splitting instruction

Instructions

add
sub
load
store
sync

The model:
Each BB may have variables, constraints, parameters, and BBs

Solving the problem

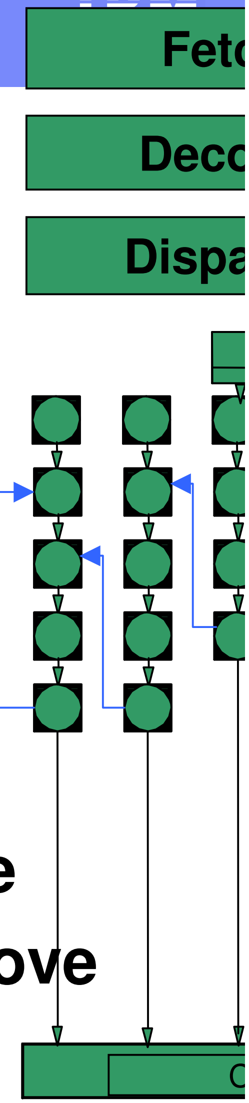


CSP Builder

CSP builder

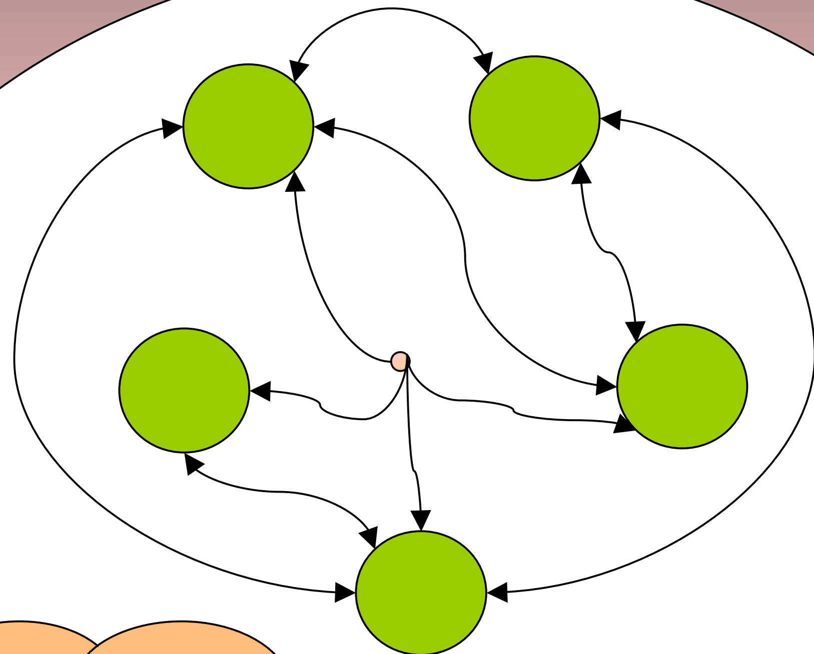
- Each building block creates its modeled CSP variables
- Each building block inserts its constraints:
  - Design independent constraints
  - Design dependent constraints (modeled)
- User's constraints are inserted

Example for constraints: pipeline stage building block



- Design independent constraints:
  - EXIT = ENTRY + TOTAL_STALL
  - TOTAL_STALL = SELF_STALL+…
- Design dependent constraints:
  - The conditions and amounts for an instruction to be stalled in the stage (SELF_STALL)
  - The rules for grouping instructions in the stage
  - The conditions and timing to stall the stage above this one based on the instruction in this stage

The Engine



Solution: Each of the required variables reaches a single value. The constraints are satisfied.

Alg': A dedicated CSP random solver based on MAC (AC3)

Maintain Arc Consistency AC3

1. Bring every arc to a consistent state (if empty set -> backtrack)
2. Choose a variable (if none –> success)
3. Choose a Value (if none –> backtrack (back jumping))
4. Go to 1

A = B + C

A = {~~1~~, ~~2~~, ~~3~~, ~~4~~, ~~5~~, 6}

B = {3, ~~7~~}

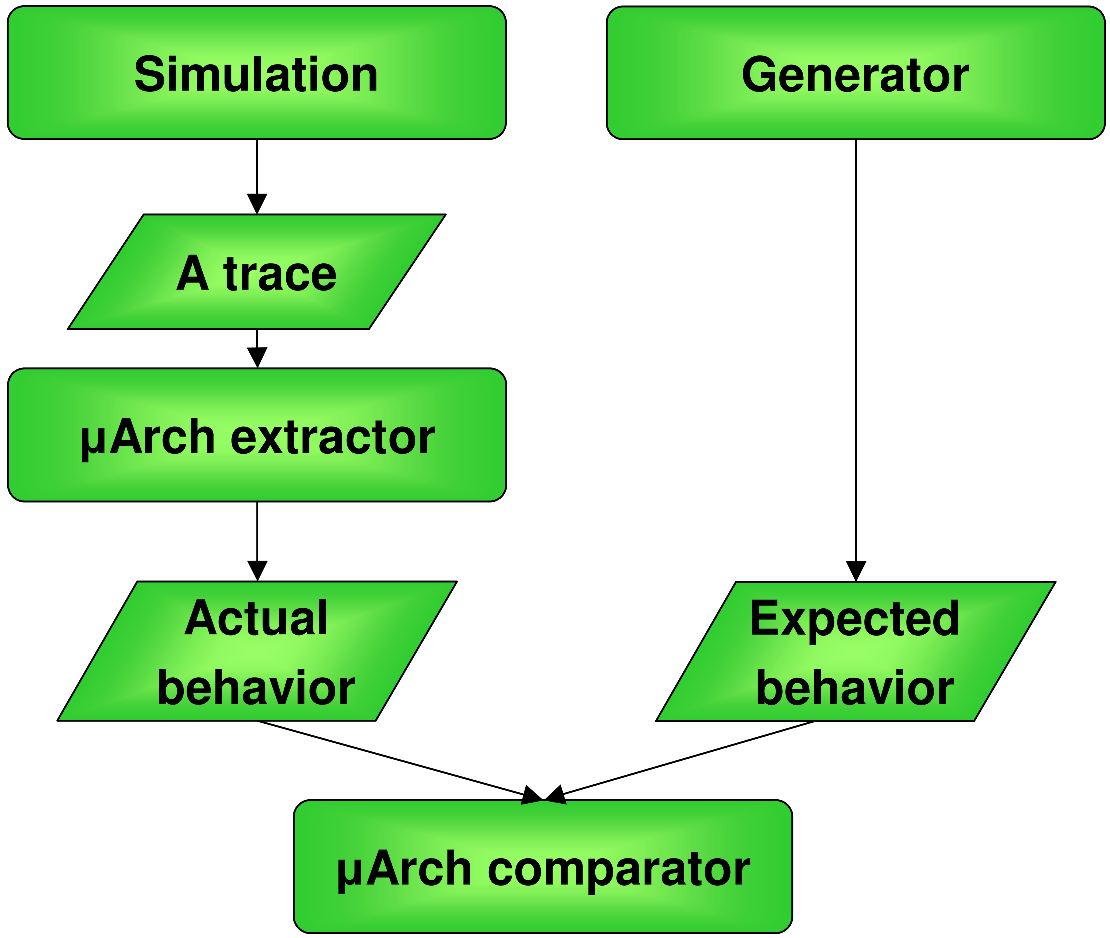
C = {~~0~~, 3}

Specific issues for big and intricate CSPs

- Insert constraints on-the-fly just when necessary
- Different techniques to improve consistency for better success rate
- Domain specific heuristics for variable ordering
- Massive sharing of sub-constraints
- Domain specific redundant constraints

More backtracks **More time on consistency**

µArch comparison



Results

- Working with mainstream high-end IBM processors
- We found many bugs; most of them by using µArch comparison
- About ¾ of the bugs are performance bugs
- Experiment results: Many interesting µArch events have a very low chance of being hit by random test generator (non-µArch)
- It is usually not easy to find a test that reveals the architectural bug based on a µArch discrepancy

Summary

- µArch comparison is useful for finding both performance and Arch bugs
- µArch input language for a generator is important for reaching interesting µArch states
- Bugs have difficulty remaining underground when covering a comprehensive µArch test plan

"Language shapes the way we think,

and determines what we can think about" - B.L.Whorf