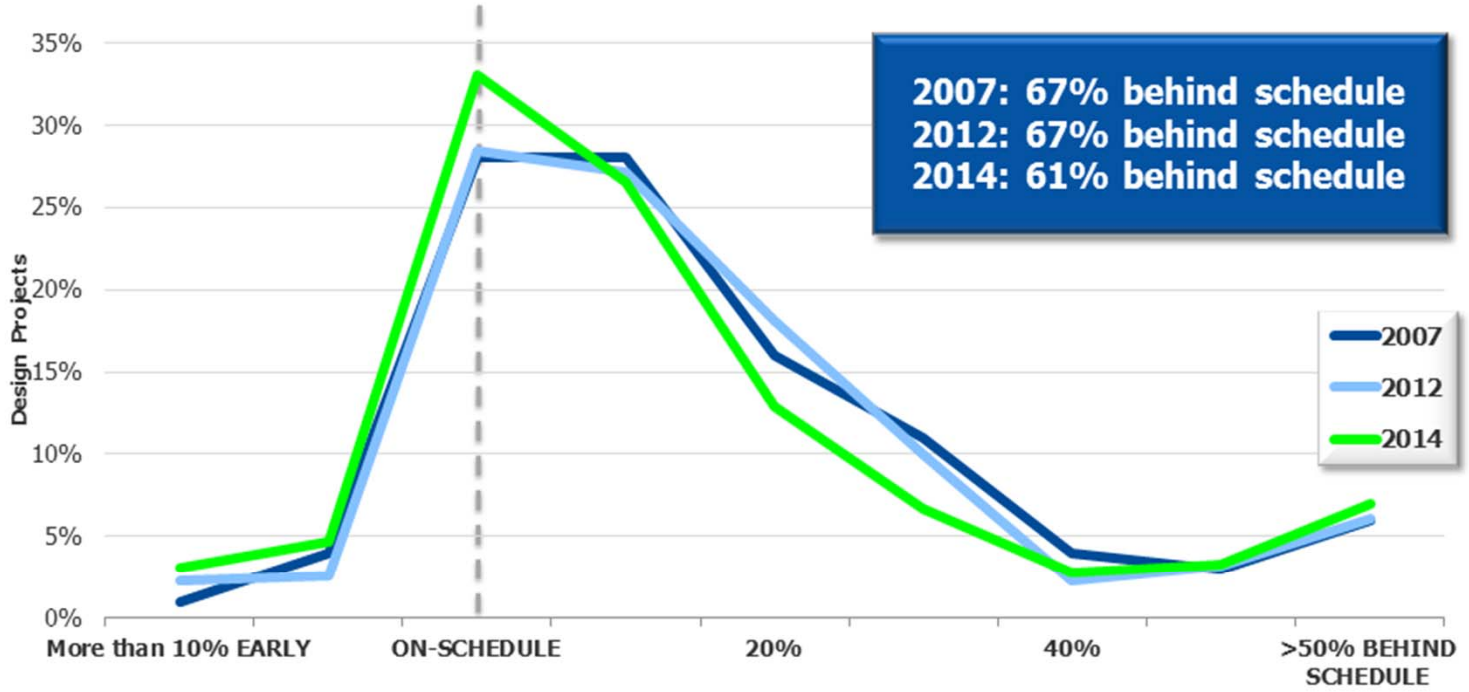
Accelerating Microprocessor Silicon Validation by Exposing ISA Diversity

Rong Xu

Zixin Wang

Ba

Many Projects Miss Schedule



Actual ASIC-IC design completion compared to project's original schedule

Ahead of schedule | **Behind Schedule**

Source: Wilson Research Group and Mentor Graphics, 2014 Functional Verification Study

Aggressive technology scaling and extreme chip integration significantly increase the complexity of microprocessor

- Infeasible to exhaustively test when simulation

Pressure on validation team to deliver correct design to market on time is higher than ever.

- 50% of microprocessor chips require extra unplanned tape-out

A billion-dollar mistake: Intel recalls a supporting chip for popular Sandy Bridge platform

DEAN TAKAHASHI JANUARY 31, 2011 8:26 AM

TAGS: COUGAR POINT, SANDY BRIDGE

Intel made a big deal at the recent Consumer Electronics Show about how its Sandy Bridge combination graphics-microprocessor chip has been one of its most successful in history. But it spoke too soon. Now the world's biggest chip maker said it has discovered a design flaw in the chip's companion chip set, forcing a production delay that will cost it $1 billion in lost revenues and replacement costs.

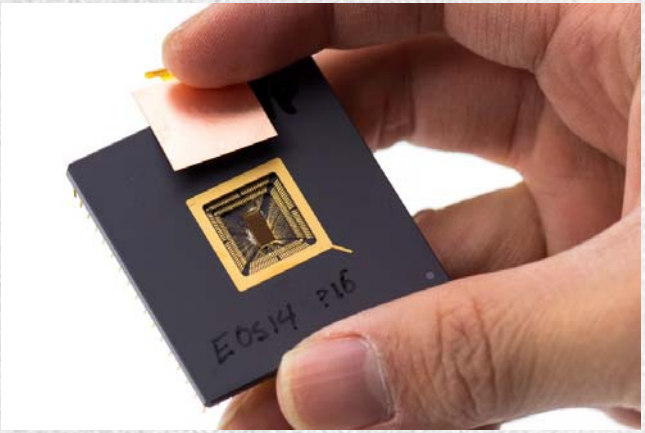
The delay could derail the shipment schedules for more than 500 computers using the Sandy Bridge processor, which combines graphics and a microprocessor in a single piece of silicon. That's going to be bad for the PC industry, since Intel can't supply millions of new chips overnight. That could give an advantage to rival Advanced Micro Devices, which is shipping a rival combo processor under the Fusion platform name.

Intel said it will reduce its revenue target for the first quarter by $300 million and incur repair and replacement costs of $700 million. That's the biggest quality problem since the Pentium bug hit Intel in 1994, when Intel recalled a math-

Number of Required Spins

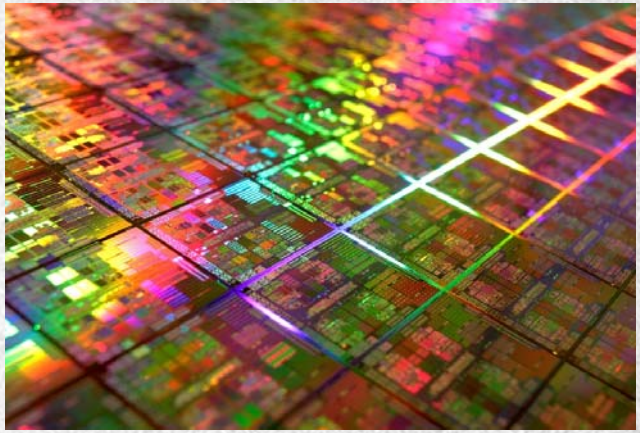
Source: Wilson Research Group and Mentor Graphics, 2014 Functional Verification Study

Prototype to Volume Production

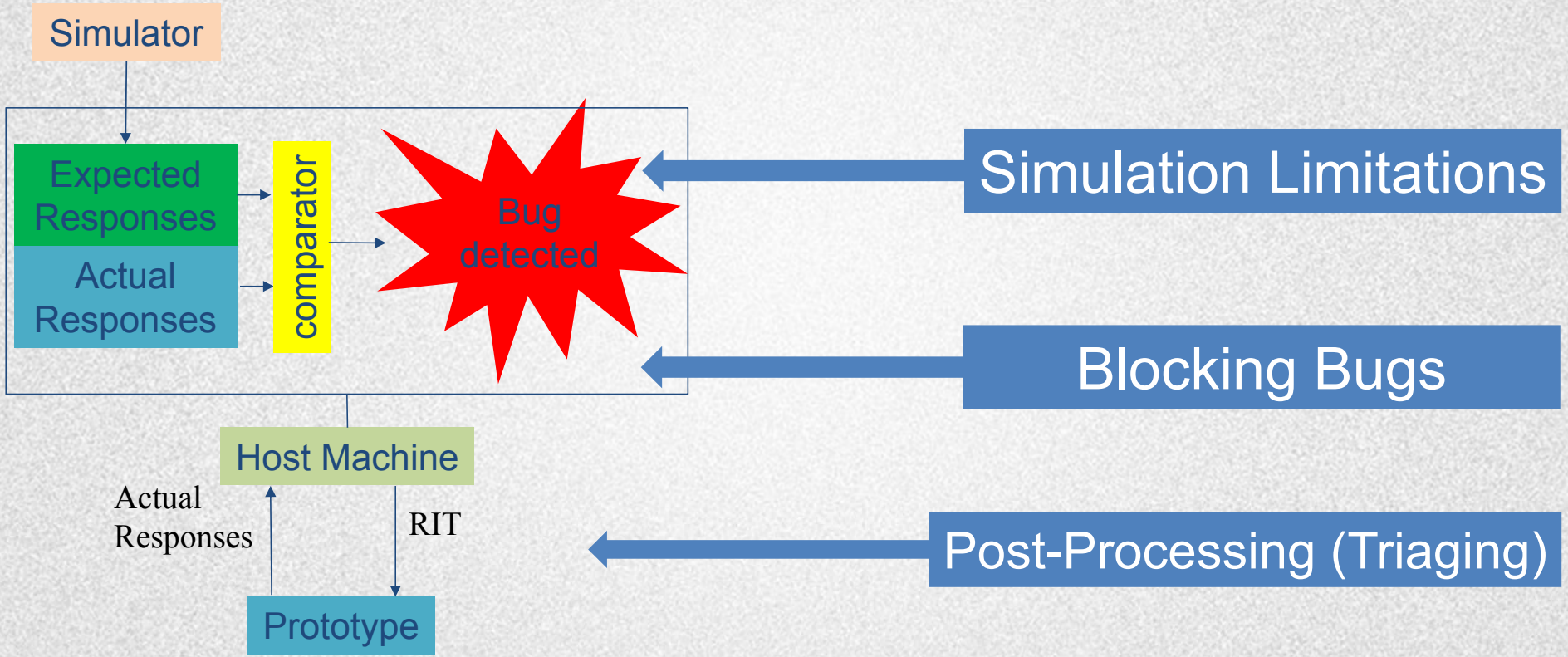


Effective post-silicon validation needed to eliminate bugs before volume production

Random instruction tests(RIT) contribute tremendously to the detection of design bugs



Challenges


Simulator
Expected Responses
Actual Responses
comparator
Bug detected
Host Machine
Actual Responses
RIT
Prototype
Simulation Limitations
Blocking Bugs
Post-Processing (Triaging)

ISA Diversity

Concept: Operations of an ISA can be performed equivalently in more than one different way.

Benefit: Same operation in different ways produce identical results but activate different logic paths

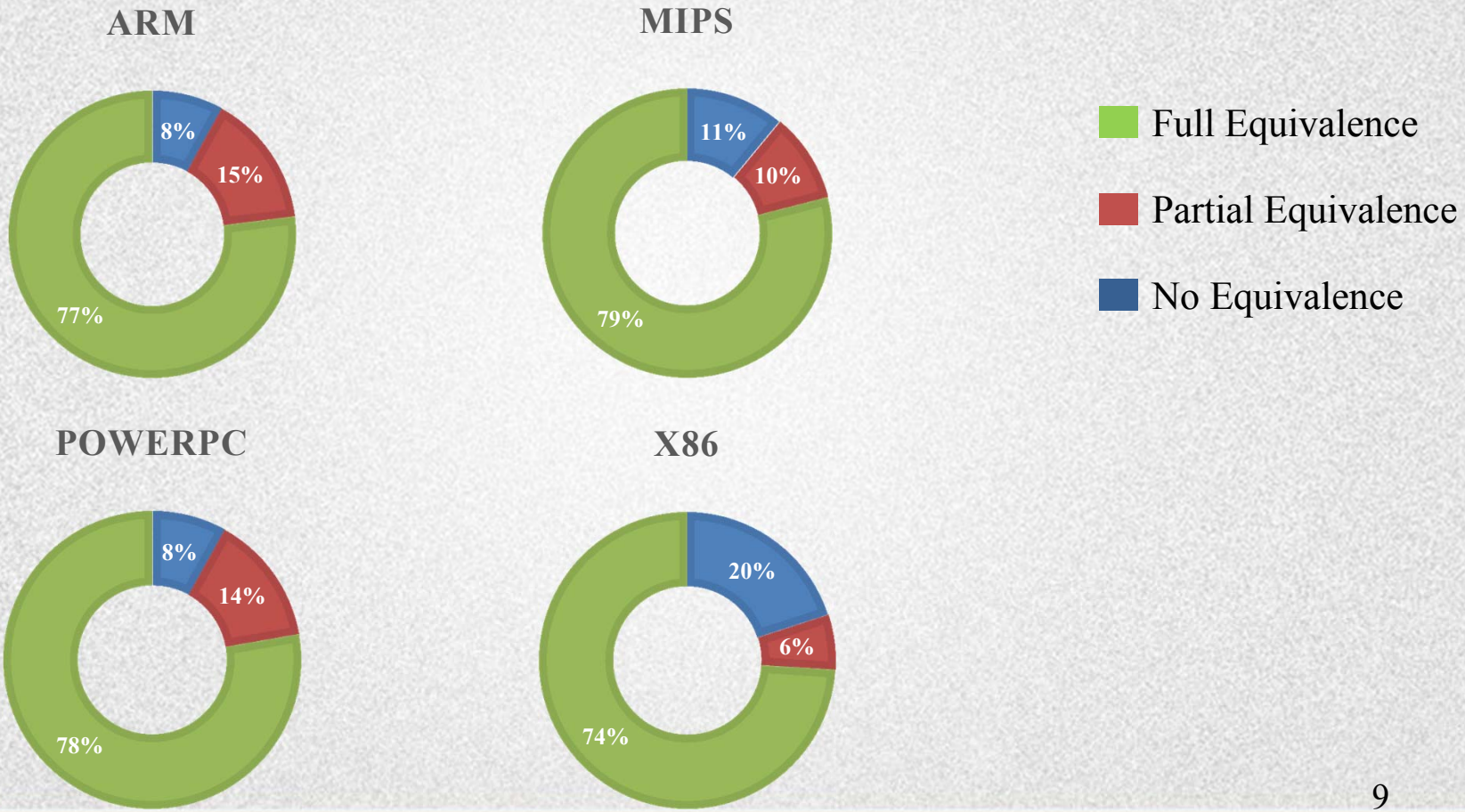
Application: Enable bug detection by comparing results of equivalent instructions (self-checking)

Statistics: In major ISAs, more than 75% instructions can be replaced with equivalent instructions

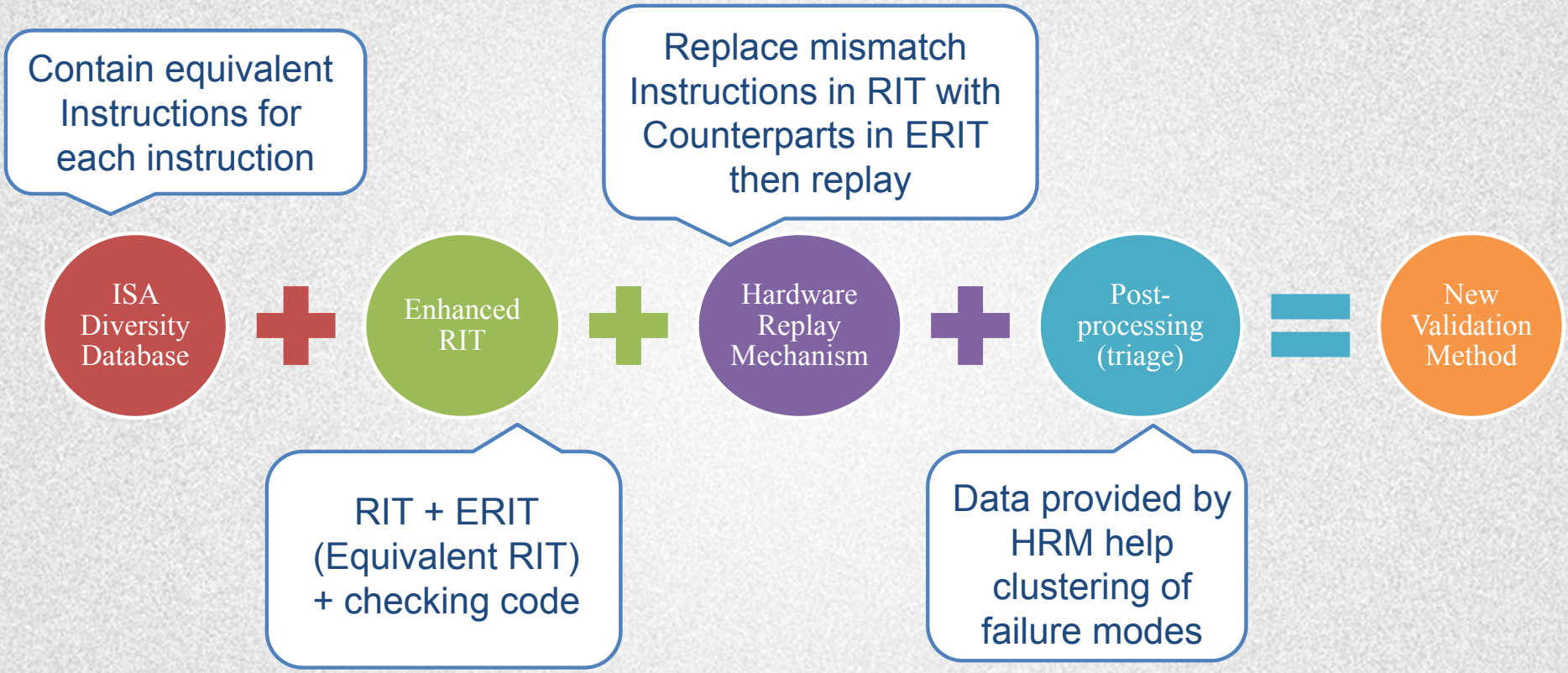
Example: MIPS ISA diversity

Original Instruction	Equivalent Sequence	Description
lw RA, addr(RB) *Load word*	**lhu RA, addr(RB)** **lhu RC, (addr+2)(RB)** **sll RC, RC, 16** **or RA, RA, RC**	Execute 2 load halfword unsigned instructions and places the second halfword to upper bytes.

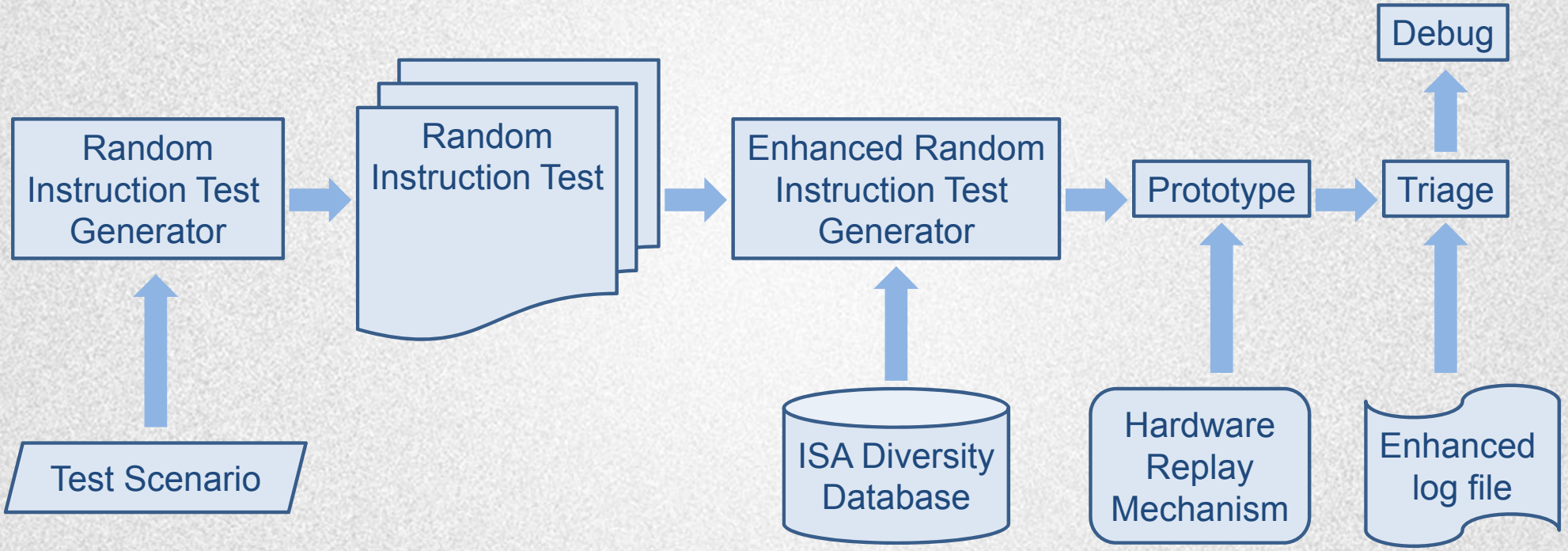
Statistics


ARM
8%
15%
77%
MIPS
11%
10%
79%
POWERPC
8%
14%
78%
X86
20%
6%
74%
Full Equivalence
Partial Equivalence
No Equivalence

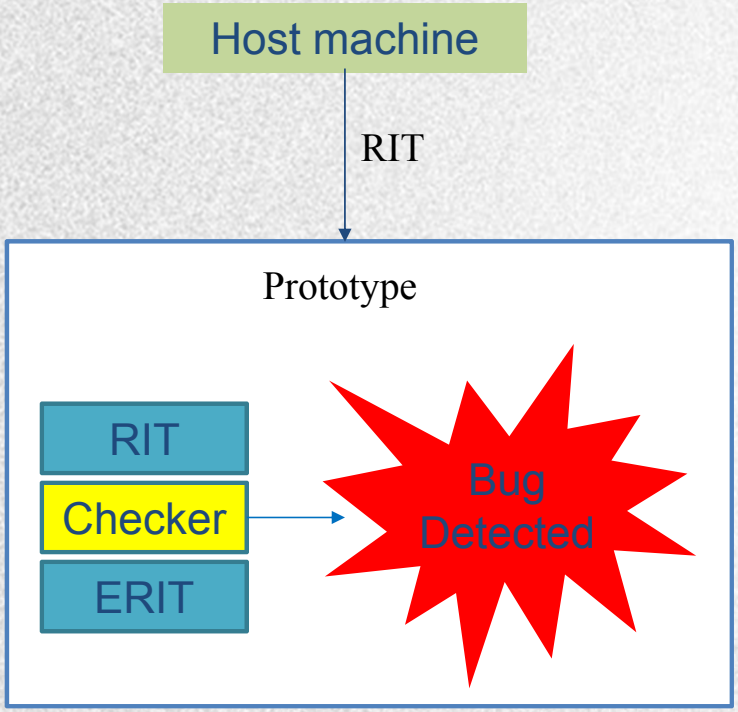
New Validation Method



Framework


Random Instruction Test Generator
Random Instruction Test
Enhanced Random Instruction Test Generator
Prototype
Triage
Debug
Test Scenario
ISA Diversity Database
Hardware Replay Mechanism
Enhanced log file

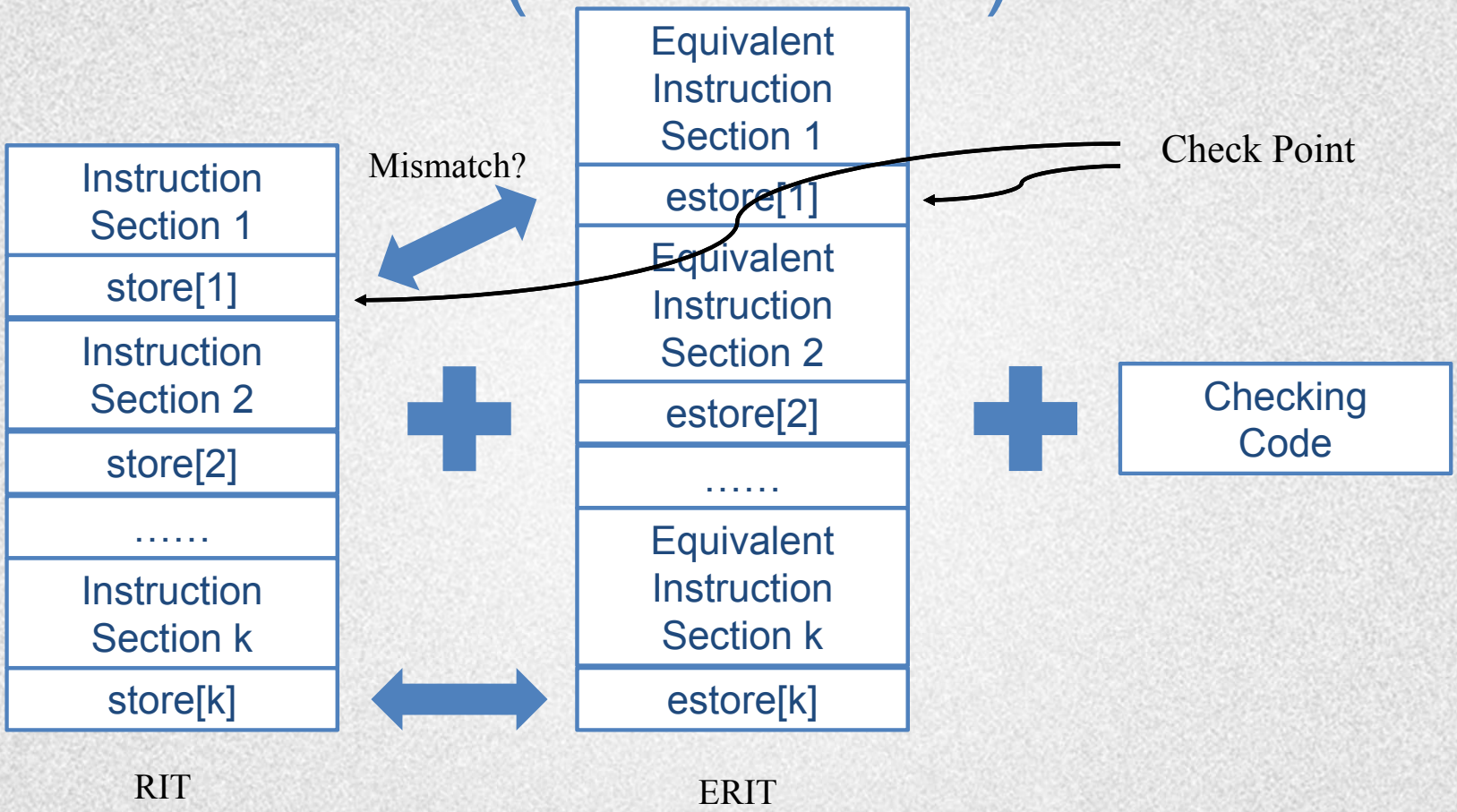
Framework (contd)



Question1:
Why replay?

Question2:
What if equivalent instruction is the offending one?

Enhanced RIT (Check Point)



Hardware Support

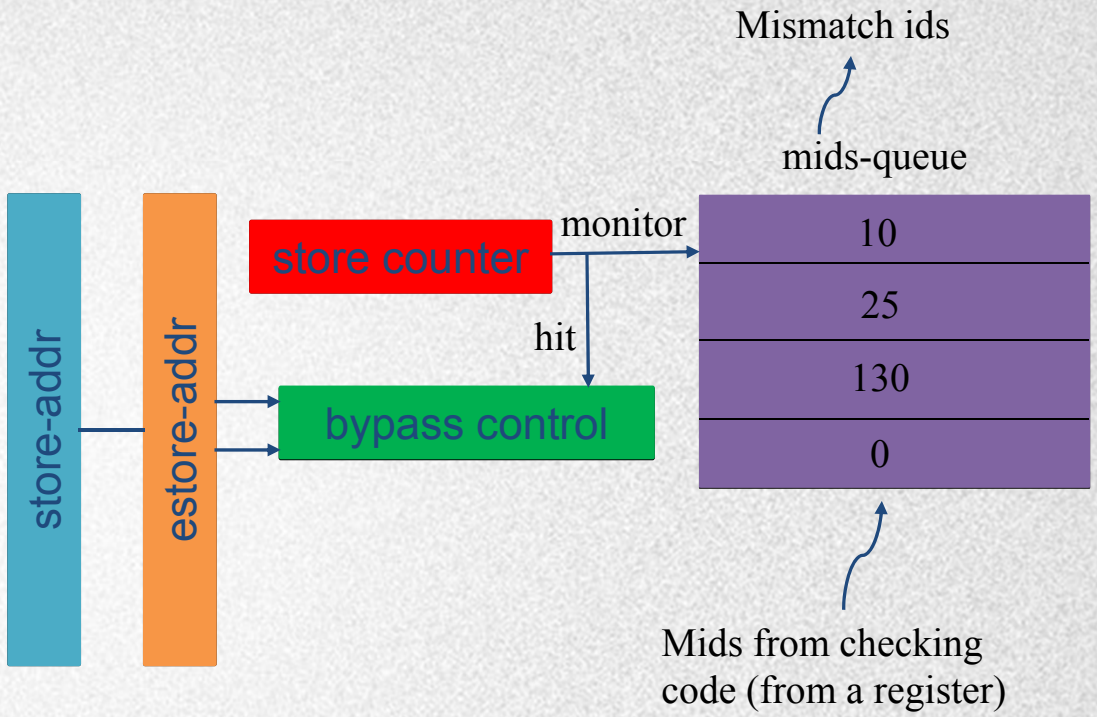
store-addr: buffer to store PC of all k store instructions.

estore-addr: buffer to store PC of all k estore instructions.

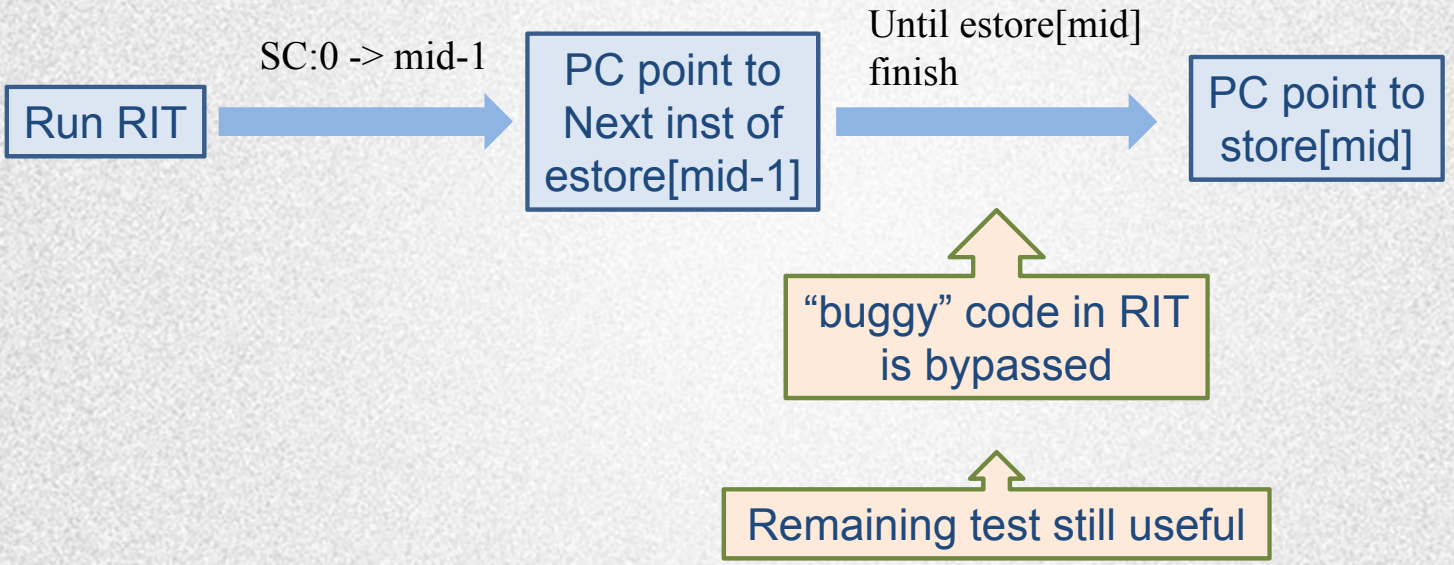
mids-queue: store every mismatch id (0~k).

store counter: counts the number of stores for each run.

bypass control: control PC to bypass buggy code.

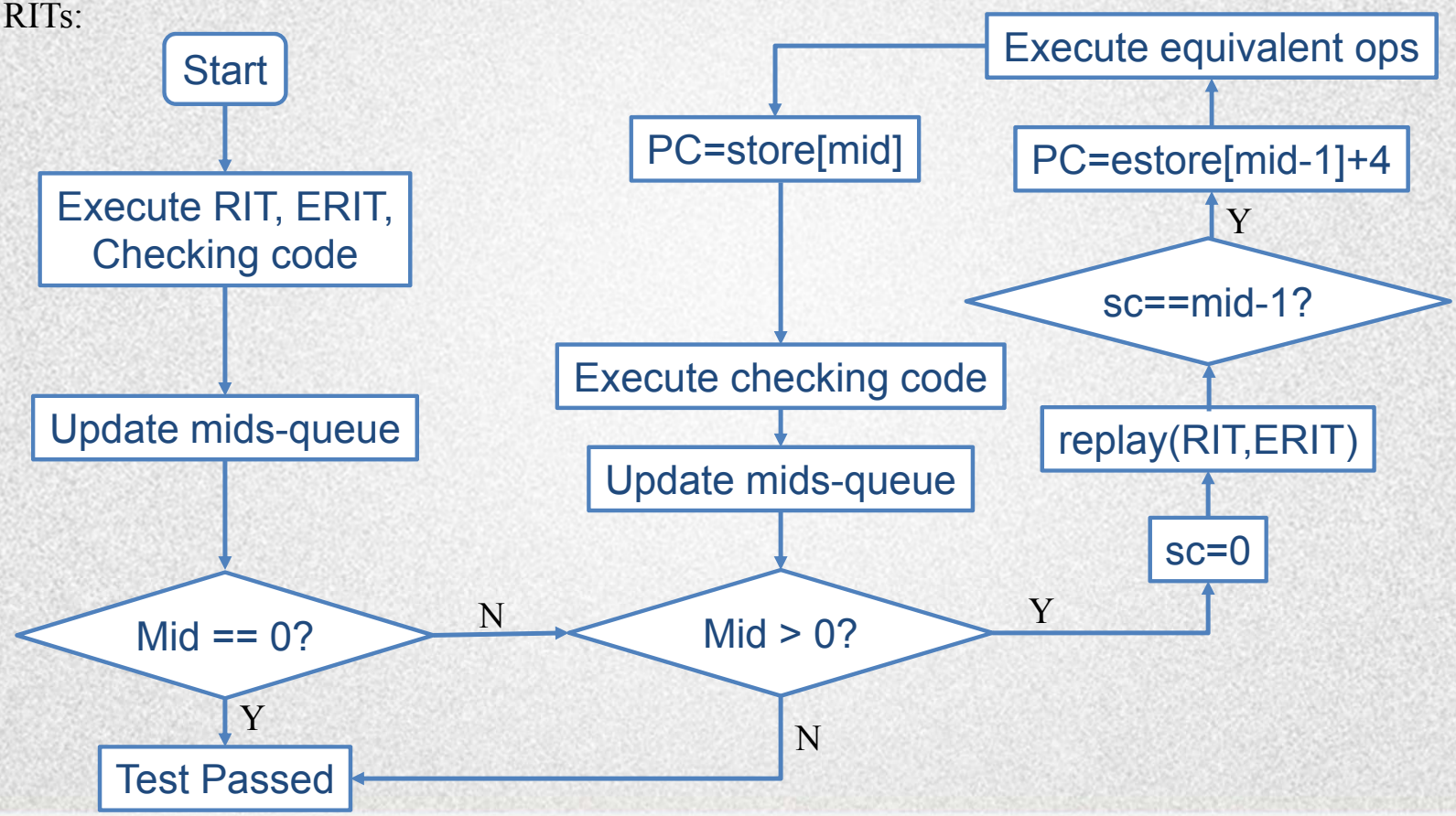


Bypass Control

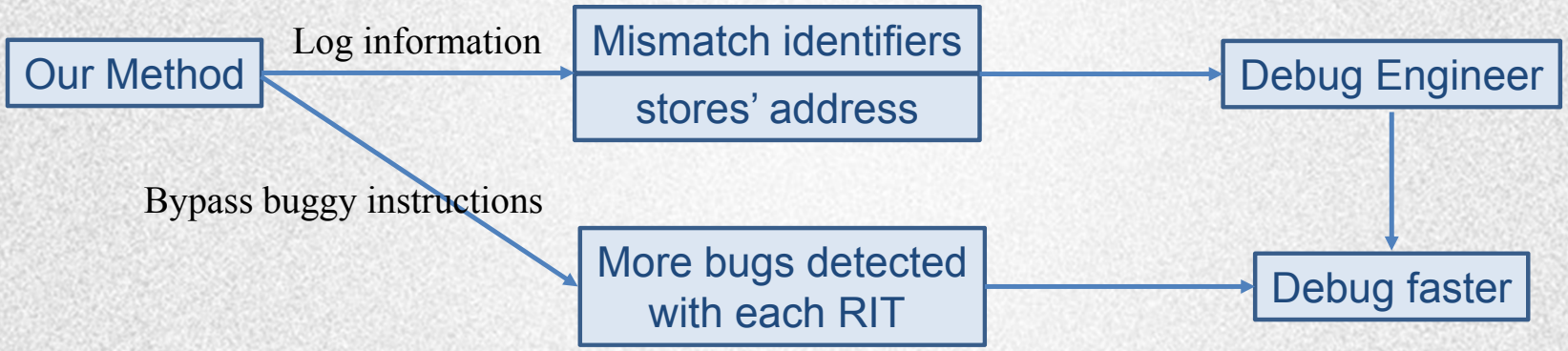


Test Flow

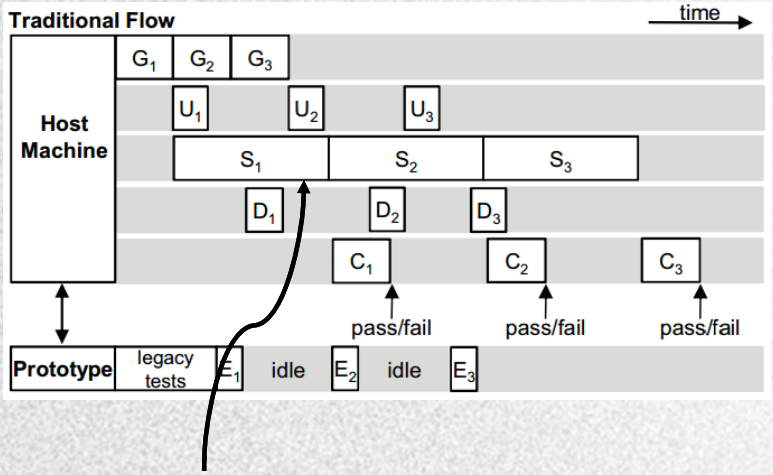
For all RITs:



Post-processing (triage)

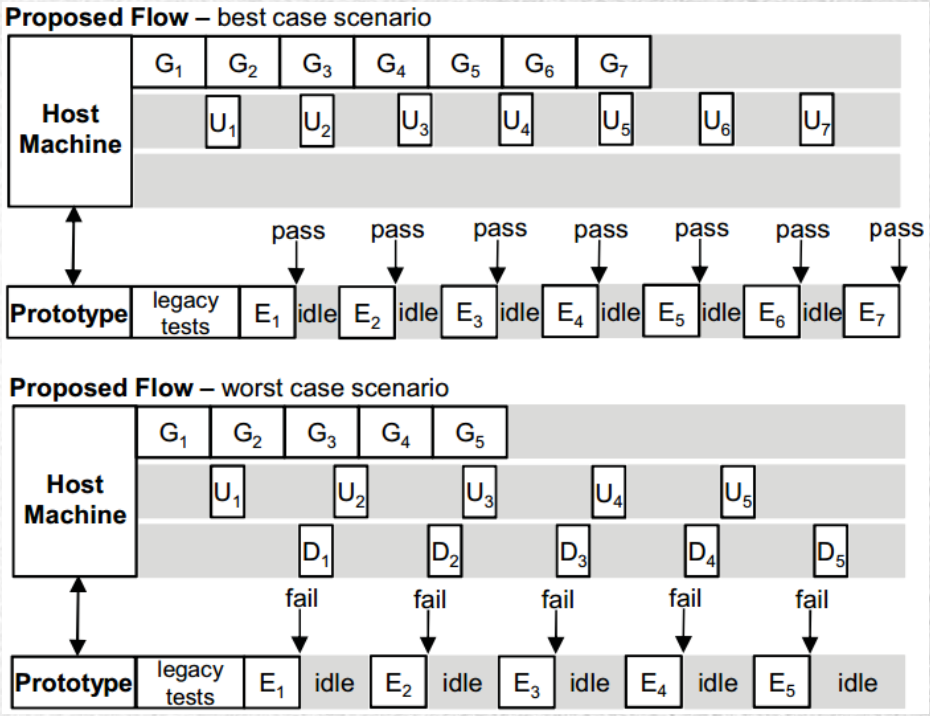


Time Analysis



Simulation time is too long!

VS

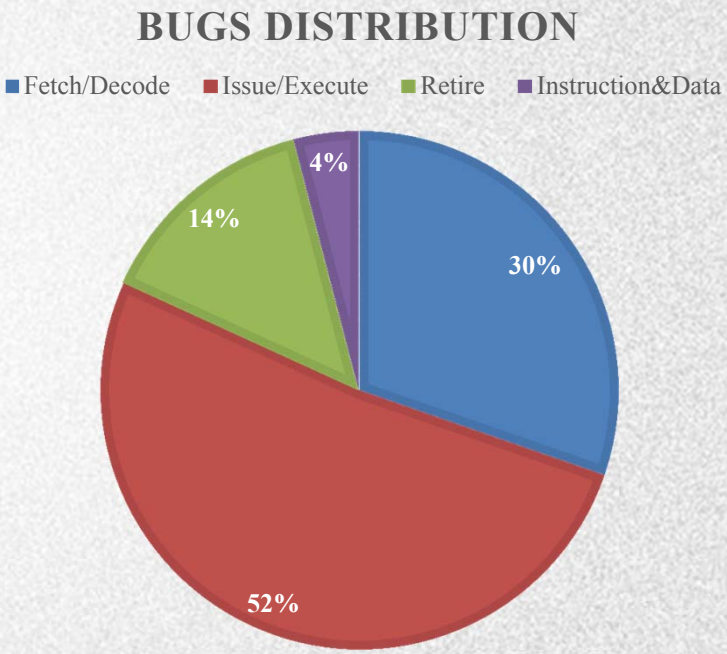


Experimental Environment

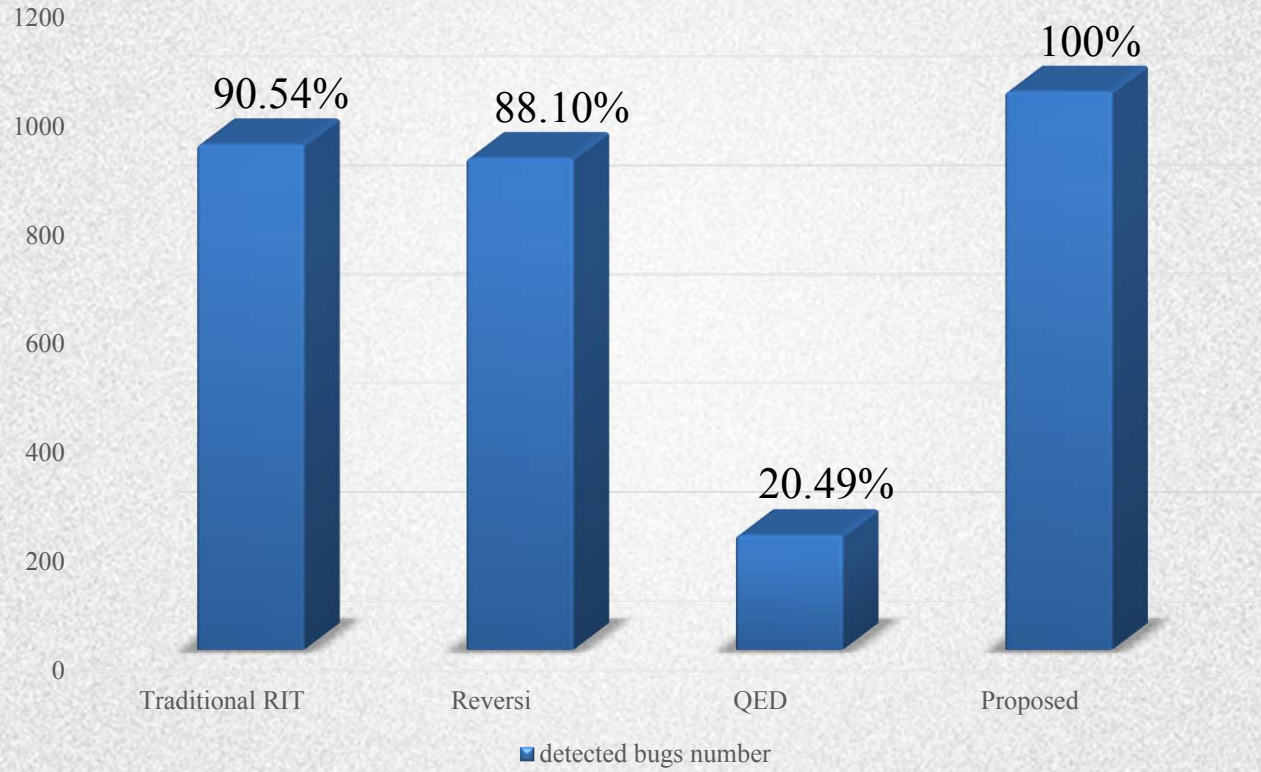
- PTLsim simulator: superscalar, OoO, single core x86-compatible
- Bugs injected: 802 logic, 225 electrical, 1025 total
- Original RIT: 154 RITS, ~4K Inst each, 616K Inst total
- Comparison: Reversi & QED
- Test Size: 2.4M Instructions for Reversi
  1.8M Instructions for QED
  3.7M Instructions for Proposed

Injected Bugs Distribution

Pipeline Stage	Component	Logic bugs	Electrical bugs	Total bugs
Fetch/ Decode	Branch Predictor	71	16	**87**
	Prefetcher	29	12	**41**
	Instruction Decoder	100	–	**100**
	Microcode	62	–	**62**
	Instruction Buffer	–	18	**18**
Issue/ Execute	Integer Arithmetic	95	–	**95**
	FP Arithmetic	97	–	**97**
	Jump logic	46	–	**46**
	Load/Store logic	66	21	**87**
	Issue Queue	42	–	**42**
	Scheduler	32	–	**32**
	Register File	61	63	**124**
Retire	Reorder Buffer	101	41	**142**
Instruction & Data		–	52	**52**
Total		**802**	**223**	**1025**



Detected Bugs

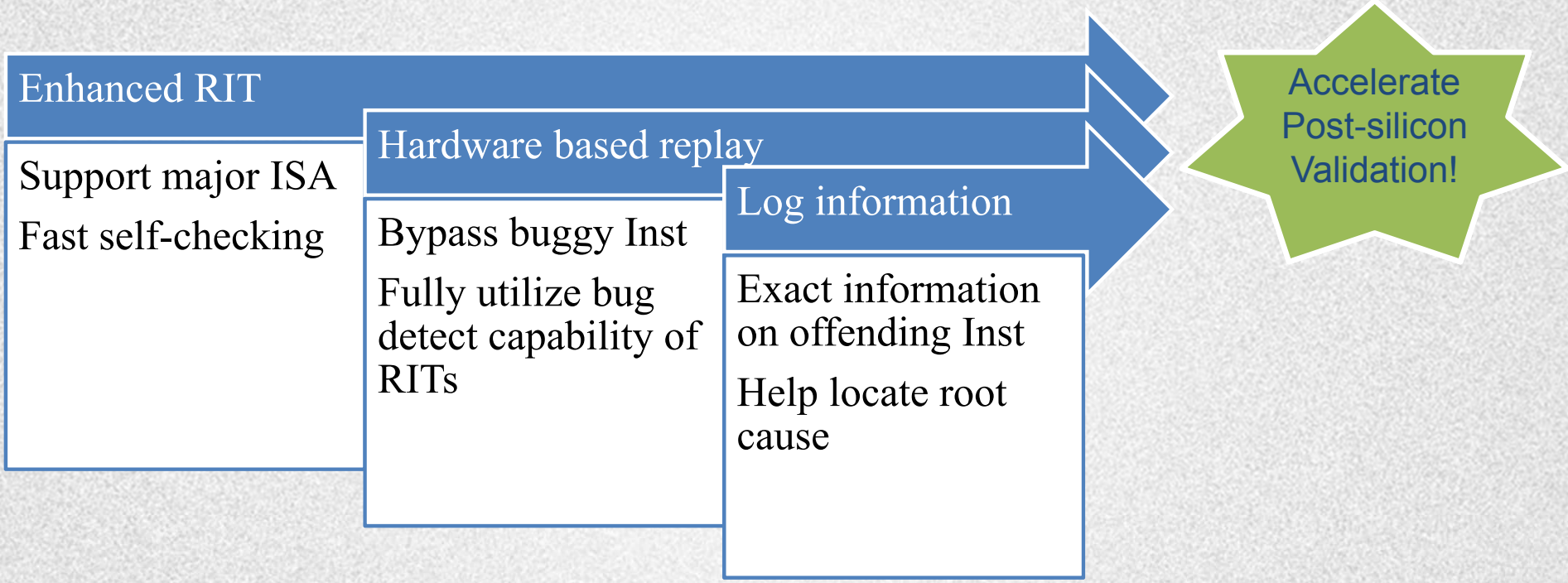


Validation Times

Time (Sec)	Trad. RIT	Reversi	QED	Proposed
Generation	4.460	6.310	5.530	7.680
Simulation	51.000	-	-	-
Execution	0.027	0.110	0.071	0.176
Total	55.487	6.420	5.601	7.856

Upload time, Download time and Compare time are nearly zero so neglected, But should be included for large tests.

Conclusion


Enhanced RIT
Support major ISA
Fast self-checking
Hardware based replay
Bypass buggy Inst
Fully utilize bug detect capability of RITs
Log information
Exact information on offending Inst
Help locate root cause
Accelerate Post-silicon Validation!

Q&A

Debate

1. Is it a good idea to apply this method to pre-silicon verification?

2. Can we replace ERIT instructions with RIT instructions when mismatch happens?