Karlsruhe Institute of Technology

ASIA SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE

CLASS: Combined Logic Architecture Soft Error Sensitivity Analysis

Mojtaba Ebrahimi, Liang Chen, Hossein Asadi, and Mehdi B. Tahoori

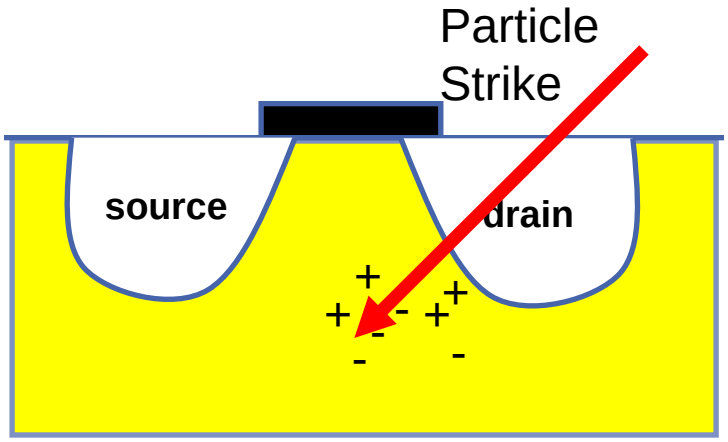
INSTITUTE OF COMPUTER ENGINEERING (ITEC) – CHAIR FOR DEPENDABLE NANO COMPUTING (CDNC)

Soft Errors

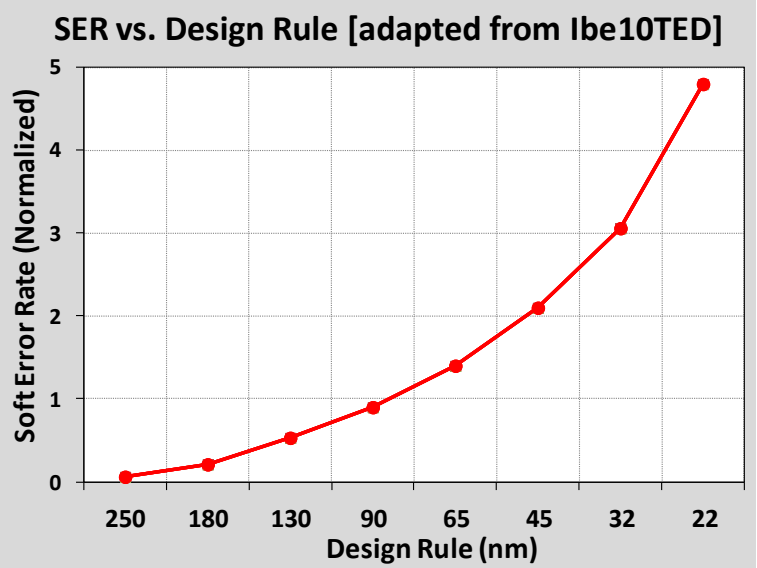
- Soft Errors caused by radiation-induced particle strikes
  - Strike releases electron & hole pairs
  - Absorbed by source & drain
  - Alter transistor state

- Soft error at logic-level and higher:
  - Bit-flip in flip-flops
  - Transient pulses in combinational gates

- System Soft Error Rate (SER) grows exponentially with Moore's law
  [Dixit11IRPS][Ibe10TED]
  - Number of transistors per system is proportional to SER



Transistor Device



Motivation: Importance of SER Estimation

- Not all soft errors cause system failures
  - Circuit level masking (logical, timing, and electrical)
  - Architecture level masking
  - Application level masking

- Nodes not equally susceptible to soft errors
  - Finding most susceptible modules
  - Applying selective protection schemes
  - Balancing reliability, cost, and performance

- SER Estimation technique is needed
  - Rank the nodes according to their SER

Motivation: What is missing?

- Previous Soft Error Rate (SER) estimation techniques
  - Fault injection (time consuming)
  - Analytical (accuracy)
    - **Architecture-level techniques** → regular structures
      (register-file, cache)
    - **Circuit-level techniques**→ irregular structures
      (controller unit, pipeline)
    - Microprocessors include both regular and irregular structures
    - Independent analysis is not accurate
- **Proposed**: CLASS: Combined Logic Architecture Soft error Sensitivity analysis
  - Combines a circuit-level technique with an architecture-level analysis
  - Discrete time Markov chains for handling different error propagation and masking scenarios

Outline

- Preliminaries
- CLASS
- Experimental Results
- Conclusions

Outline

- Preliminaries
- CLASS
- Experimental Results
- Conclusions

Analytical Architecture Level SER Estimation Techniques

- ACE (Architecturally Correct Execution) analysis is the most common technique [Mukherjee03Micro]
  - a.k.a Life-time analysis
- Lifetime of a bit is divided into :
  - ACE
    - e.g. Program counter almost always in ACE state
  - un-ACE (unnecessary for ACE)
    - e.g. Branch predictor always in un-ACE state
- AVF (Architecture Vulnerability Factor)
  - Fraction of time that system is on ACE state
    - Program counter ~ 100%, Branch predictor = 0%
- ACE analysis overestimates failure rate
  - In some cases up to 7x [George10DSN] → Overdesign

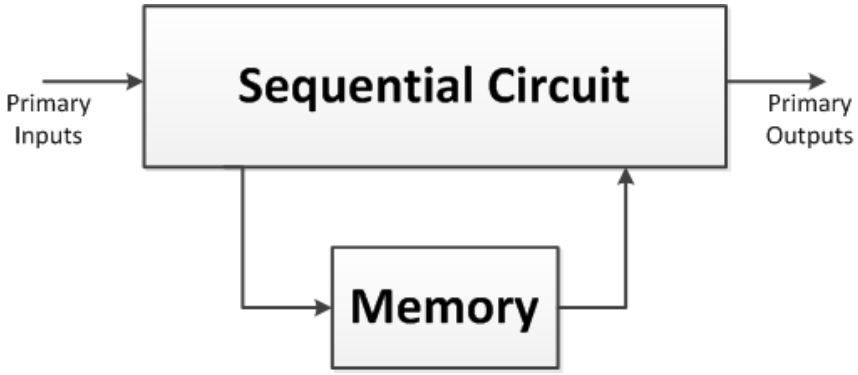
Analytical Circuit Level SER Estimation Techniques

- Binary/Algebraic Decision Diagrams (BDD/ADD)-based techniques [Zhang06ISQED] [Norman05TCAD] [Krishnaswamy05DATE] [Miskov08TCAD]
  - Based on decision diagrams
  - Highly accurate
  - Scalability problem

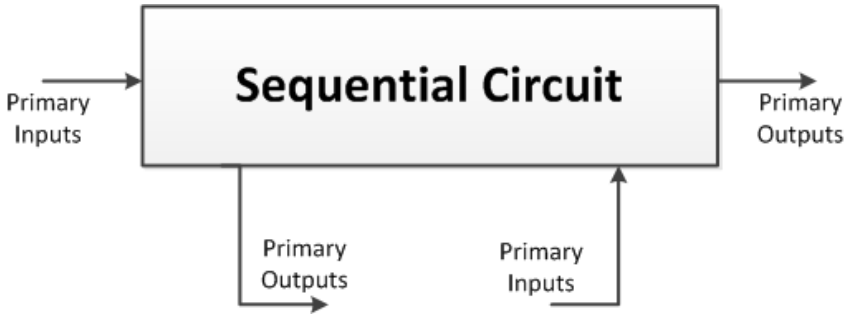
- Error Probability Propagation (EPP)-based techniques [Asadi06ISCAS] [Fazeli10DSN] [Chen12IOLTS]
  - Based on propagation rules
  - Reasonable accuracy
  - Scalable

Shortcoming of Previous techniques

- Circuit-level techniques consider:



A circuit



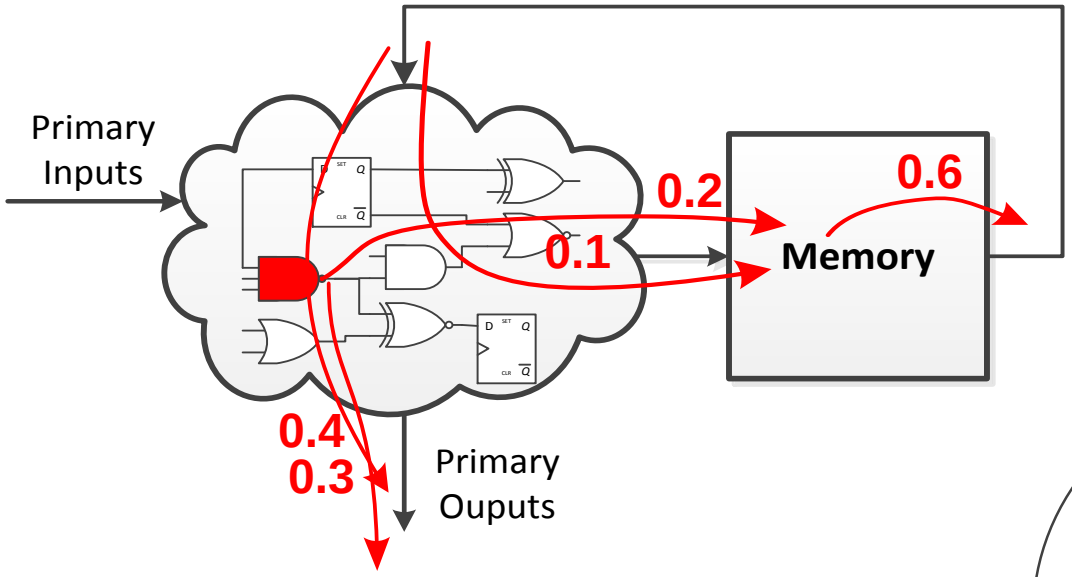
What seen by circuit level techniques

- ACE cannot completely model effect of irregular structures in error propagation from regular structures

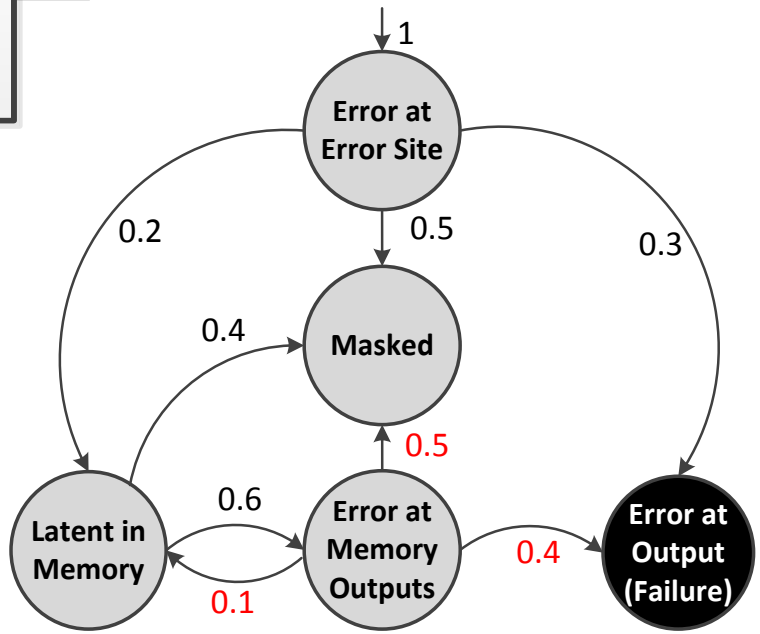
Overview

- Preliminaries
- CLASS
- Experimental Results
- Conclusions

CLASS: A Simple Example

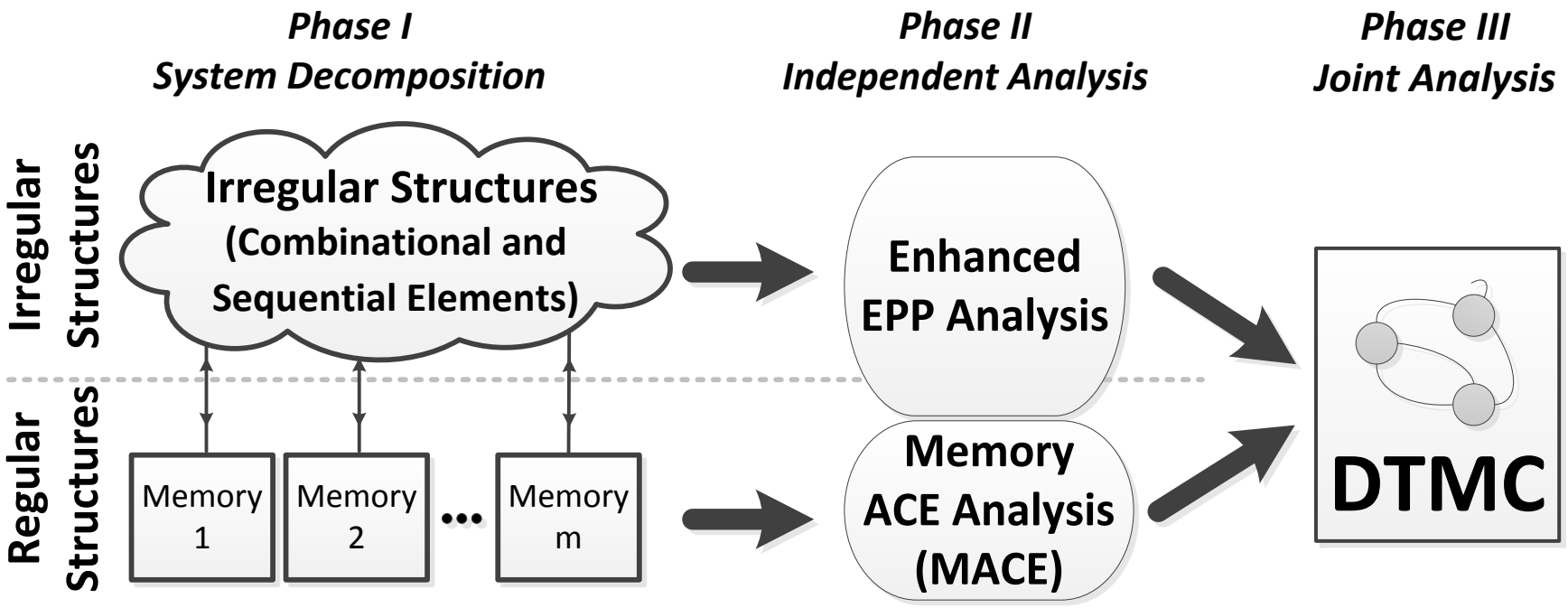


- Propagation from memory elements to memory elements (Iterative)
  - Saturates in few iterations



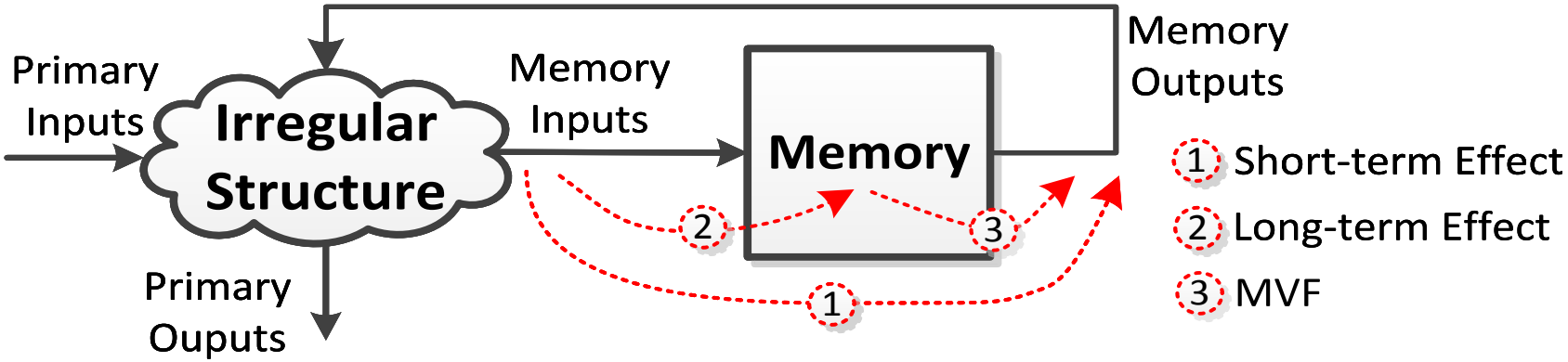
Discrete Time Markov Chain (DTMC)

Overview of CLASS

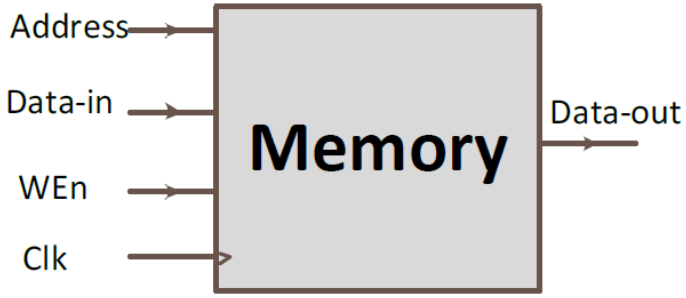


Important Error Propagations

- Propagation from irregular structures to regular structures and vice versa :
  1. **Short-term effect**: error in memory inputs affects its outputs immediately
  2. **Long-term effect**: error in memory inputs contaminates its contents
  3. **MVF (Memory Vulnerability Factor)**: propagation probability of an error from memory contents to its output



Computation of Short- and Long-term Effects



A simple memory

Error at Port	Probability of Error			
	None	Output (Short-term)	Contents (Long-term)	Both
Address	0	1-SP(WEn)	SP(WEn)	0
Data-in	1-SP(WEn)	0	0	SP(Wen)
WEn	0	0	0	1

Probability of having short- and long-term effect for each input port of memory

Short-term effect

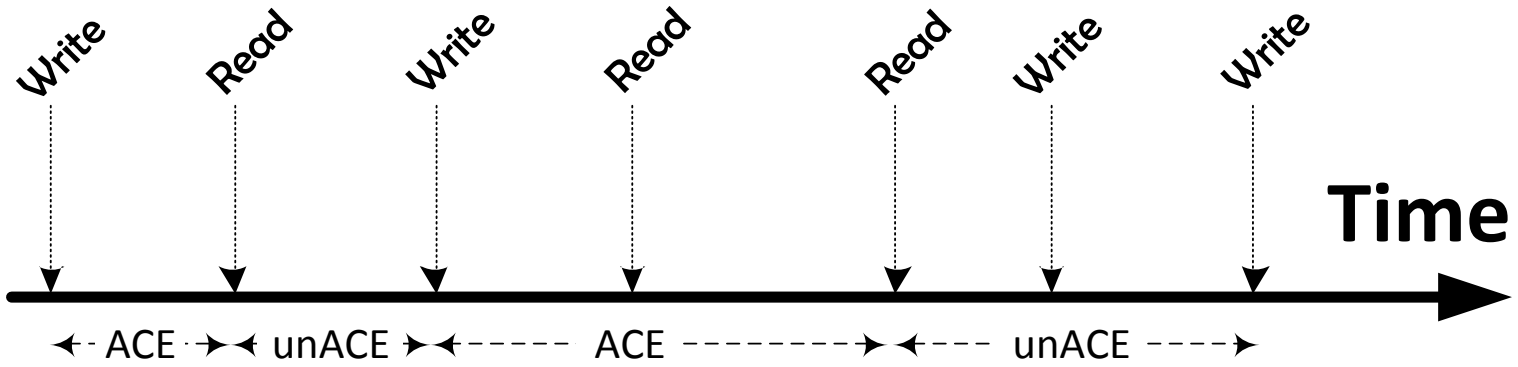
$$P_{ST} = [1 - SP(WEn)] \times P_e(Address) + SP(WEn) \times P_e(Data\text{-}in) + P_e(WEn)$$

Long-term effect

$$P_{LT} = SP(WEn) \times P_e(Address) + SP(WEn) \times P_e(Data\text{-}in) + P_e(WEn)$$

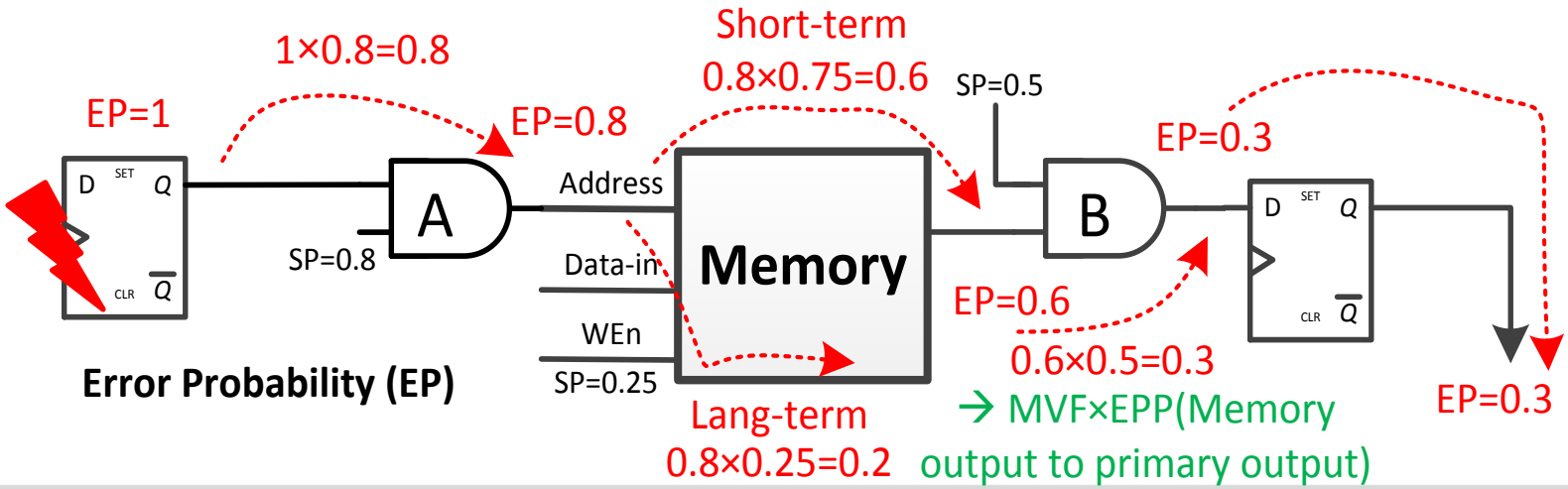
Memory ACE (MACE)

- Looking at memory boundaries
- Intervals leading to a read access are ACE
- MVF_{Word}: Fraction of time that it has an ACE state
- MVF_{Memory} : Average of all MVF_{Word}

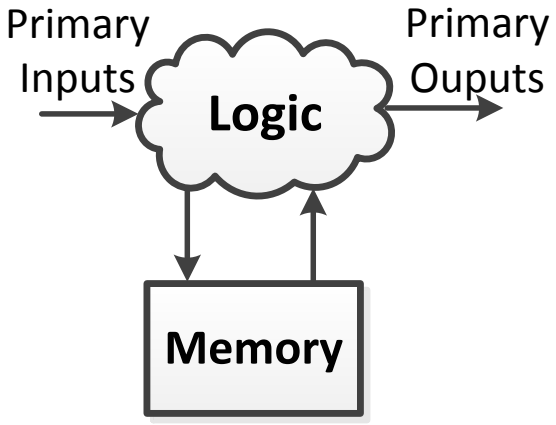


Enhanced Circuit-level Technique

- CLASS is compatible with different circuit-level techniques
  - It inherits characteristics of employed circuit-level technique
  - Multi-cycle 4-value EPP [Fazeli10DSN] is employed
    - Based on topological traverse of netlist
    - O(n^2), Inaccuracy within 2%
- Circuit-level technique should be enhanced to:
  - Model short-term effect of error in memories
  - Compute long-term effect probability (treated as independent error)

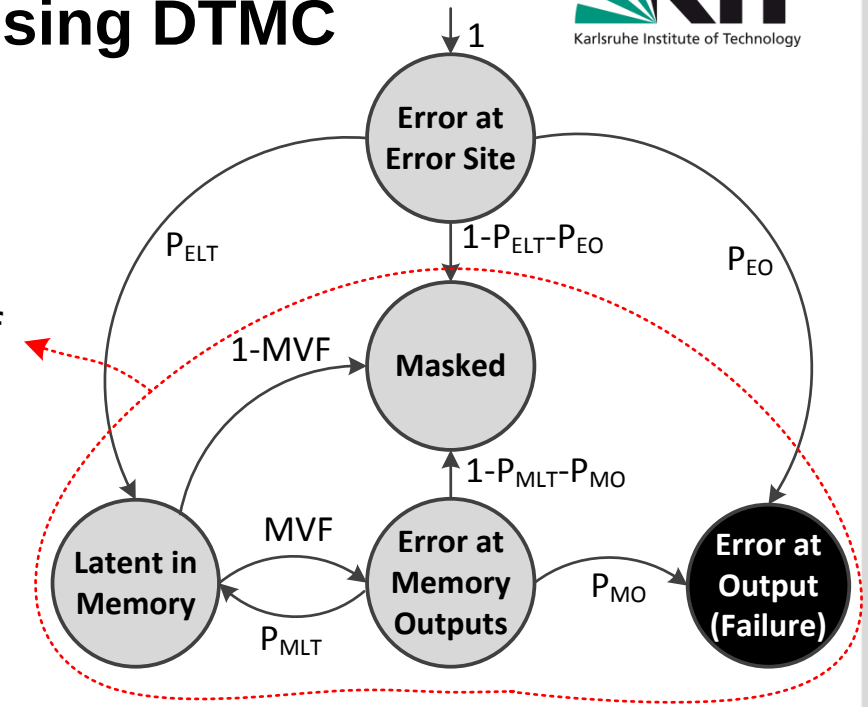


Integration of MACE and EPP Using DTMC



A simple system

Independent of error site



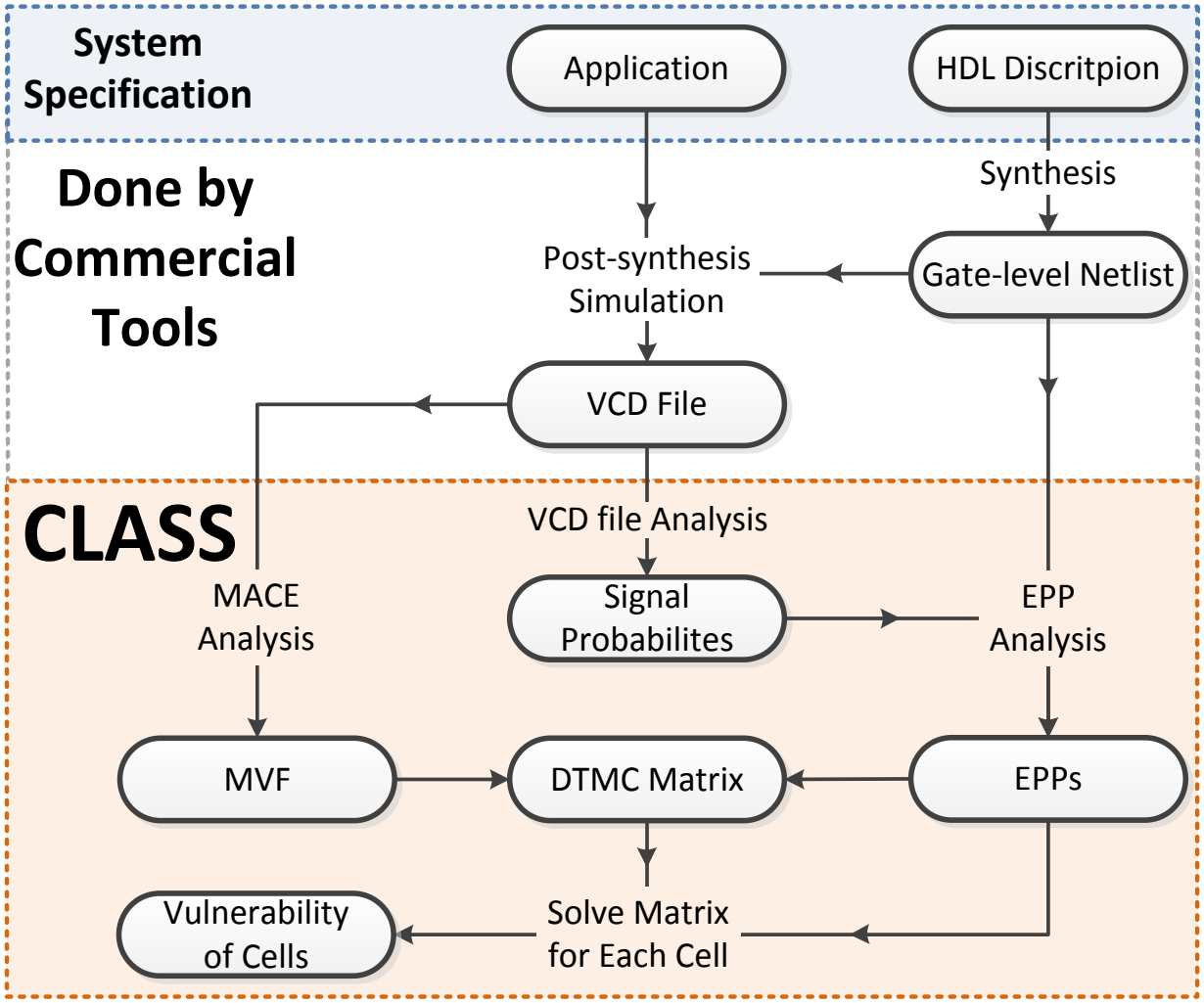
Error propagation DTMC

Symbol	Definition
P_{EO}	P(Error site → Primary Outputs)
P_{ELT}	P(Error site → Latent in Memory)
P_{MO}	P(Memory Output → Primary Outputs)
P_{MLT}	P(Memory Output → Latent in Memory)
MVF	P(Latent in Memory → Memory Outputs)

Steady state solution of DTMC

$$P_f = P_{EO} + \frac{P_{ELT} \times MVF \times P_{MO}}{1 - MVF \times P_{MLT}}$$

Overall Flow

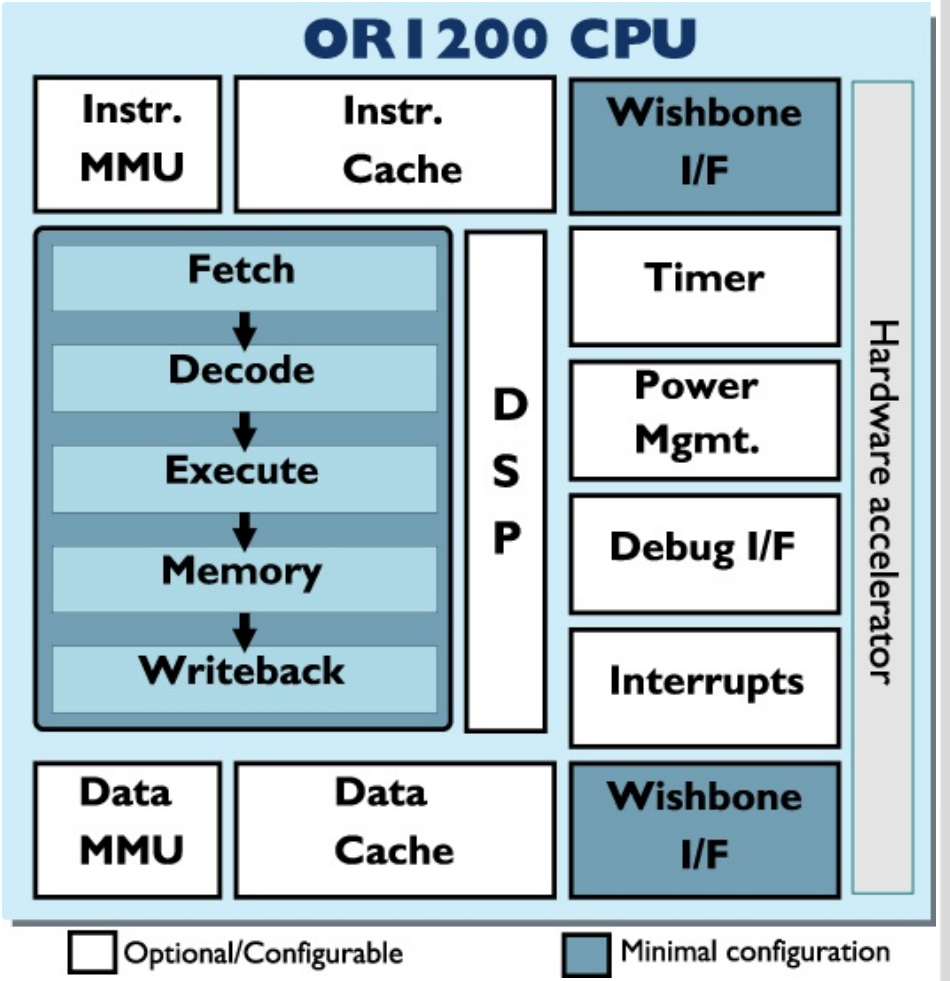


Outline

- Preliminaries
- CLASS
- Experimental Results
- Conclusions

Experimental Setup

- OpenRISC 1200 processor
- Synthesis results:
  - 44,480 gates
  - 4,960 flip-flops
  - 4 Memory Units
  - Overall: 49,444 error sites
- Benchmarks are selected from Mibench:
  - QSort, CRC32, Stringsearch



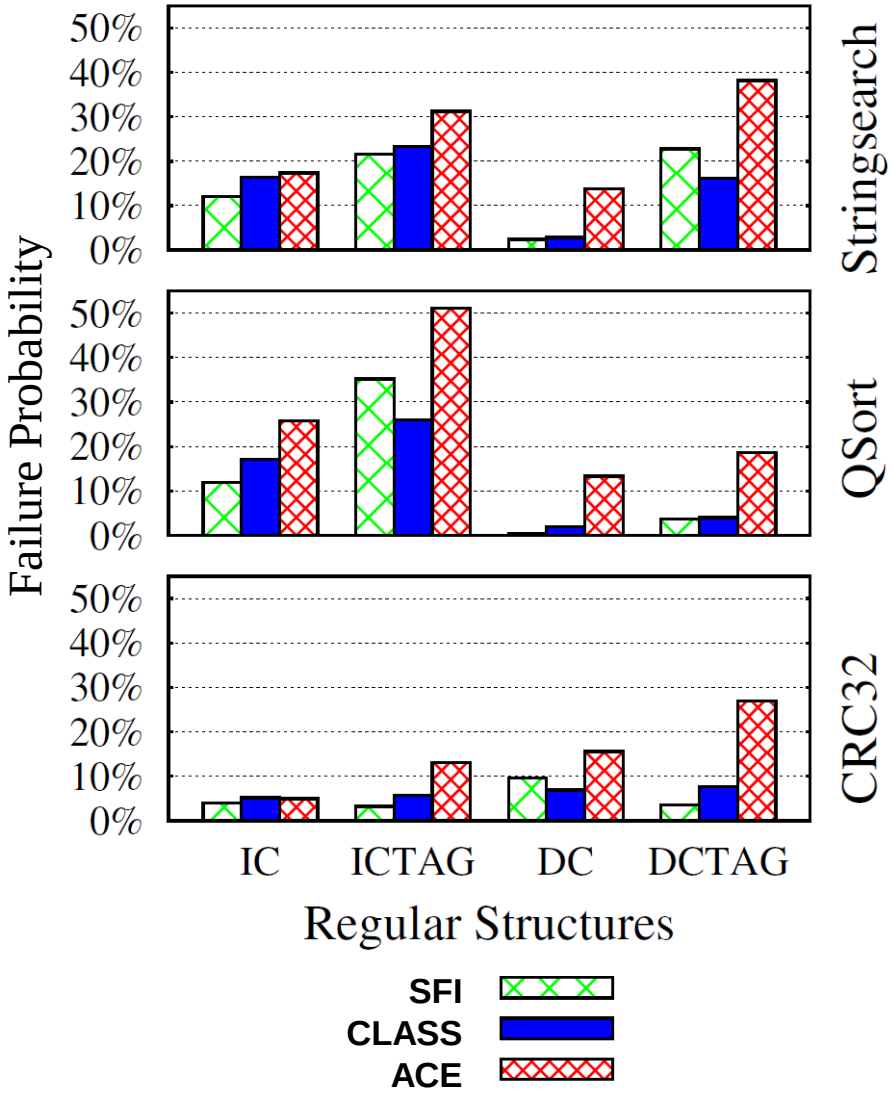
[www.opencores.org]

Comparison with Other Techniques

- **EPP** [Asadi06ISCAS][Fazeli10DSN]
- **ACE**[Biswas05ISCA]
  - implemented for caches and their tags
- **Simulation-based Statistical Fault Injection (SFI)**
  - Faults are injected during post-synthesis simulation
  - One fault per running entire application
  - Each FI on average takes 87 seconds
  - Only logic masking is considered
    - Comparable with functional simulation
- System configuration:
  - Intel Xeon E5520
  - 16 GB RAM

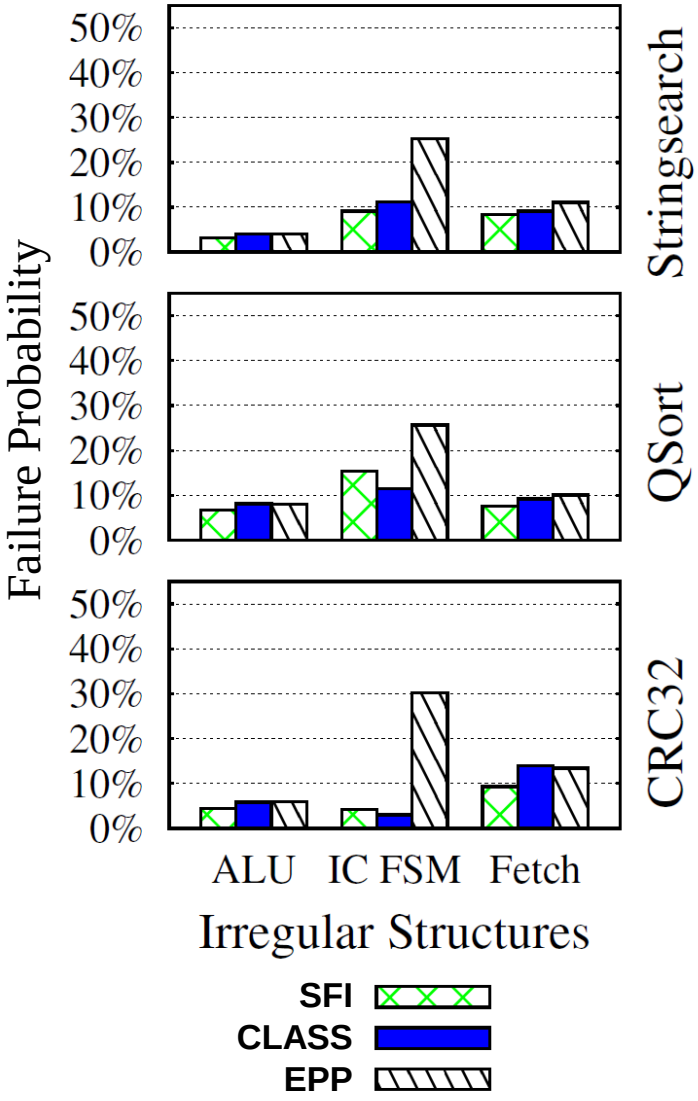
Inaccuracy Analysis

- Regular structures:
  - OpenRISC Regular Structures
    - Instruction Cache (IC)
    - Instruction Cache Tag (ICT)
    - Data Cache (DC)
    - Data Cache Tag (DCT)
  - CLASS
    - Average inaccuracy: 3.4%
    - Maximum inaccuracy : 9.2%
  - ACE
    - Average inaccuracy : 11.7%
    - Maximum inaccuracy : 23.3%
- CLASS may overestimate or underestimate
  - Due to inaccuracy of 4-value EPP



Inaccuracy Analysis

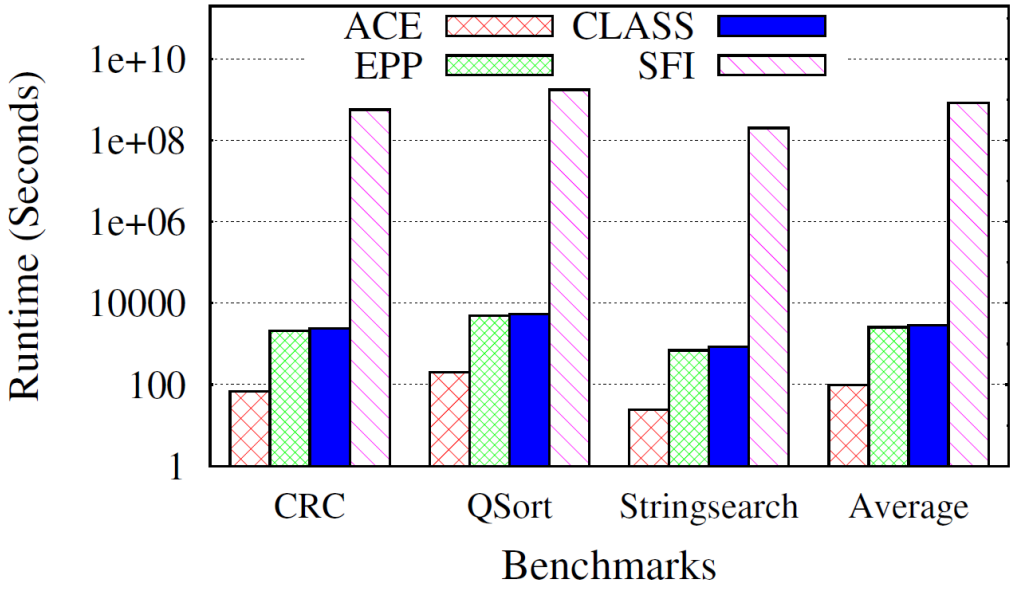
- Irregular structures:
  - Three representative blocks of OpenRISC:
    - ALU
    - Instruction Cache FSM (IC FSM)
    - Pipeline fetch unit
  - CLASS
    - Average inaccuracy : 2.1%
    - Maximum inaccuracy : 4.0%
  - EPP
    - Average inaccuracy : 7.2%
    - Maximum inaccuracy : 26.3%
- Average inaccuracy of individual error sites is less than 7%.



Runtime

- Individual inaccuracy of 7% needs 196 faults at each error site
  - 49,444 possible error sites
  - 49,444 x 196 x 87 → more than 27 years
- CLASS is 5 orders of magnitude faster than simulation-based SFI → Less than one hour

- Runtime Breakdown:
  - VCD Analysis 85%
  - EPP 14%
  - Solving DTMCs 1%



Outline

- Preliminaries
- CLASS
- Experimental Results
- Conclusions

Conclusion

- CLASS: SER estimation technique for entire systems
  - Unlike existing techniques which only focused on either regular or irregular structures
- Interactions between regular and irregular structures are considered
- Compared to SFI:
  - Inaccuracy is less than 7%
  - 5 orders of magnitude faster
- Multiple times more accurate than EPP (4x) and ACE (3x)

Karlsruhe Institute of Technology

Thanks for your attention!

Q & A

Backup Slides

Sources of Inaccuracy

- Sources of inaccuracy in current implementation
  - Propagation of an error to several memory units
  - Multiple propagation of an error to memory outputs

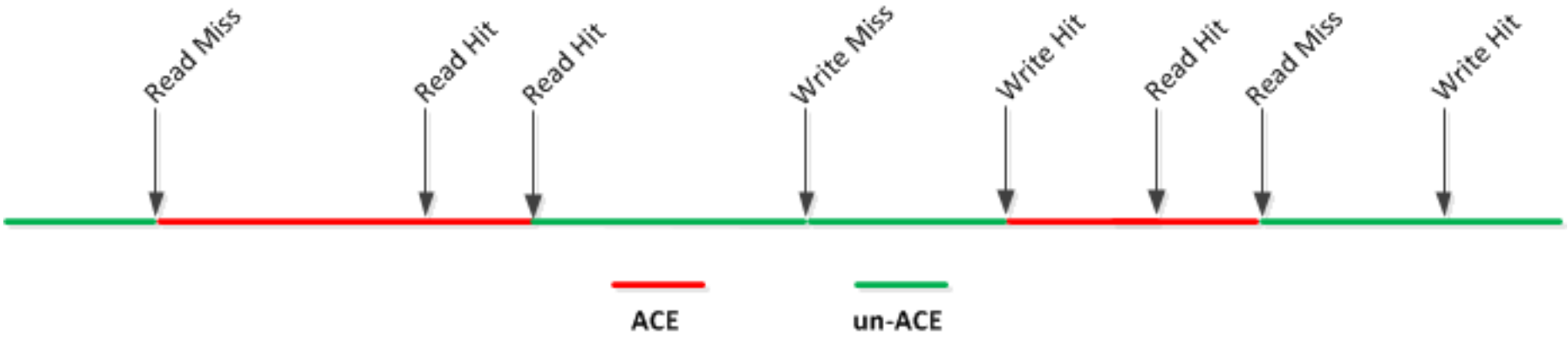
ACE Analysis: Instruction Cache

- Access Types: Read Miss (RM) and Read Hit (RH)

- Un-ACE Intervals * →RM
- ACE Intervals: *→RH

ACE Analysis: Data Cache (Write-back)

- Access types: Read Miss (RM), Read Hit (RH), Write Miss (WM), Write Hit (WH)
- ACE intervals: *→RH
- Un-ACE intervals: *→WH
- Potentially ACE intervals: *→WM, *→RM



Enhanced EPP

- Short-term effect of error in memories
- Long-term effect are considered as propagated as independent error
- Enhanced to calculate the probability of error propagation to memory inputs

$$P_{Mem \cap \bar{PO}} = P_{Mem \cup PO} - P_{PO}$$

- Only logic masking is considered
- Multi-cycle 4-value EPP [Fazeli10DSN] is employed
  - CLASS has no limitation regarding circuit-level techniques