Inherent Time Redundancy (ITR): Using Program Repetition for Low-Overhead Fault Tolerance

Vimal Reddy

Eric Rotenberg

Center for Efficient, Secure and Reliable Computing (CESR)

Dept. of Electrical & Computer Engineering

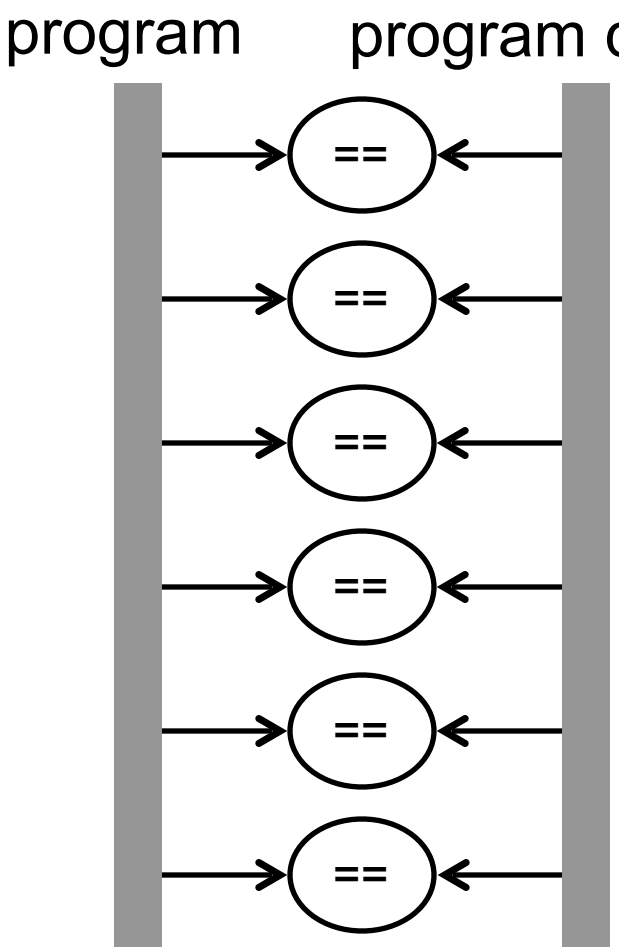
North Carolina State University

Raleigh, NC

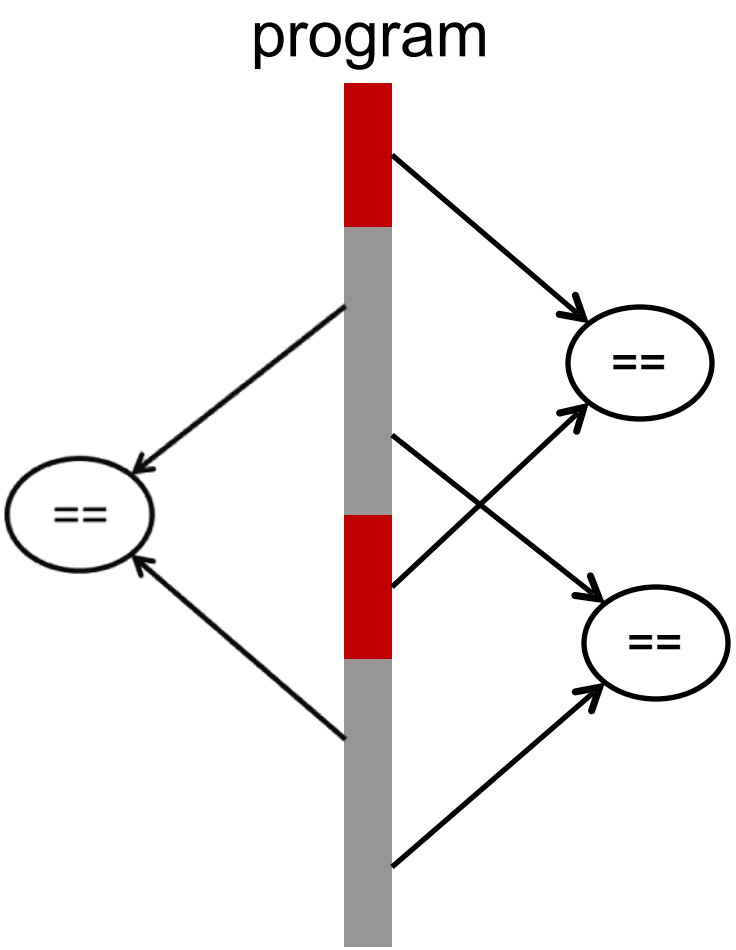
Processor Fault Tolerance

- Main approaches: Time or space redundancy
  - Explicitly duplicate in time or space
  - Match values/signals
  - E.g., AR-SMT (time), IBM S/390 G5 (space)
- High overheads
  - Area/power/performance
  - Unsuitable for commodity high-performance processors

New Idea: Inherent Time Redundancy



Conventional time redundancy



Inherent time redundancy

Outline

- Exploiting ITR for Fault Tolerance
- ITR Cache for Checking Decode Signals
- ITR Cache Hit/Miss Scenarios
- Microarchitecture Extensions to Superscalar
- Results
  - Repetition
  - Fault Coverage Based on Cache Hits/Misses
  - Fault Coverage Based on Fault Injection
- Area and Power Overhead
- Conclusion

Exploiting ITR for Fault Tolerance

- Key ideas:
  - Record input-independent signals for instructions
  - Confirm their correctness upon repetition
- This paper focuses on decode signals
  - Provides protection for fetch and decode stages
- Longer term goal
  - Apply ITR to other input-independent stages
  - My thesis combines ITR with other microarch.-level checks for a comprehensive fault regimen

Outline

- Exploiting ITR for Fault Tolerance
- ITR Cache for Checking Decode Signals
- ITR Cache Hit/Miss Scenarios
- Microarchitecture Extensions to Superscalar
- Results
  - Repetition
  - Fault Coverage Based on Cache Hits/Misses
  - Fault Coverage Based on Fault Injection
- Area and Power Overhead
- Conclusion

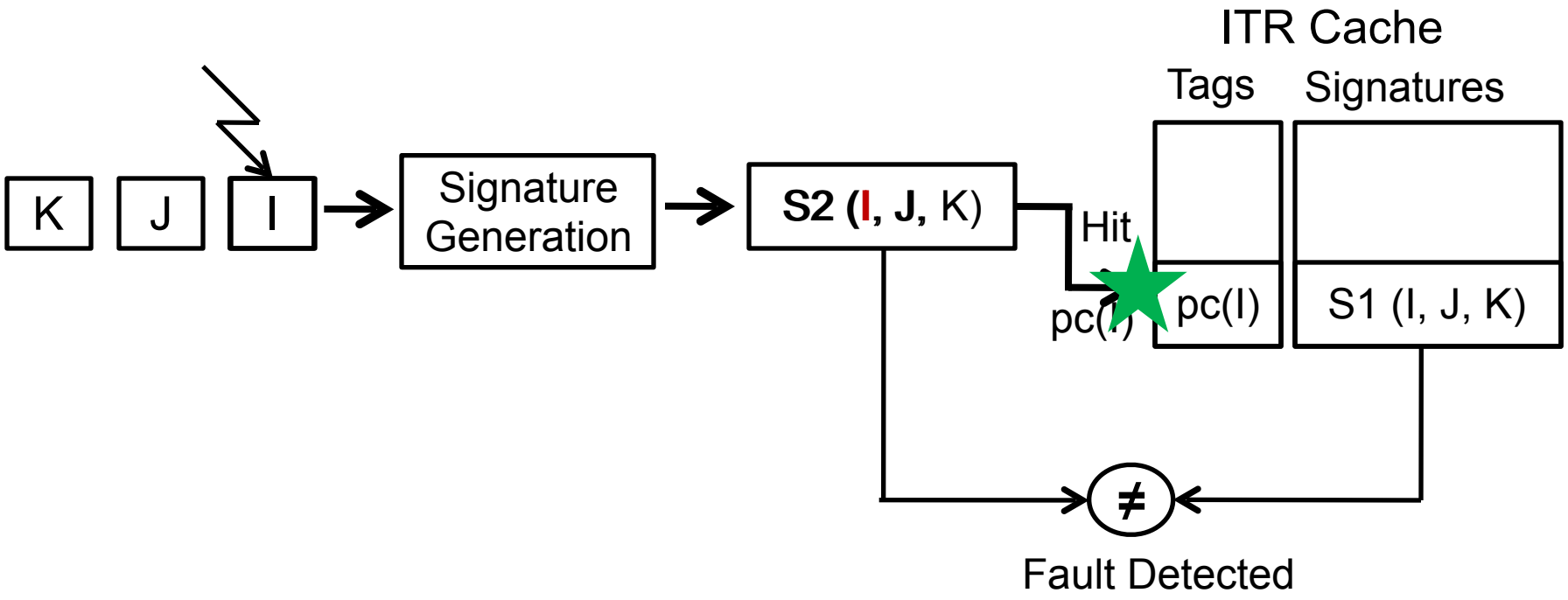
ITR Cache for Checking Decode Signals

- Create decode signatures at trace granularity
  - Trace ends at branch or 16 instructions (arbitrary)
  - Combine decode signals of instructions in a trace
- Record decode signatures in an *ITR cache*
  - Indexed by start PC (program counter) of a trace
- For each new trace, compare with ITR cache
  - Fault if new signature mismatches ITR cache

Outline

- Exploiting ITR for Fault Tolerance
- ITR Cache for Checking Decode Signals
- ITR Cache Hit/Miss Scenarios
- Microarchitecture Extensions to Superscalar
- Results
  - Repetition
  - Fault Coverage Based on Cache Hits/Misses
  - Fault Coverage Based on Fault Injection
- Area and Power Overhead
- Conclusion

Scenario 1: ITR Cache Hit

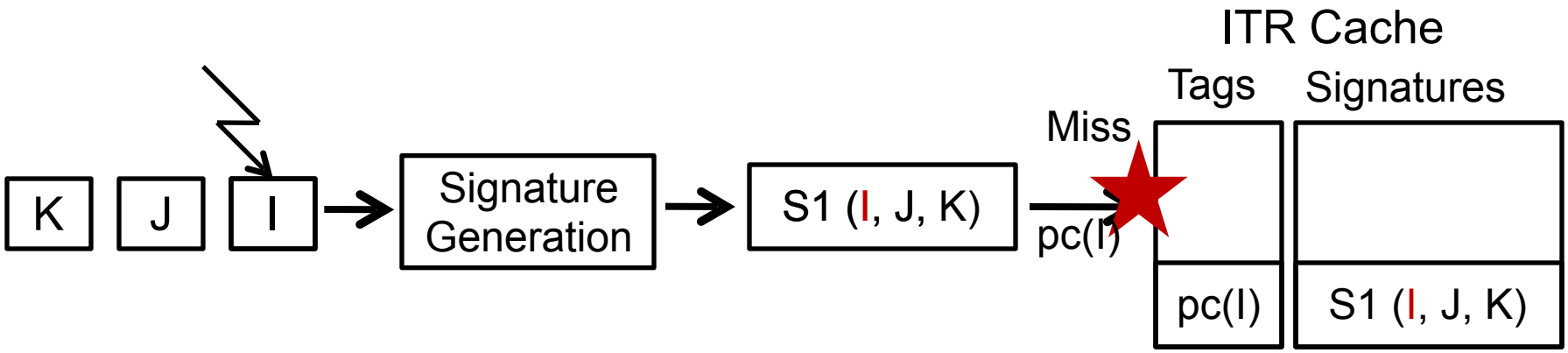


Notes:

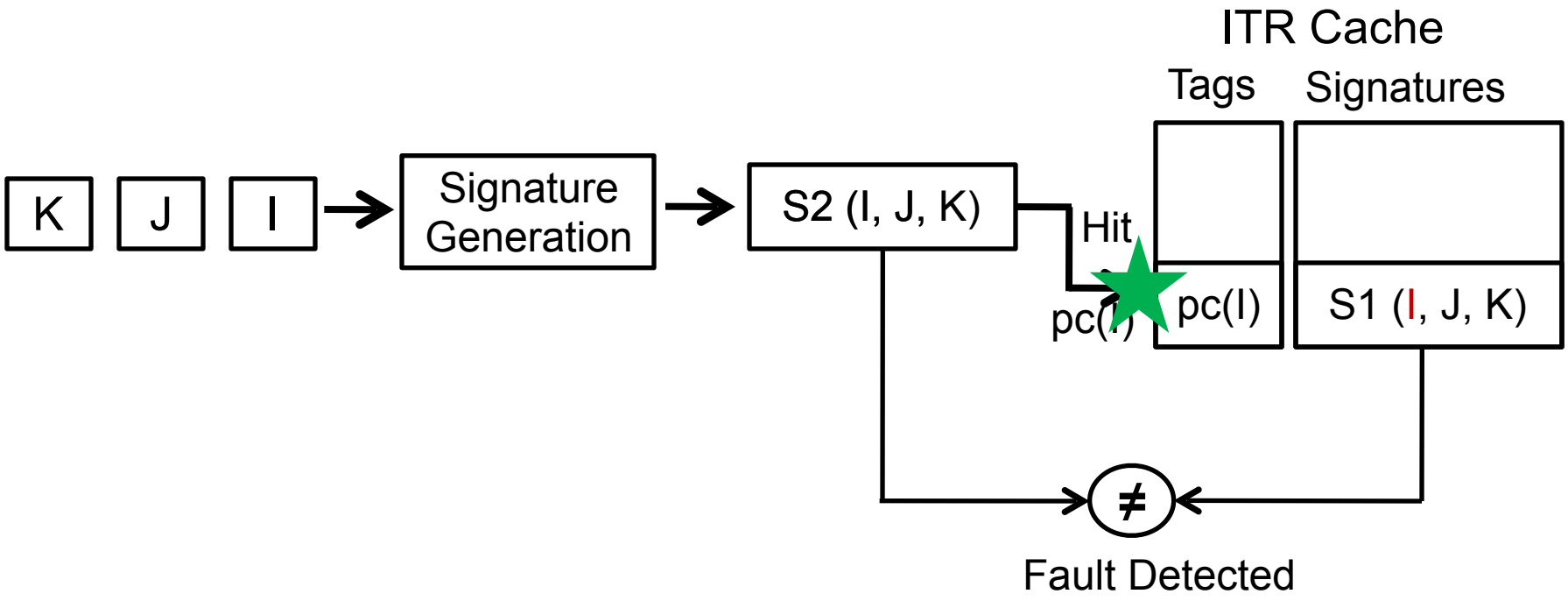
-- Faults on traces that hit (S2) are detectable

-- These faults are recoverable by flushing the processor pipeline and restarting from the faulting trace

Scenario 2: ITR Cache Miss Followed by Hit



Scenario 2: ITR Cache Miss Followed by Hit

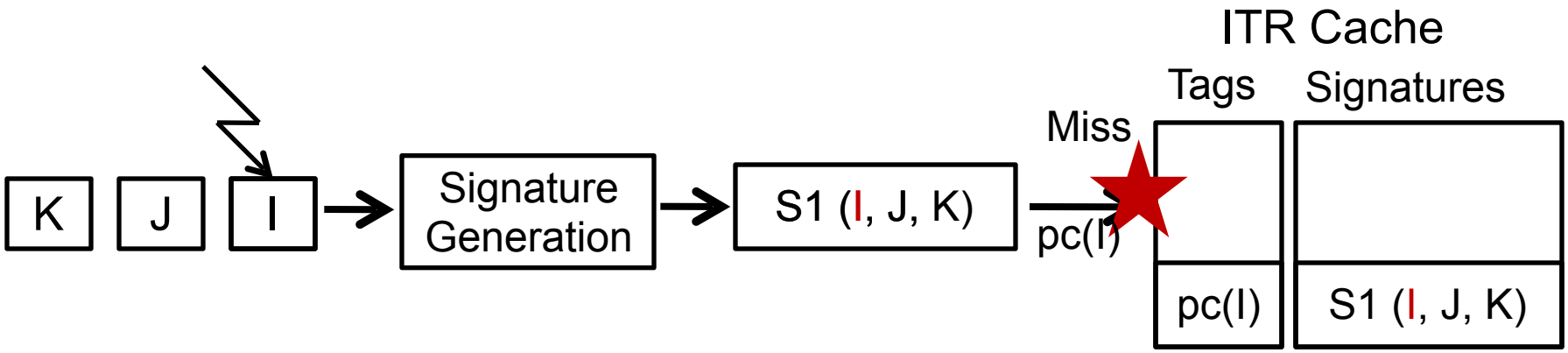


Notes:

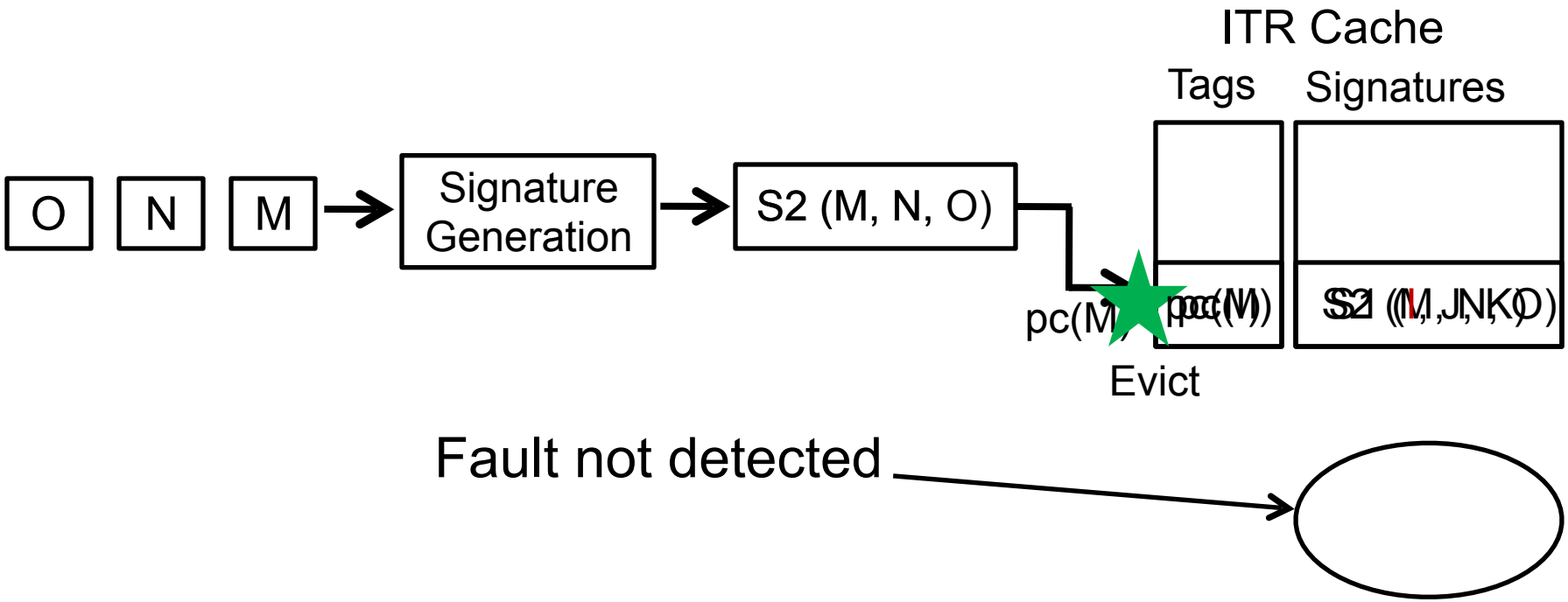
-- Faults on traces that miss but later hit (faulty S1) are detectable

-- Faults on S1 need checkpoint for recovery or must abort

Scenario 3: ITR Cache Miss Followed by Eviction



Scenario 3: ITR Cache Miss Followed by Eviction



Notes:

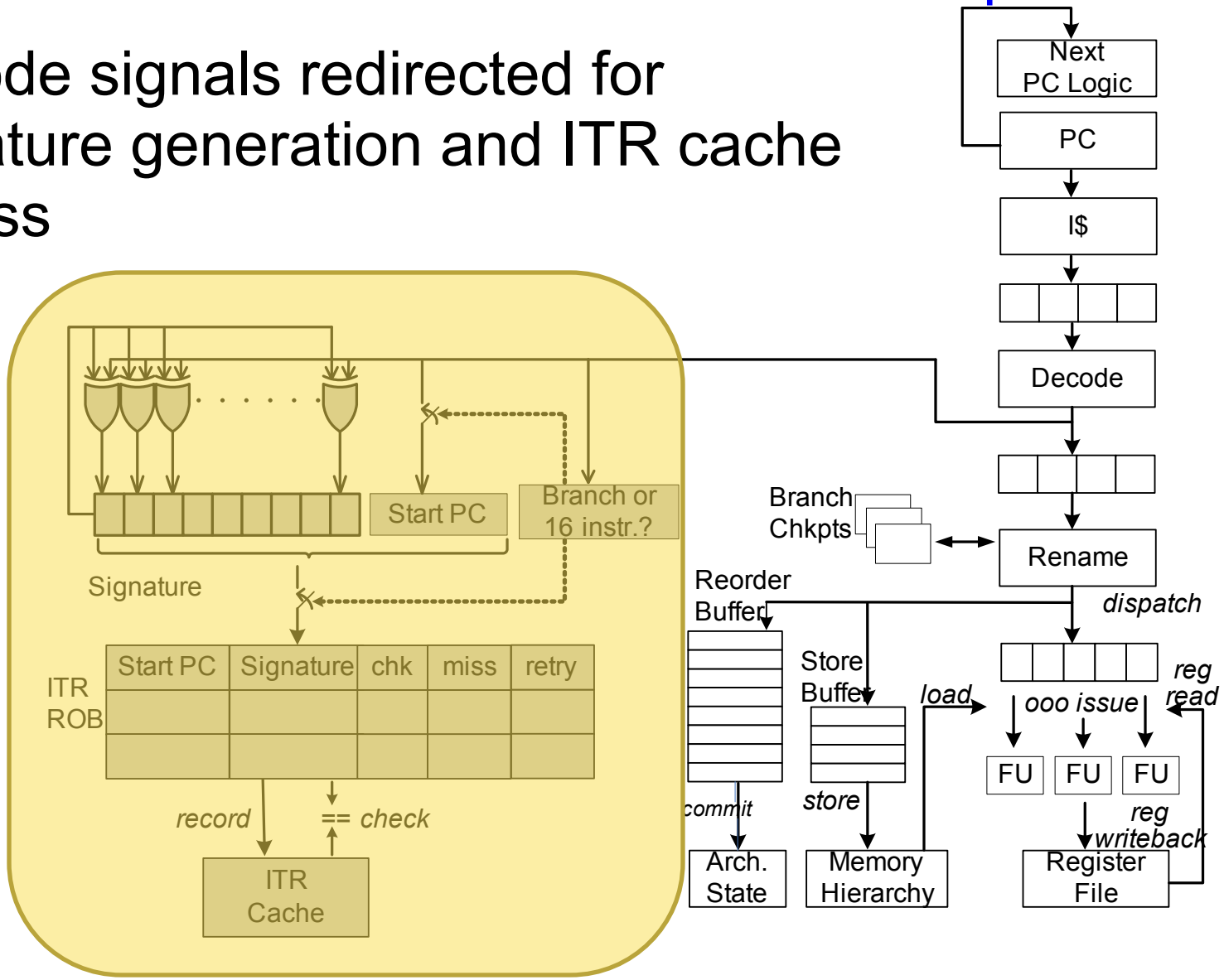
-- Faults on missed & evicted traces (S1) are not detected

Outline

- Exploiting ITR for Fault Tolerance
- ITR Cache for Checking Decode Signals
- ITR Cache Hit/Miss Scenarios
- Microarchitecture Extensions to Superscalar
- Results
  - Repetition
  - Fault Coverage Based on Cache Hits/Misses
  - Fault Coverage Based on Fault Injection
- Area and Power Overhead
- Conclusion

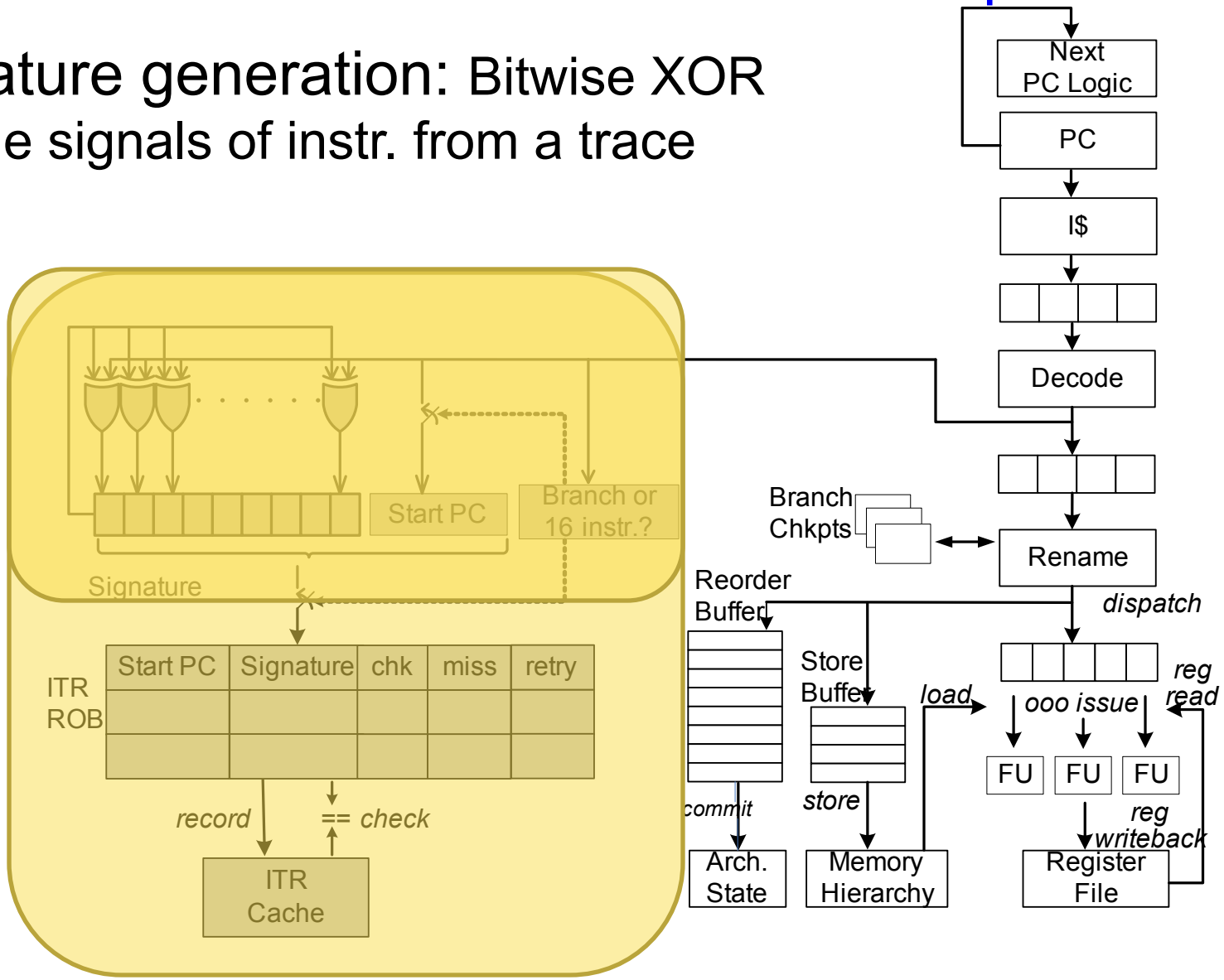
Microarchitecture Extensions to Superscalar

Decode signals redirected for signature generation and ITR cache access



Microarchitecture Extensions to Superscalar

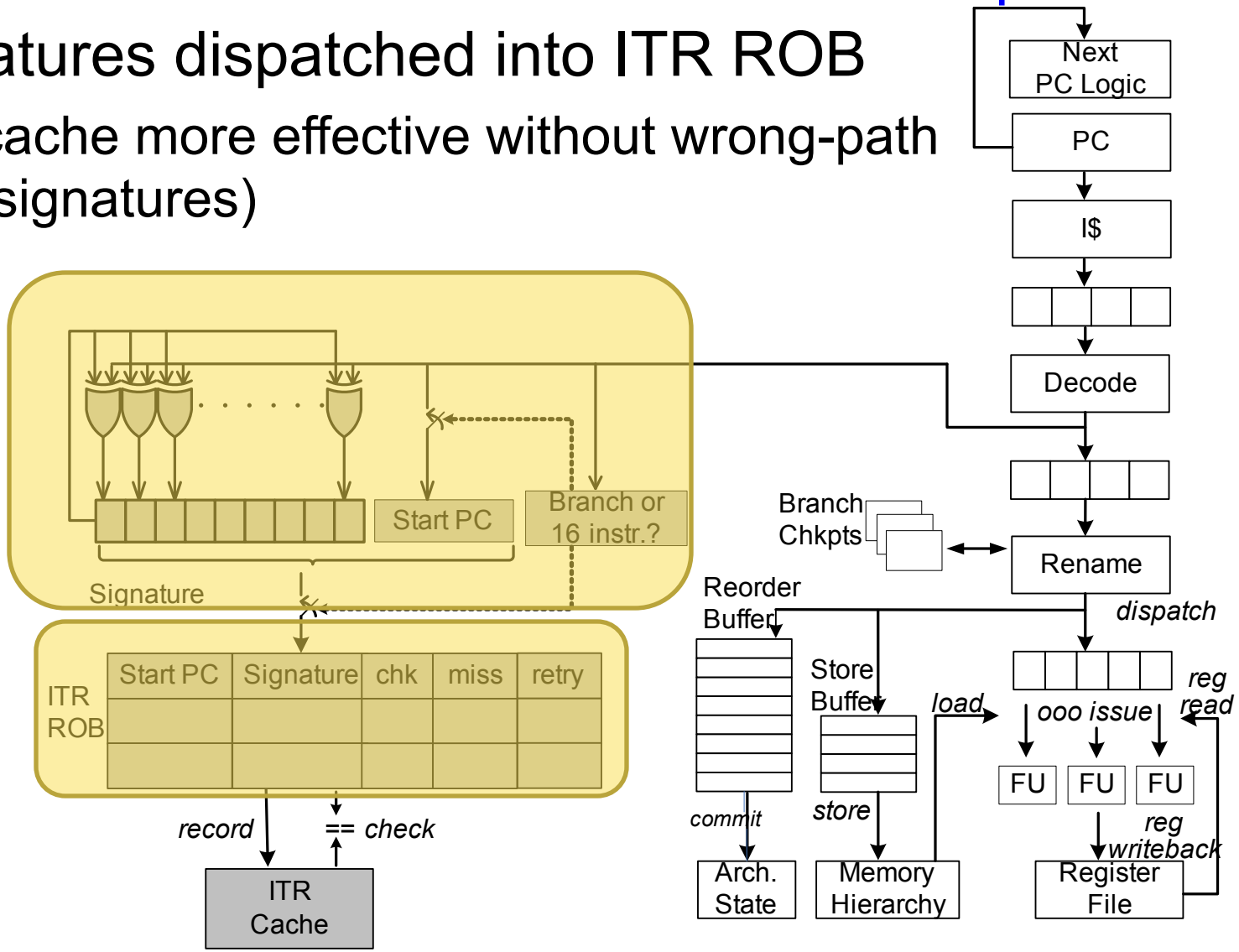
Signature generation: Bitwise XOR
decode signals of instr. from a trace



Microarchitecture Extensions to Superscalar

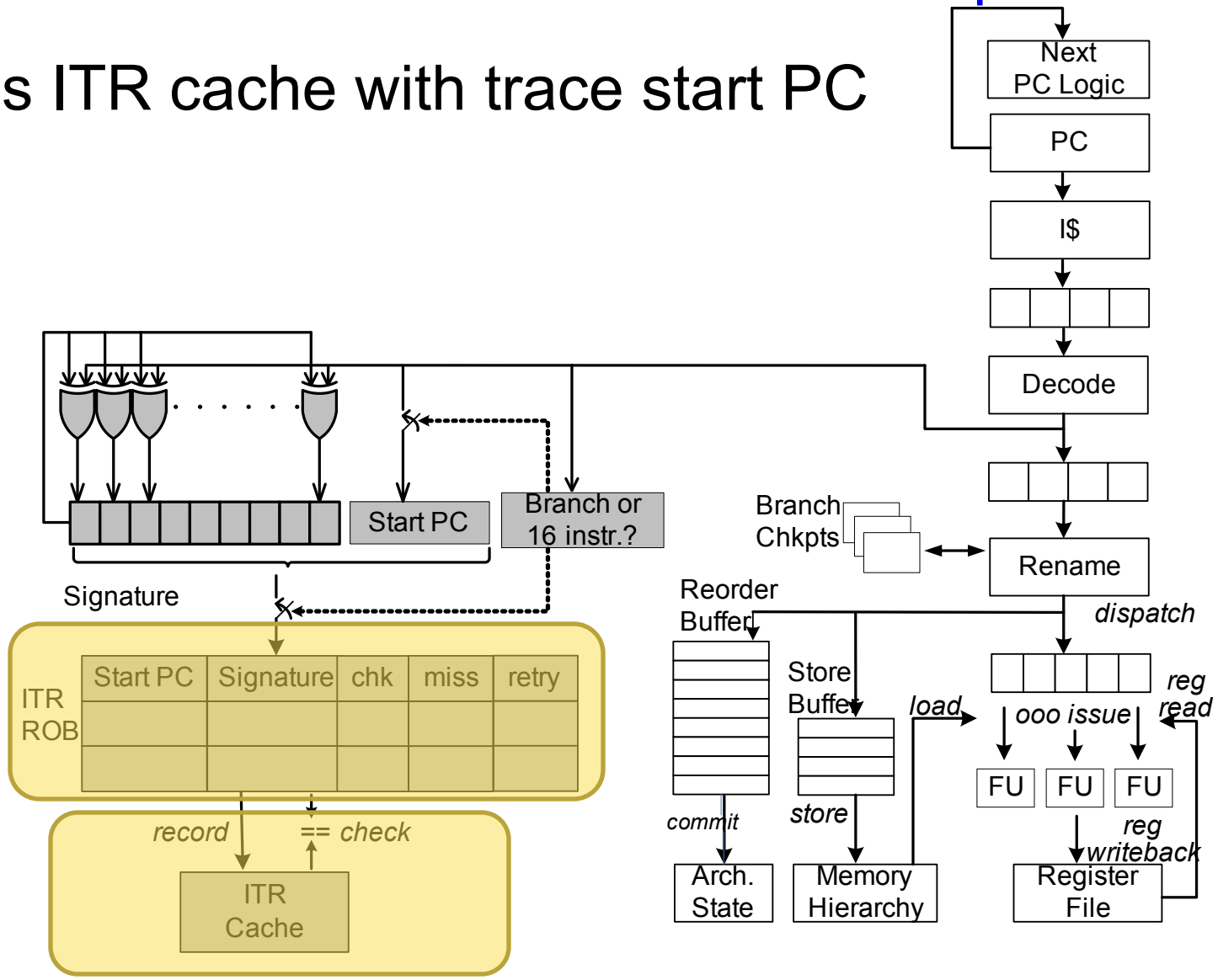
Signatures dispatched into ITR ROB

(ITR cache more effective without wrong-path trace signatures)



Microarchitecture Extensions to Superscalar

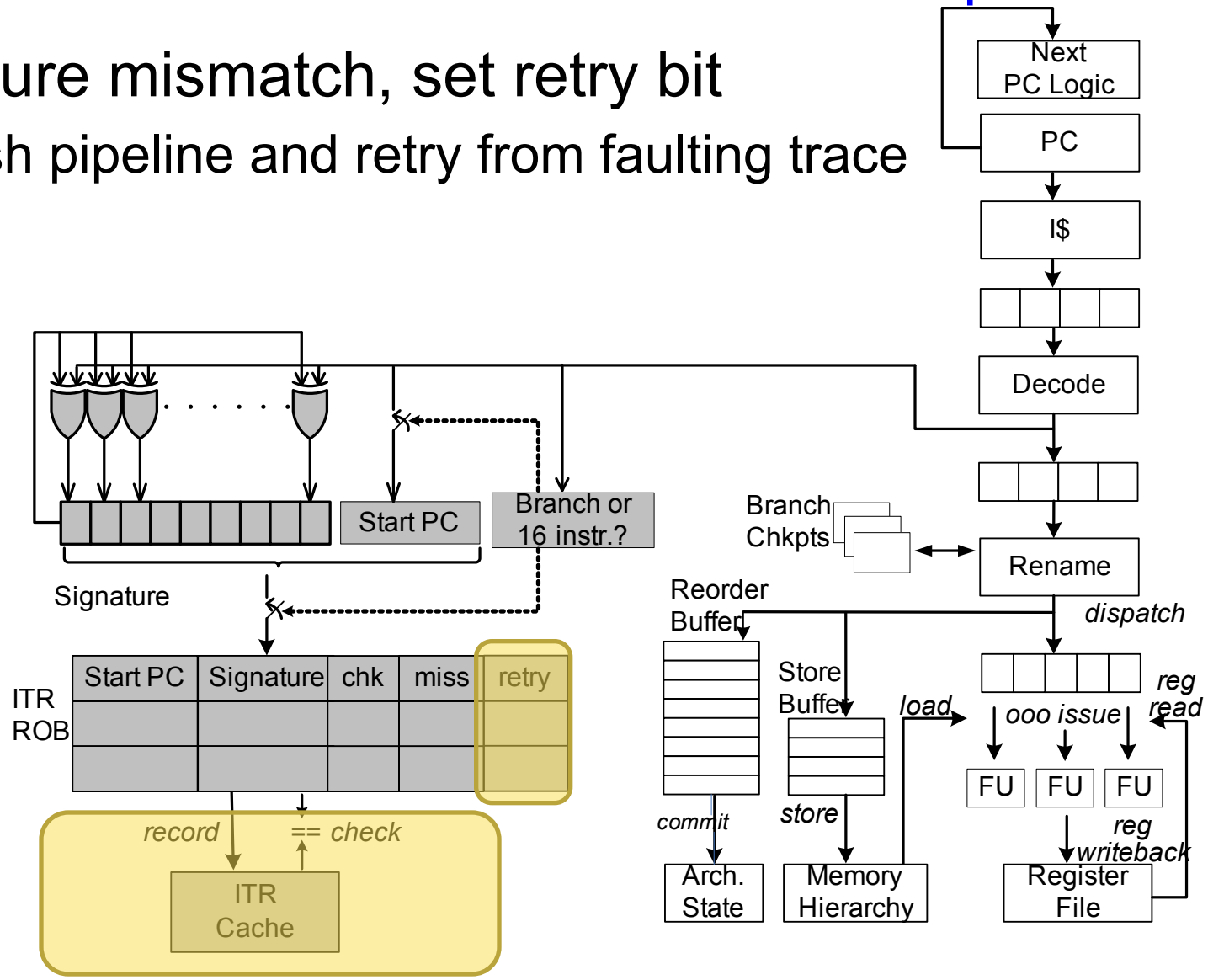
Access ITR cache with trace start PC



Microarchitecture Extensions to Superscalar

Signature mismatch, set retry bit

-- Flush pipeline and retry from faulting trace

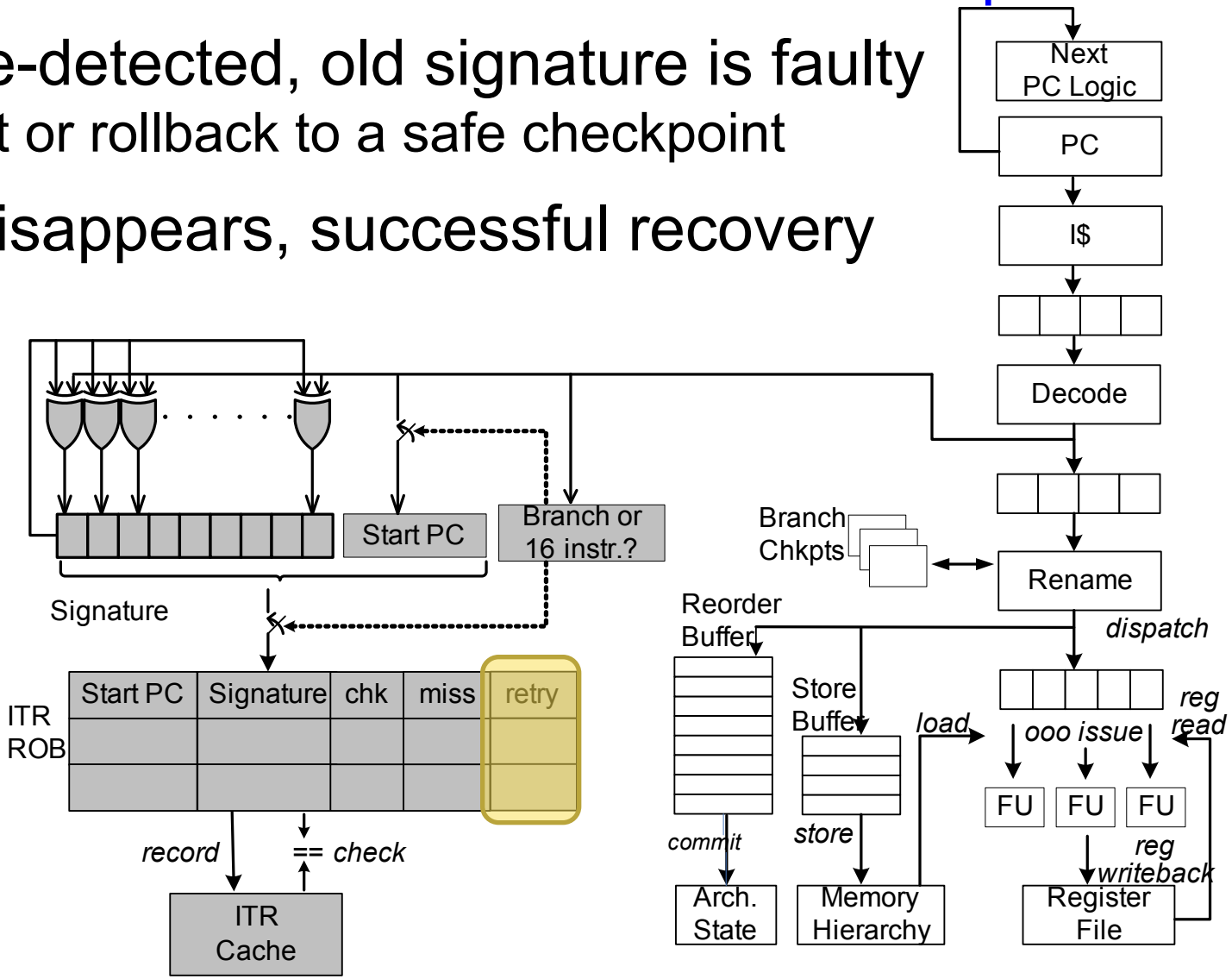


Microarchitecture Extensions to Superscalar

Fault re-detected, old signature is faulty

-- Abort or rollback to a safe checkpoint

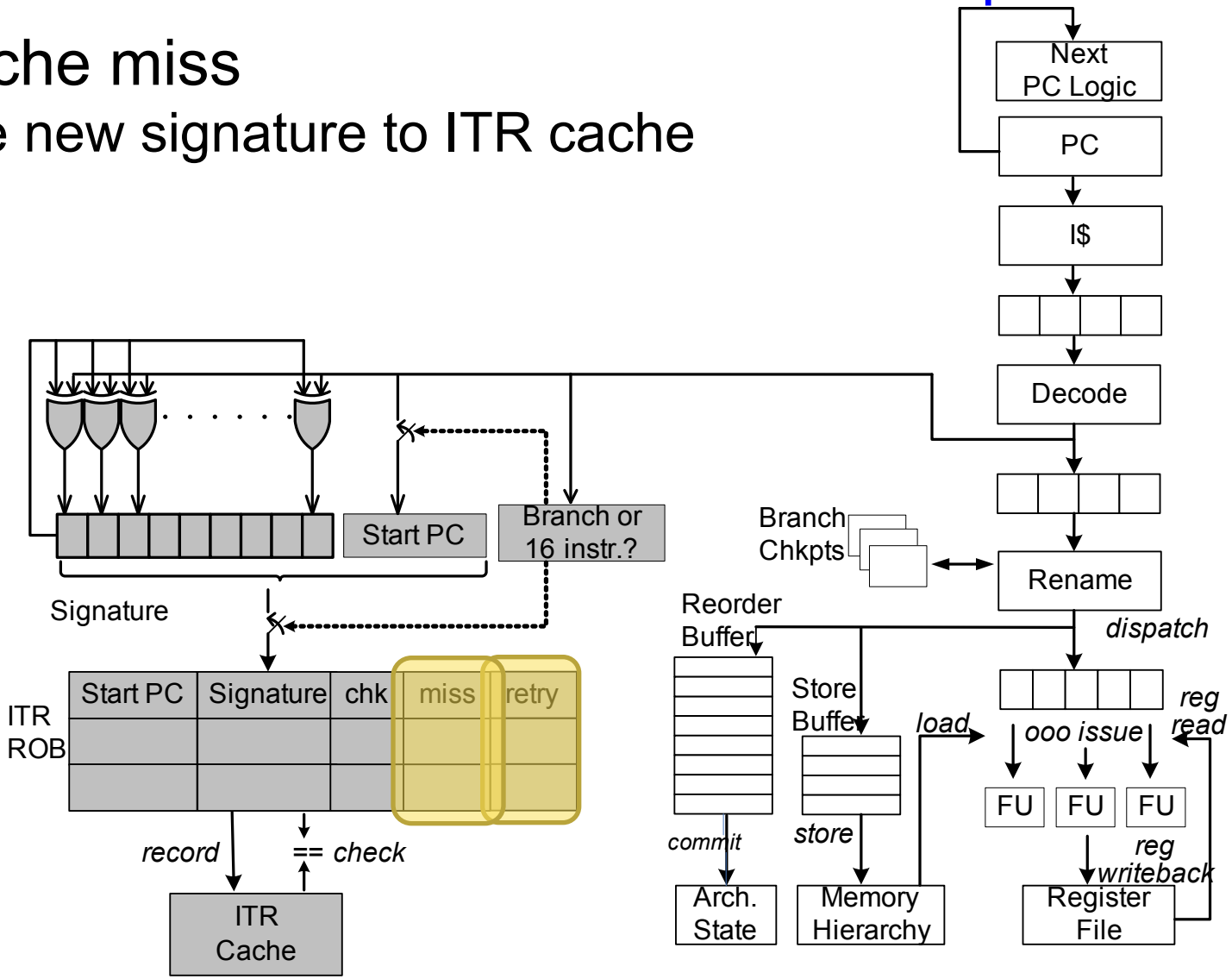
Fault disappears, successful recovery



Microarchitecture Extensions to Superscalar

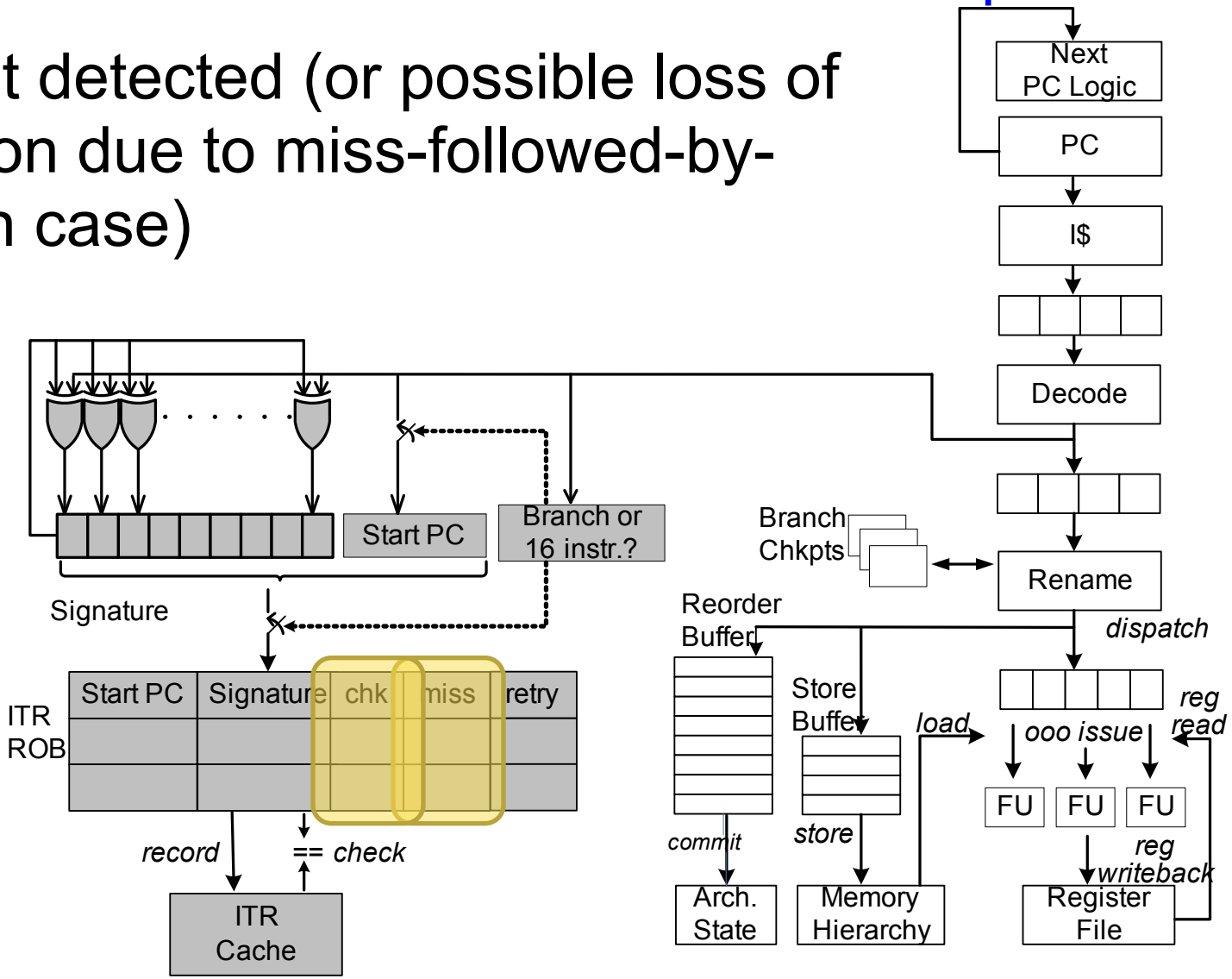
ITR cache miss

-- Write new signature to ITR cache



Microarchitecture Extensions to Superscalar

No fault detected (or possible loss of detection due to miss-followed-by-eviction case)



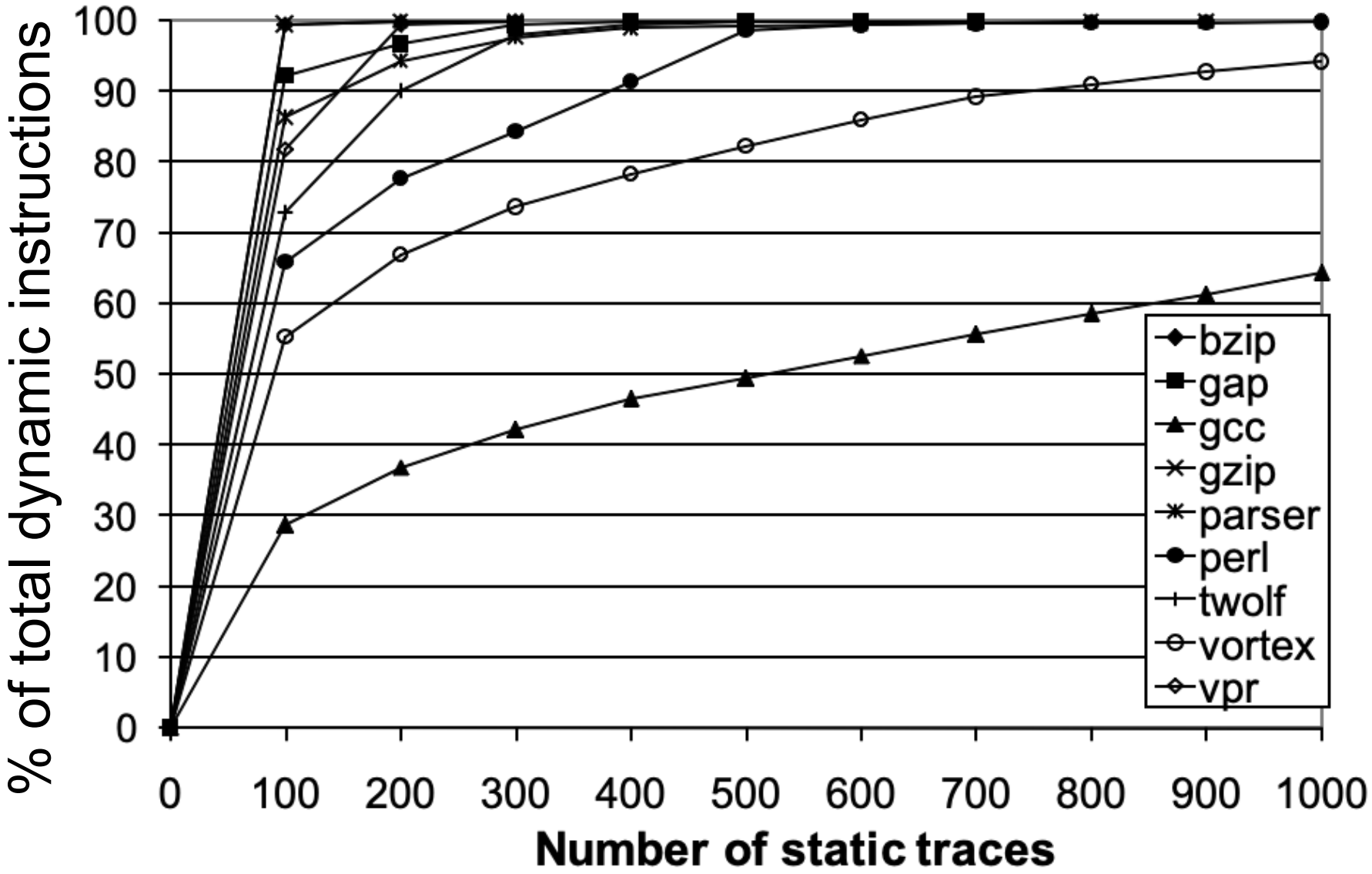
Outline

- Exploiting ITR for Fault Tolerance
- ITR Cache for Checking Decode Signals
- ITR Cache Hit/Miss Scenarios
- Microarchitecture Extensions to Superscalar
- Results
  - Repetition
  - Fault Coverage Based on Cache Hits/Misses
  - Fault Coverage Based on Fault Injection
- Area and Power Overhead
- Conclusion

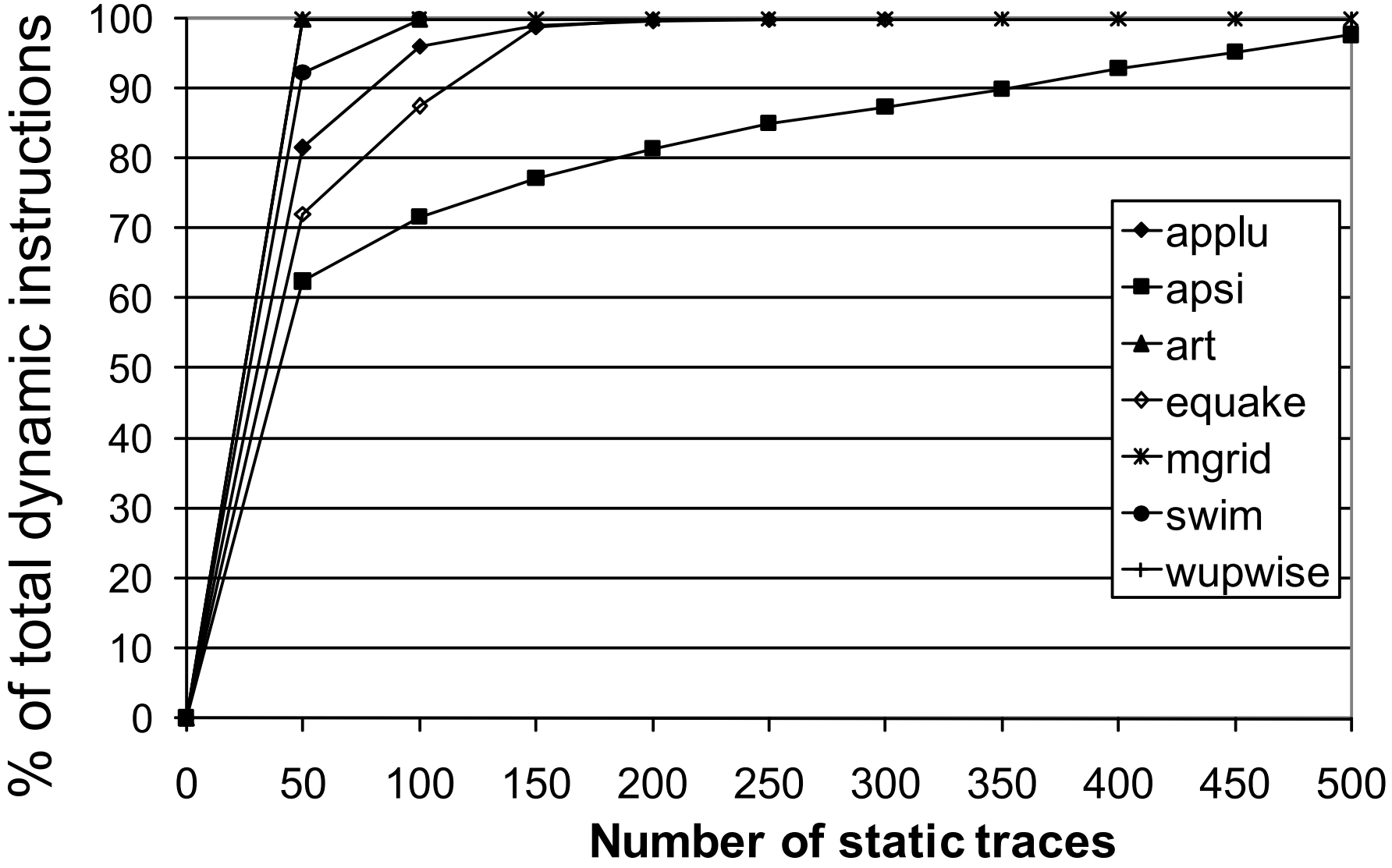
Outline

- Exploiting ITR for Fault Tolerance
- ITR Cache for Checking Decode Signals
- ITR Cache Hit/Miss Scenarios
- Microarchitecture Extensions to Superscalar
- Results
  - Repetition
  - Fault Coverage Based on Cache Hits/Misses
  - Fault Coverage Based on Fault Injection
- Area and Power Overhead
- Conclusion

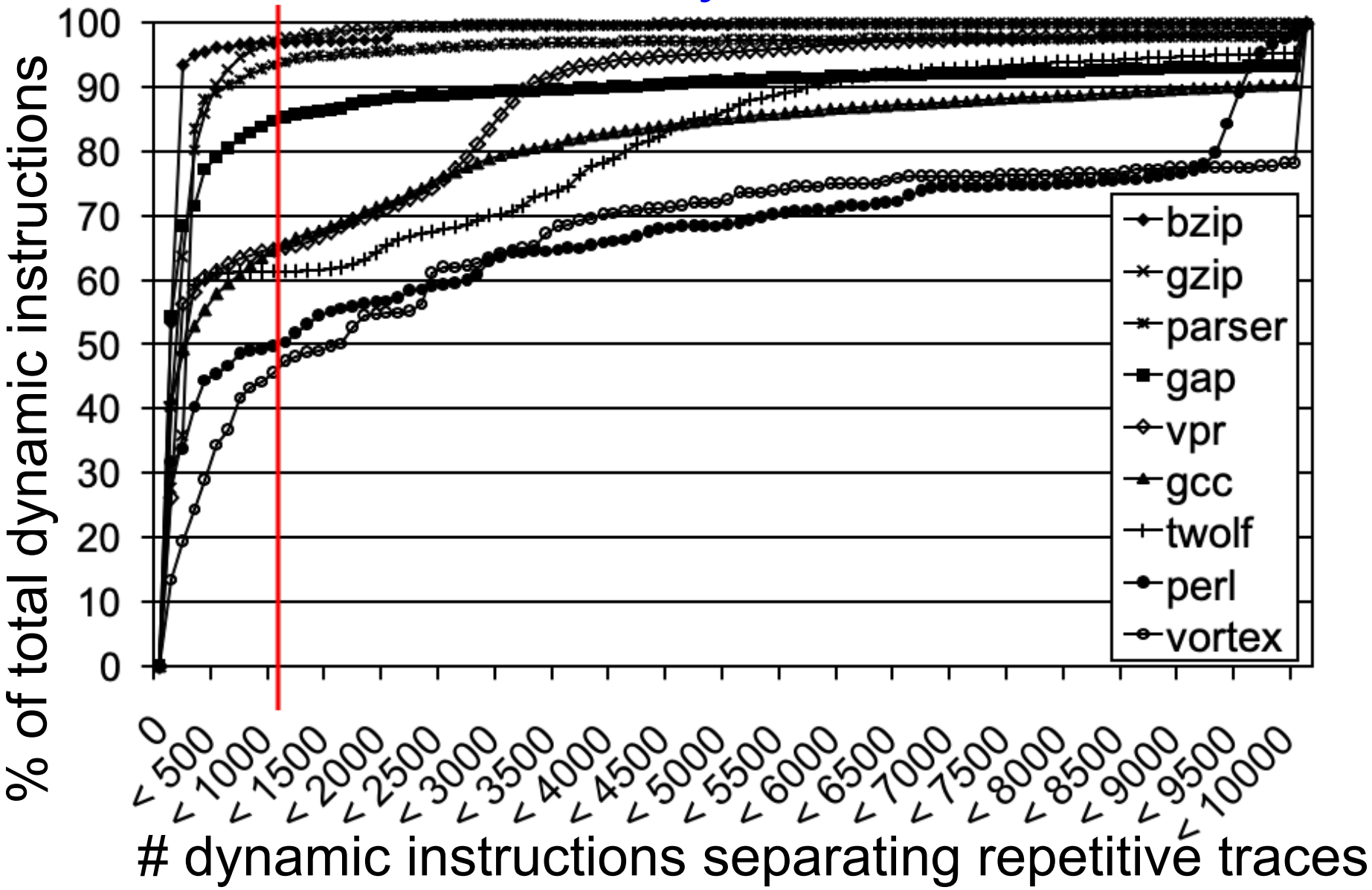
Results: Repetition in SPEC2K INT



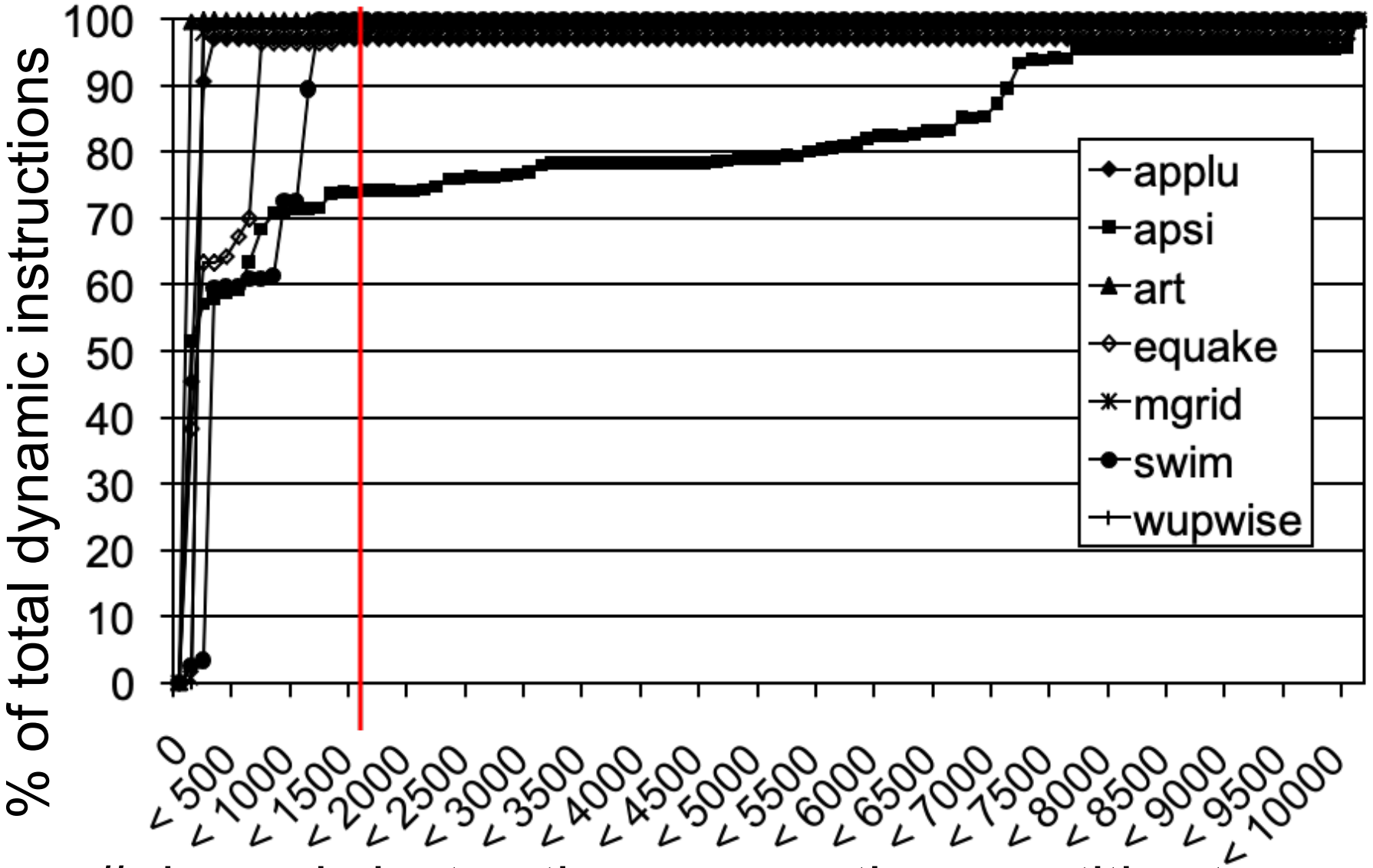
Results: Repetition in SPEC2K FP



Results: Proximity in SPEC2K INT



Results: Proximity in SPEC2K FP



Outline

- Exploiting ITR for Fault Tolerance
- ITR Cache for Checking Decode Signals
- ITR Cache Hit/Miss Scenarios
- Microarchitecture Extensions to Superscalar
- Results
  - Repetition
  - Fault Coverage Based on Cache Hits/Misses
  - Fault Coverage Based on Fault Injection
- Area and Power Overhead
- Conclusion

Fault Coverage Based on ITR Cache Misses

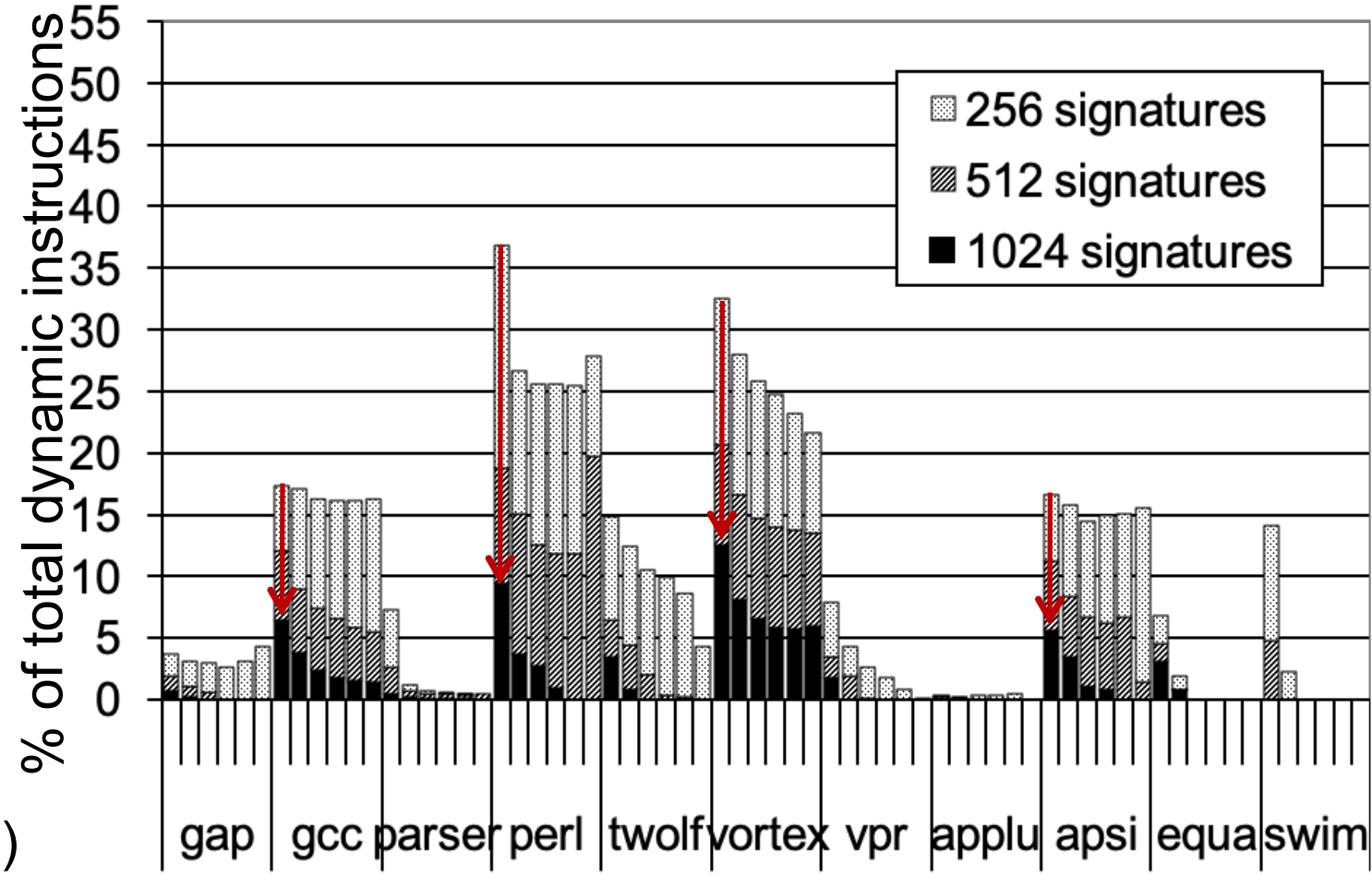
- Loss in fault detection coverage
  - Miss-followed-by-eviction causes loss in fault detection
  - (# of missed & evicted traces) x instructions/trace
- Loss in fault recovery coverage (w/o checkpoints)
  - Misses cause loss in fault recovery
  - (# of missed traces) x instructions/trace
- Tried various ITR cache configurations
  - Direct-mapped (DM), 2,4,8,16-way, fully-associative (FA)
  - 256, 512 and 1024 signatures
  - Bzip, gzip, art, mgrid and wupwise had <1% loss in coverage (results not shown)

Results: Loss in Fault Detection Coverage

Coverage loss correlates with proximity of repetition e.g., perl and vortex

Capacity is important for poor performers

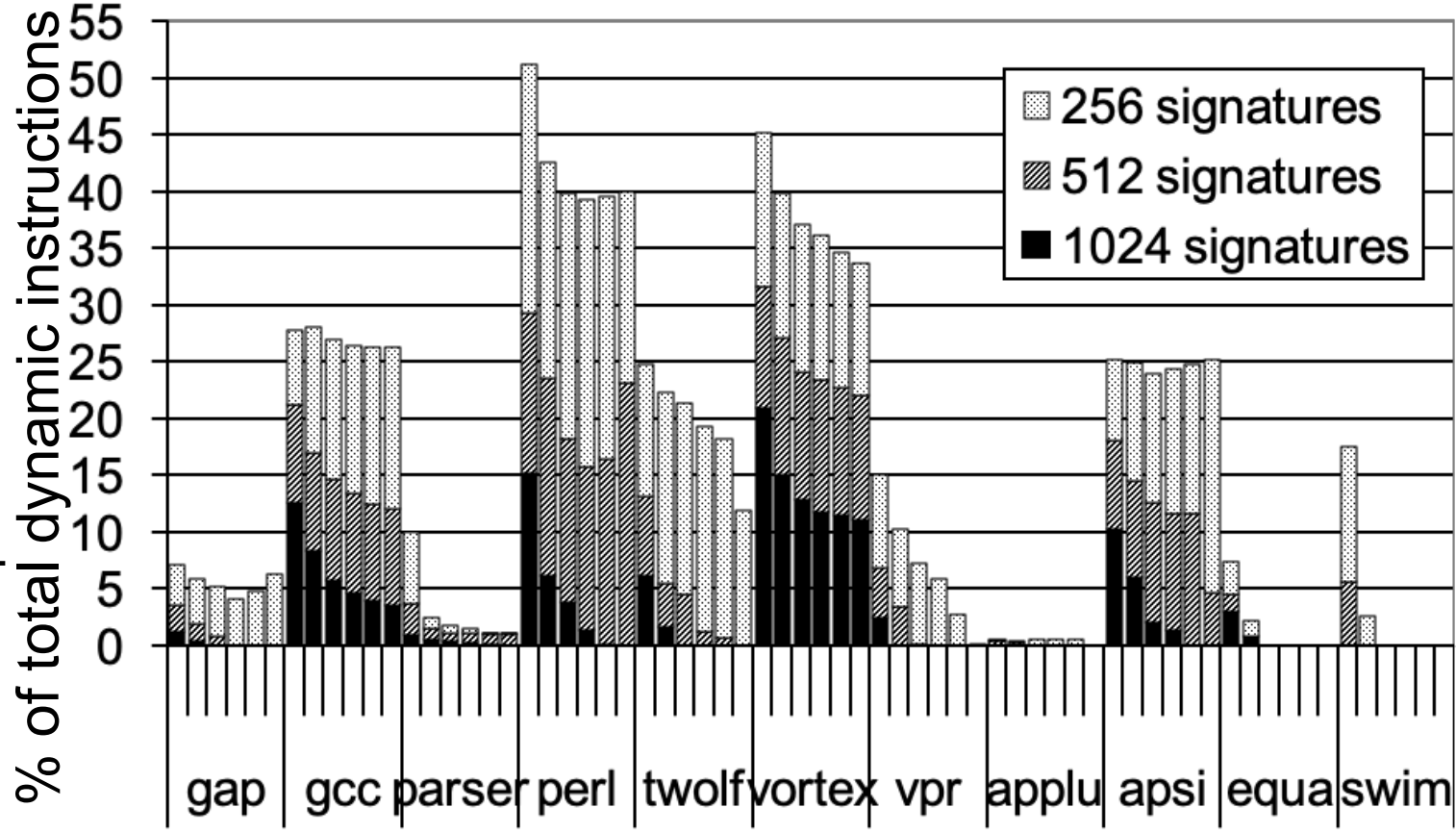
For 2-way, 1024 signatures: 8% max. (vortex) and 1.3% avg. loss in detection coverage



Results: Loss in Fault Recovery Coverage

Recovery coverage loss is bigger than loss in detection coverage

For 2-way, 1024 signatures: 15% max. (vortex) and 2.5% avg. loss in recovery coverage



Outline

- Exploiting ITR for Fault Tolerance
- ITR Cache for Checking Decode Signals
- ITR Cache Hit/Miss Scenarios
- Microarchitecture Extensions to Superscalar
- Results
  - Repetition
  - Fault Coverage Based on Cache Hits/Misses
  - Fault Coverage Based on Fault Injection
- Area and Power Overhead
- Conclusion

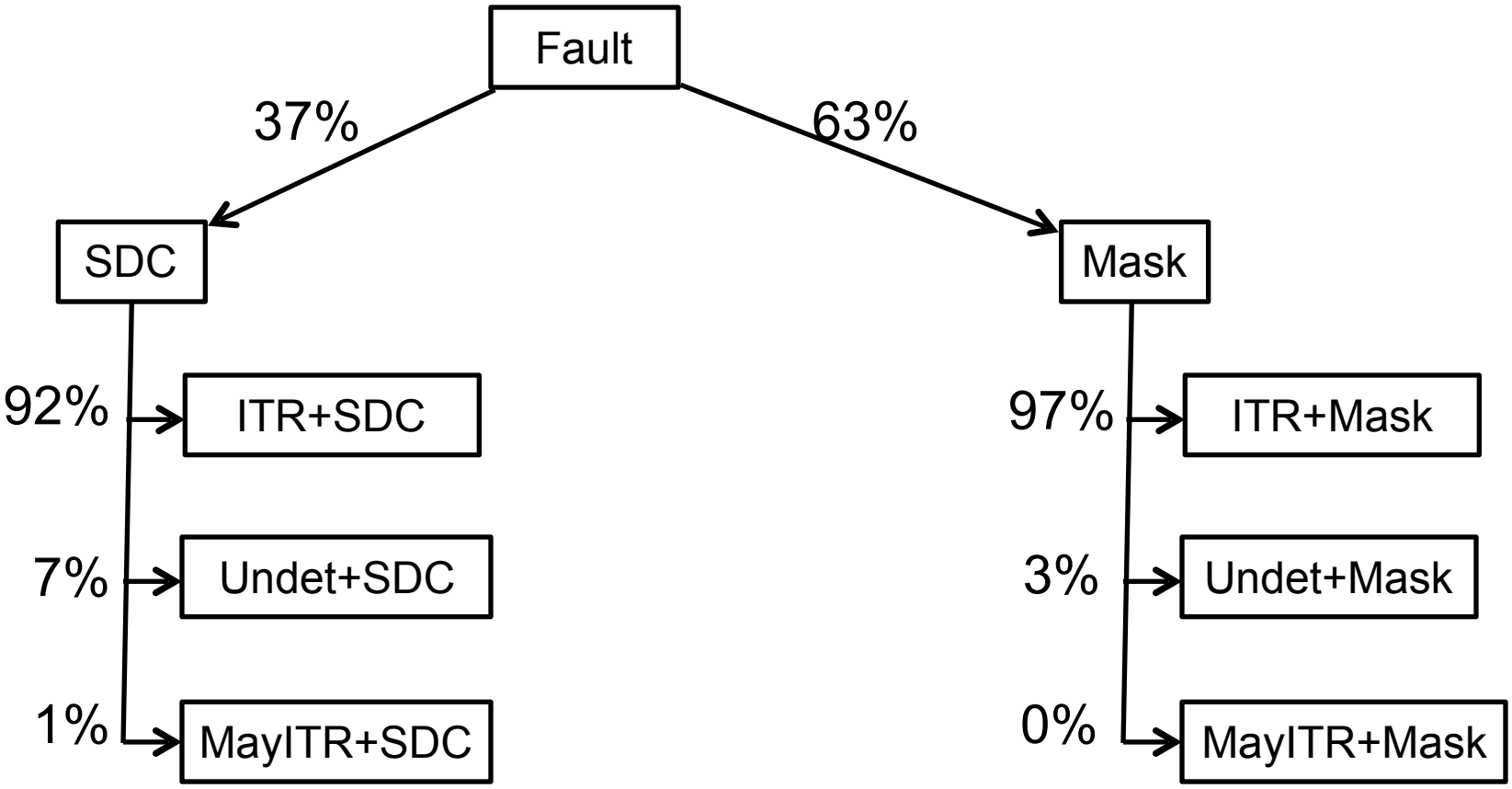
Fault Injection Experiments

- Injected faults on decode signals of a cycle-accurate simulator
- Based on MIPS R10K processor
- 1000 random faults per benchmark

Decode Signals Modeled

Field	Description	Width
opcode	instruction opcode	8
flags	decoded control flags (is_int, is_fp, is_signed/unsigned, is_branch, is_uncond, is_ld, is_st, mem_left/right, is_RR, is_disp, is_direct, is_trap)	12
shamt	shift amount	5
rsrc1	source register operand	5
rsrc2	source register operand	5
rdst	destination register operand	5
lat	execution latency	2
imm	immediate	16
num_rsrc	number of source operands	2
num_rdst	number of destination operands	1
mem_size	size of memory word	3
	Total width	64

Fault Injection Outcomes

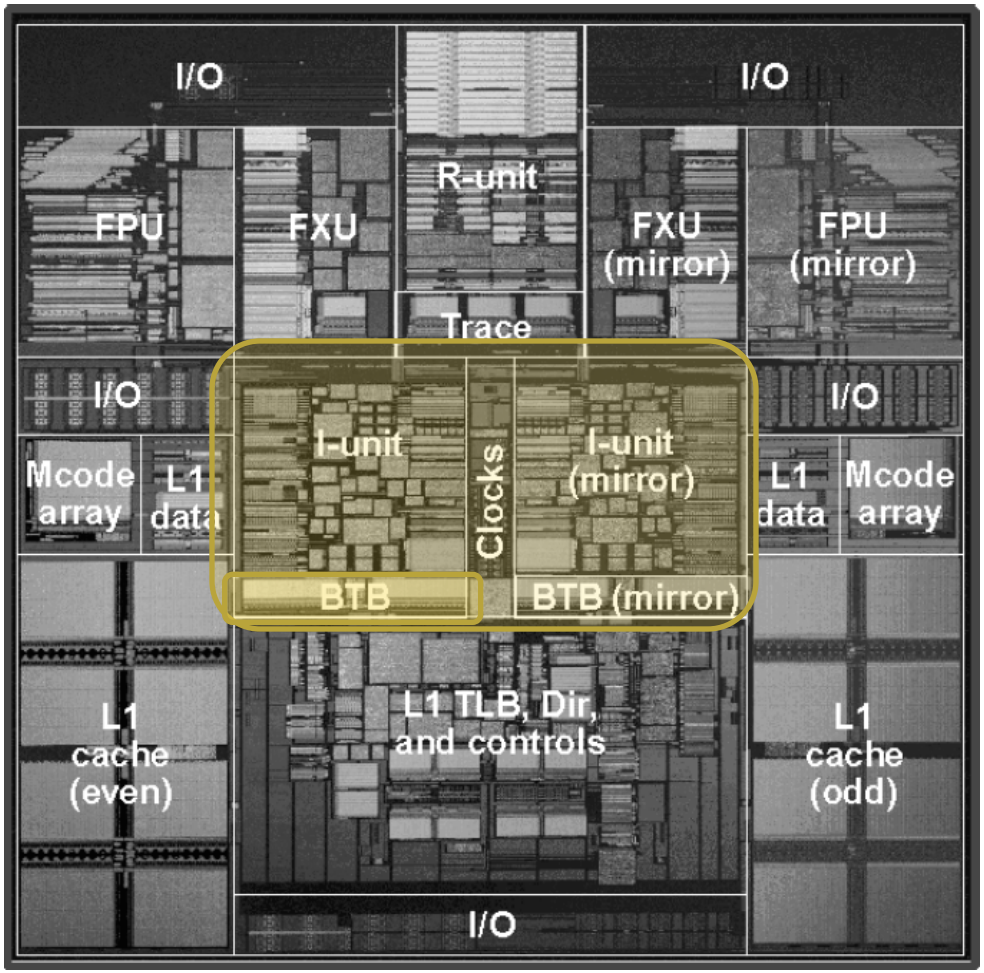


Refer paper for other interesting fault injection results

Outline

- Exploiting ITR for Fault Tolerance
- ITR Cache for Checking Decode Signals
- ITR Cache Hit/Miss Scenarios
- Microarchitecture Extensions to Superscalar
- Results
  - Repetition
  - Fault Coverage Based on Cache Hits/Misses
  - Fault Coverage Based on Fault Injection
- Area and Power Overhead
- Conclusion

Area Overhead

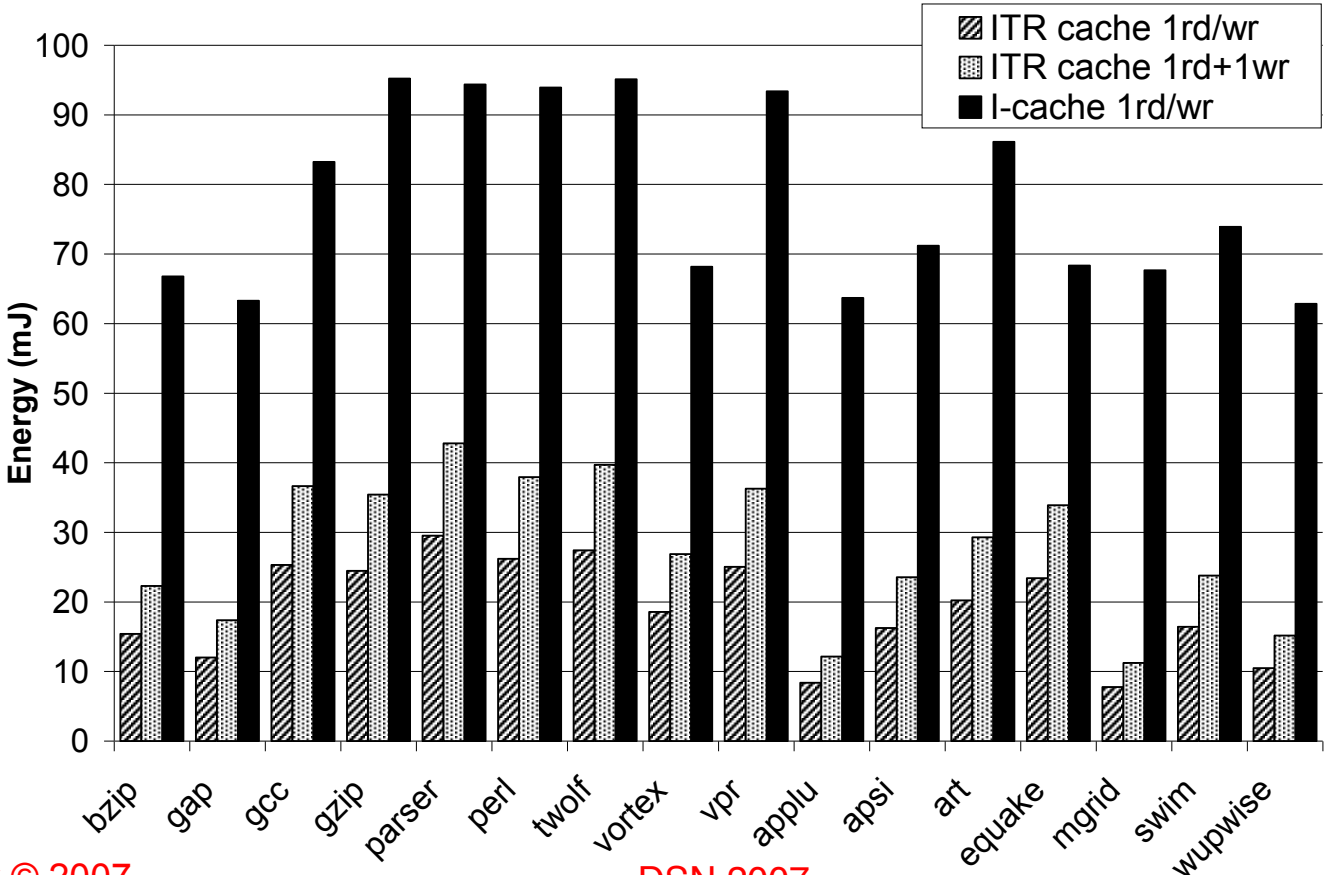


- The IBM S/390 G5 replicates the fetch and decode units (I-unit)
- ITR cache configuration similar to BTB in the picture (2K entries, 2-way assoc., 35 bits per entry)
- ITR cache is 1/7th the I-unit

*From [slegel-IEEE-Micro-1999]

Power Overhead

- Redundant fetching is a major power overhead
- Compare I-cache of IBM power4 (64KB, dm, 128B line, 1 rd./ wr. port) with ITR cache (16KB, 2-way, 4B line, 1 rd. and 1wr. port)
- Excluded redundant decoding energy (more savings in reality)



Outline

- Exploiting ITR for Fault Tolerance
- ITR Cache for Checking Decode Signals
- ITR Cache Hit/Miss Scenarios
- Microarchitecture Extensions to Superscalar
- Results
  - Repetition
  - Fault Coverage Based on Cache Hits/Misses
  - Fault Coverage Based on Fault Injection
- Area and Power Overhead
- Conclusion

Conclusion

- Inherent Time Redundancy is a new opportunity for efficient fault tolerance
- 96% of faults on decode signals detected by small ITR cache (~16 KB)
- Future work:
  - Handling benchmarks with less repetition
  - Exploiting ITR for other input-independent parts of the processor
  - Evaluating a comprehensive fault regimen that includes ITR