Enhanced Fault Coverage Analysis Using ABVFI

Scott Bingham and John Lach
Department of Electrical and Computer Engineering
University of Virginia

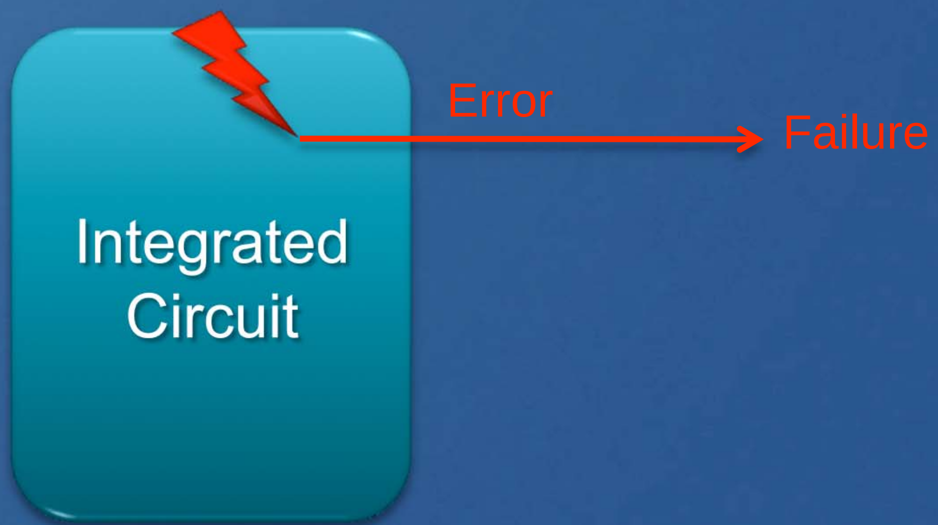
June 29, 2009

UNIVERSITY of VIRGINIA ENGINEERING

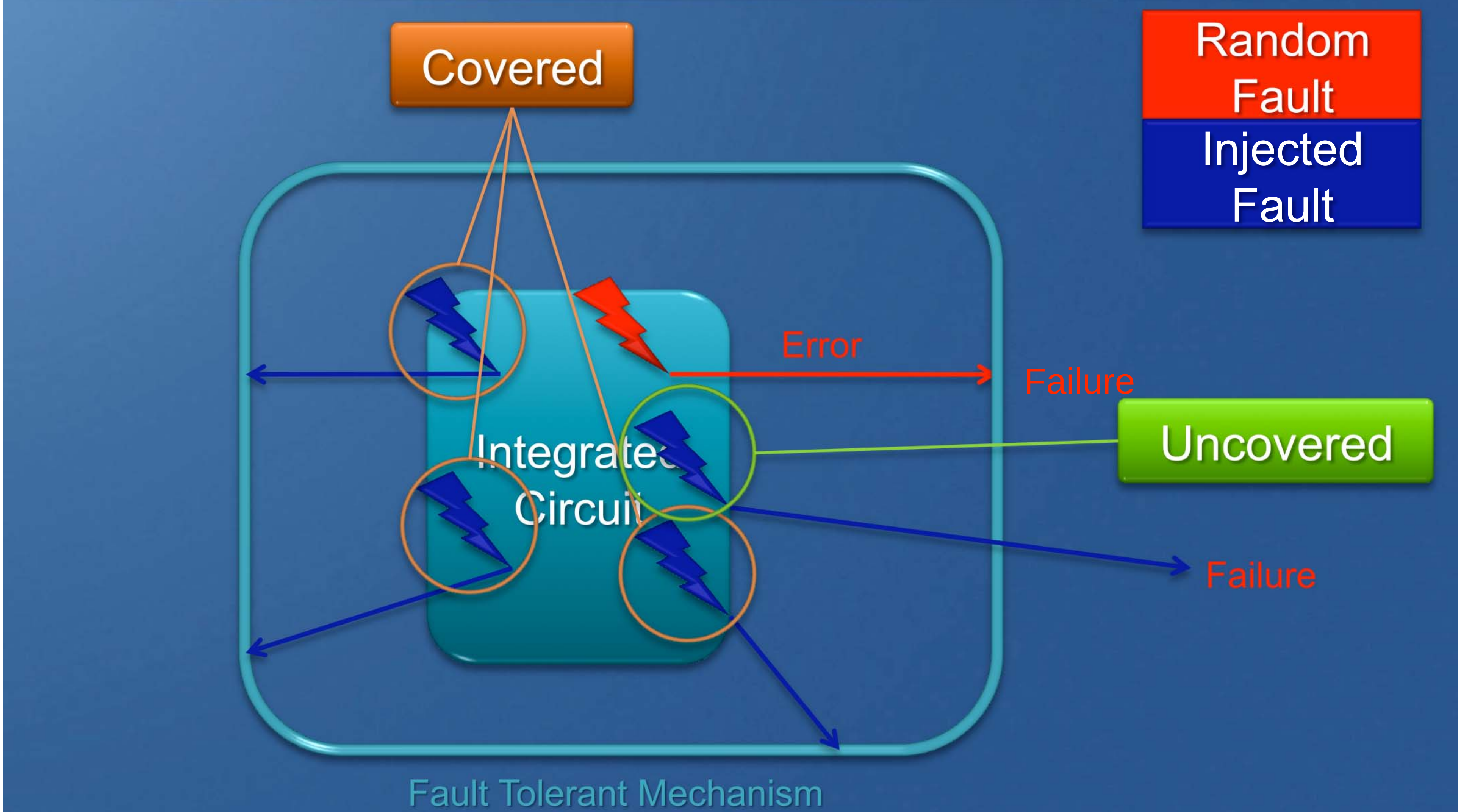
Motivation

- Trends in integrated circuit (IC) manufacturing
  - Transistor sizes are decreasing
  - Transistor counts per die are increasing
- Trends in system design
  - ICs are playing a more central role in systems
  - ICs are being used in safety-critical systems
- ICs are becoming more susceptible to transient faults
- Need for dependable designs
  - For quality products
  - Required by safety-critical systems

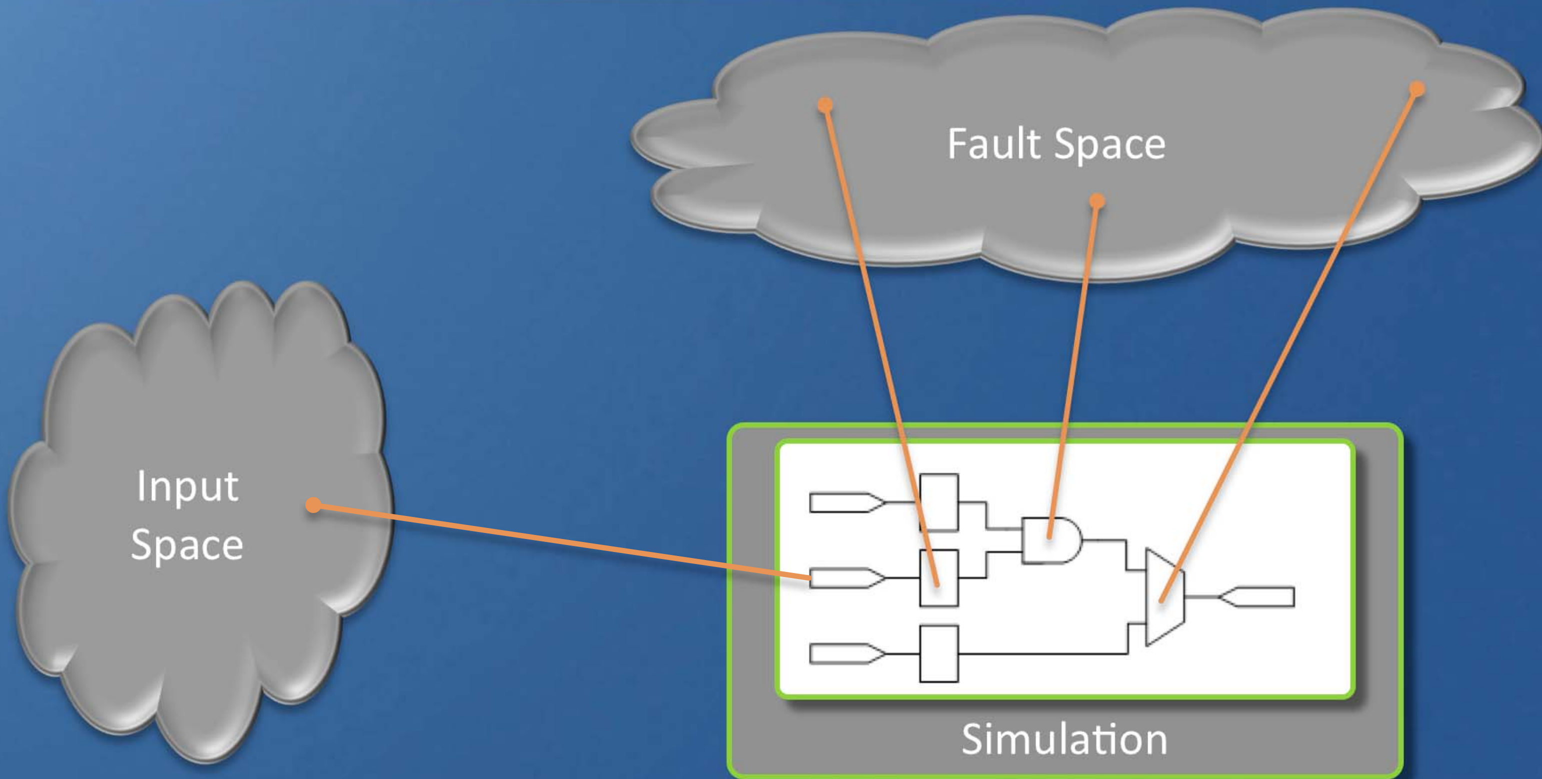
Random Failures Affect Dependability



Determining Dependability through Fault Injection



Fault Injection (continued)



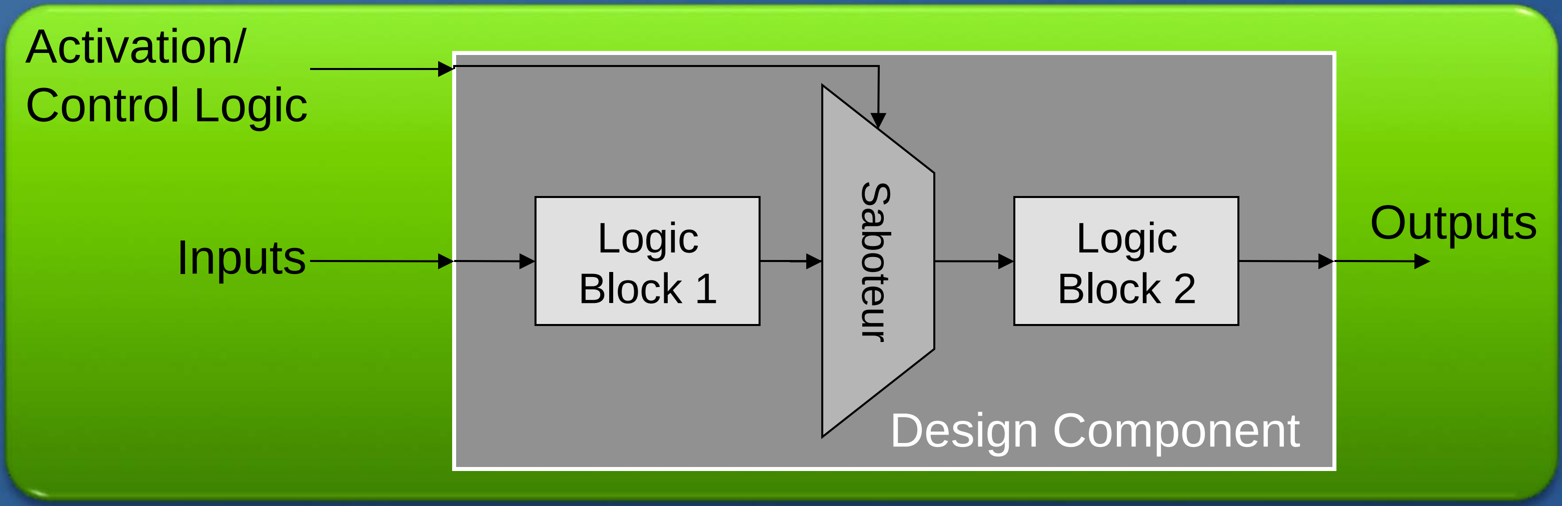
- Because of the size of the input and fault space, it is infeasible to test all possible faults with simulation

State-of-the-practice for ICs

- Simulation-based Fault Injection
  - Performed at the register transfer level (RTL)
  - Alter the design to include logic that mimics the behavior of faults when activated
    - Saboteurs – Modify the value of a design signal
    - Mutants – Change the behavior of a component
  - Benefits
    - High *observability* and *controllability*
    - Can model many types of faults

Simulation-based Fault Injection (cont'd)

- Limitations
  - Only a subset of the fault and input space is tested
    - Simulation is time and computationally expensive
  - Extensive design changes may be required
    - Additional logic
    - Control signals
    - Interface modification
      - Requiring changes to other components
    - It then becomes difficult to justify that the modifications haven't altered the behavior of the design



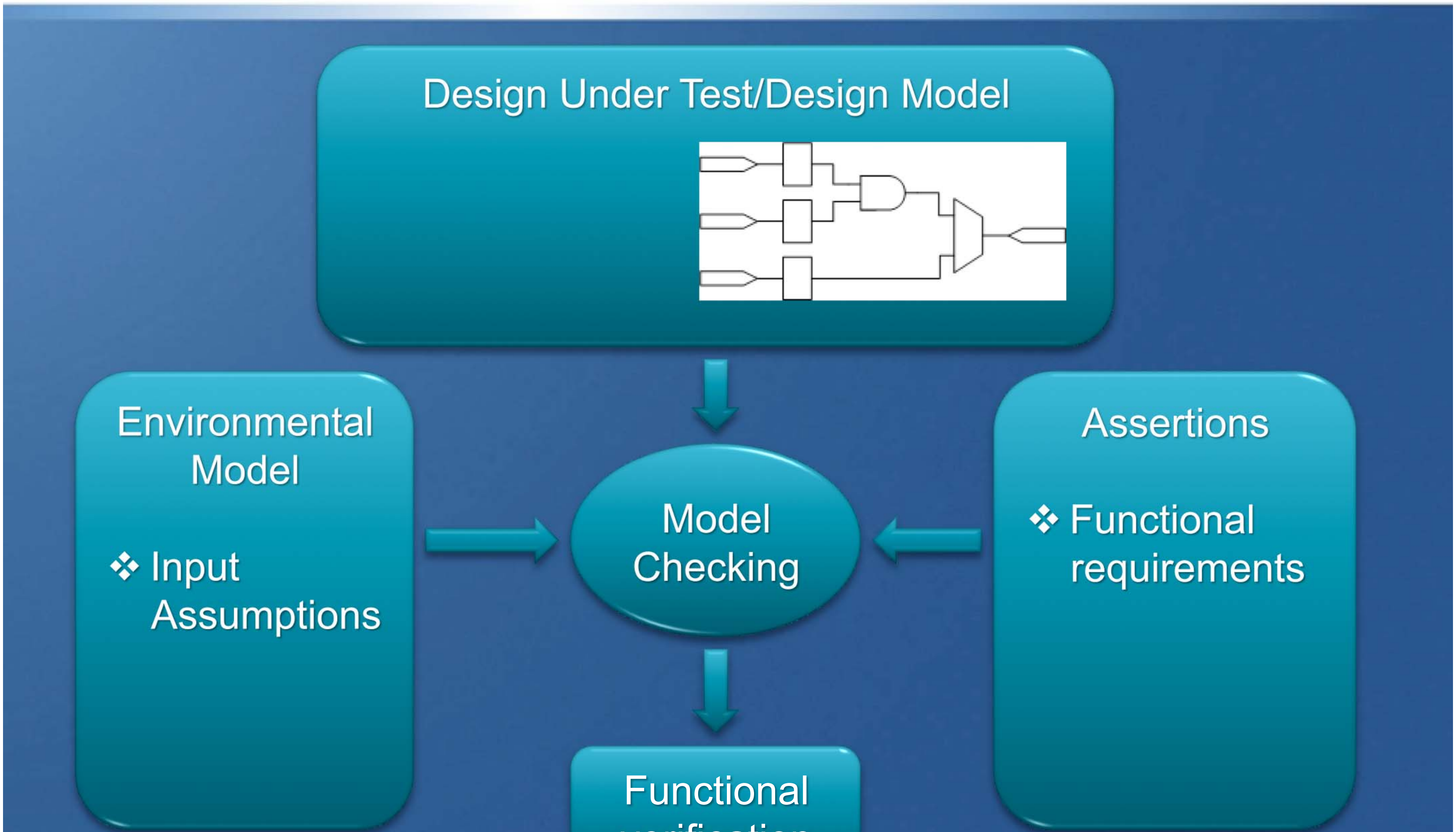
Research Objective

- Develop a fault injection methodology that
  - Includes a more rigorous and complete analysis method than simulation
  - Is able to model relevant faults
    - And be able to model new faults as manufacturing technologies change
  - Provides high observability and controllability
  - Includes a less-invasive design modification
  - Is accessible to designers and verifiers

Outline

- Overview of model checking and ABV
- The ABVFI methodology
- ABVFI implementation
  - Fault Injection Mechanism
    - Design instrumentation
  - The Flexible Fault Framework
- Case Study – The PHFT processor
- Conclusion

Overview of Model Checking and ABV



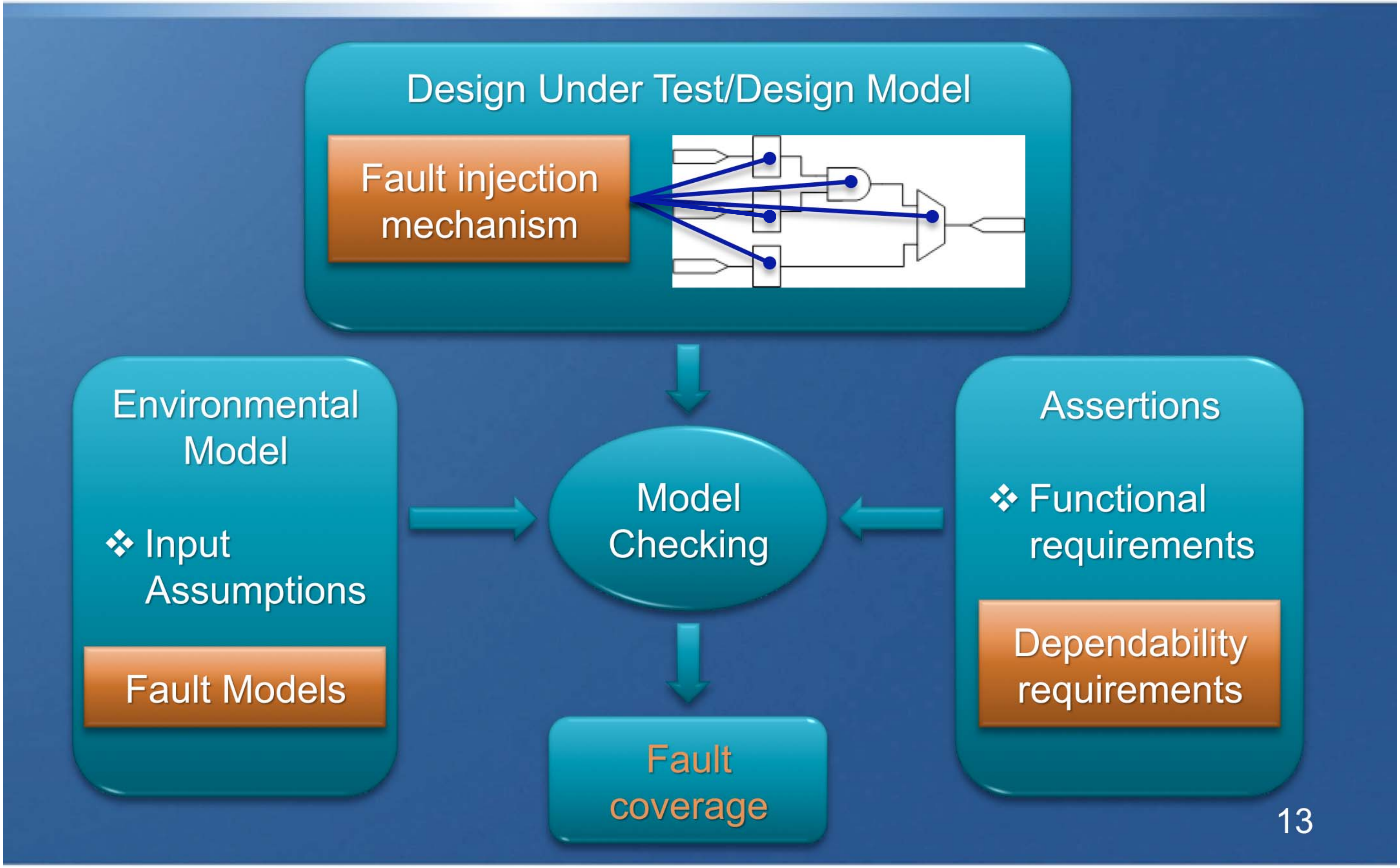
The Assertions

- Assertions are directives that define a property that should be checked
- Properties are propositional statements about the behavior of the design
  - e.g. "signal read and signal write should never be asserted at the same time"
  - Either true or false
- Defined by temporal logics
  - Unambiguous

Outline

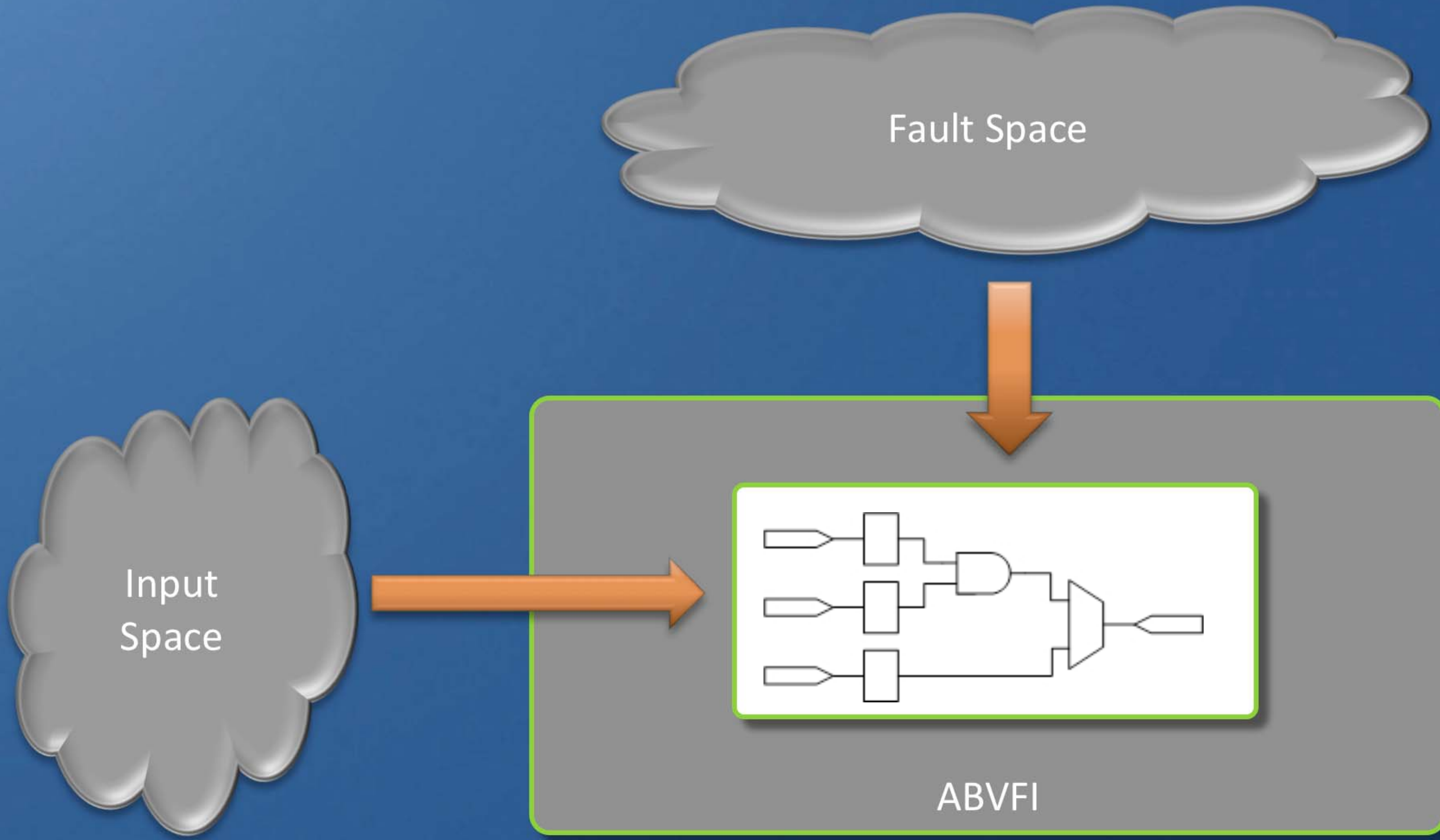
- Overview of model checking and ABV
- **The ABVFI methodology**
- ABVFI implementation
  - Fault Injection Mechanism
    - Design instrumentation
  - The Flexible Fault Framework
- Case Study – The PHFT processor
- Conclusion

ABVFI = ABV + Fault Injection



Fault Injection with ABV

- For each assertion:



An "Exhaustive" Proof

- During verification
  - Define an operational profile
  - Identify the design to be verified
  - Define fault models
- The model is compiled into a mathematical representation
- The mathematical proof is a search of this representation for property violations
  - Essentially an exhaustive simulation!
- "Exhaustive"ness is constrained by:
  - The design
  - The environmental model (the operational profile)
  - Fault models
  - The properties defined

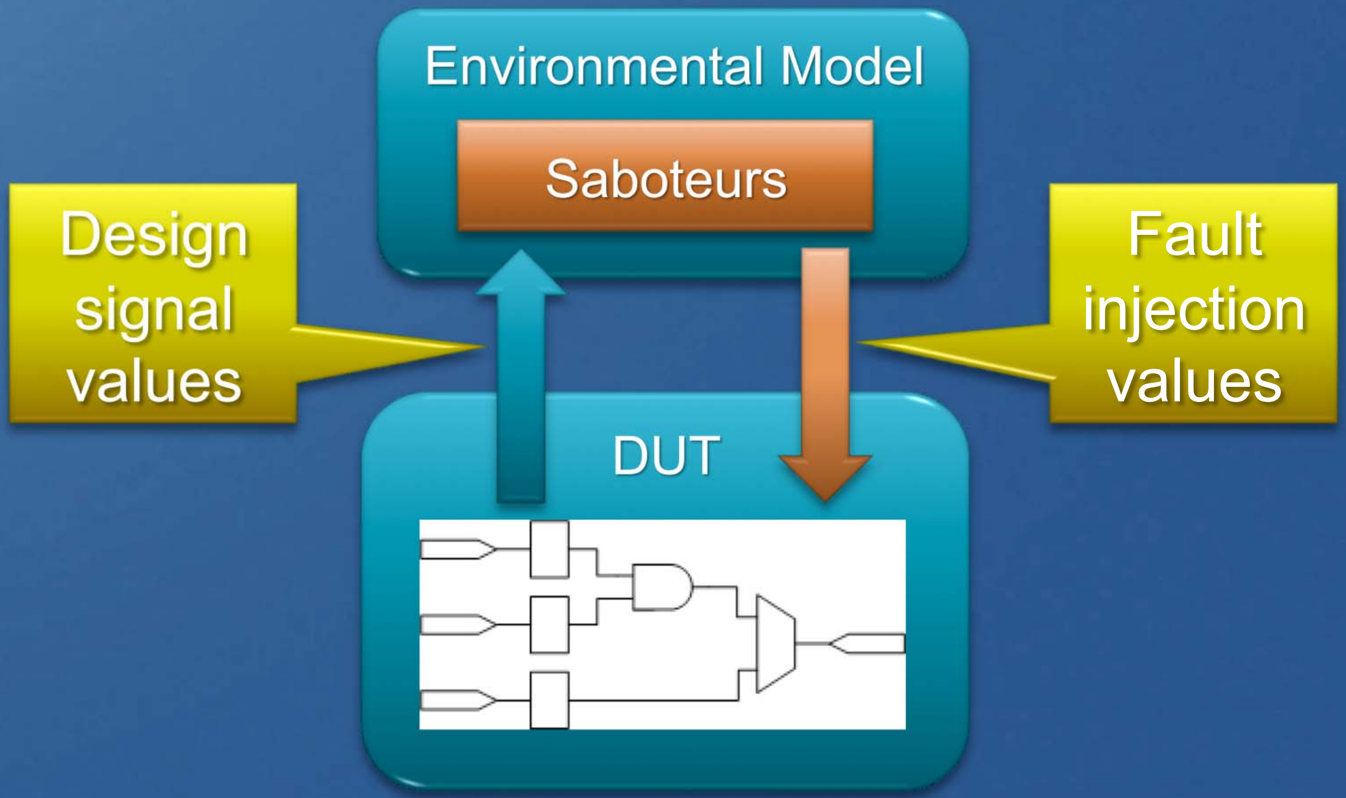
Same as simulation

Outline

- Overview of model checking and ABV
- The ABVFI methodology
- ABVFI implementation
  - Fault Injection Mechanism
    - Design instrumentation
  - The Flexible Fault Framework
- Case Study – The PHFT processor
- Conclusion

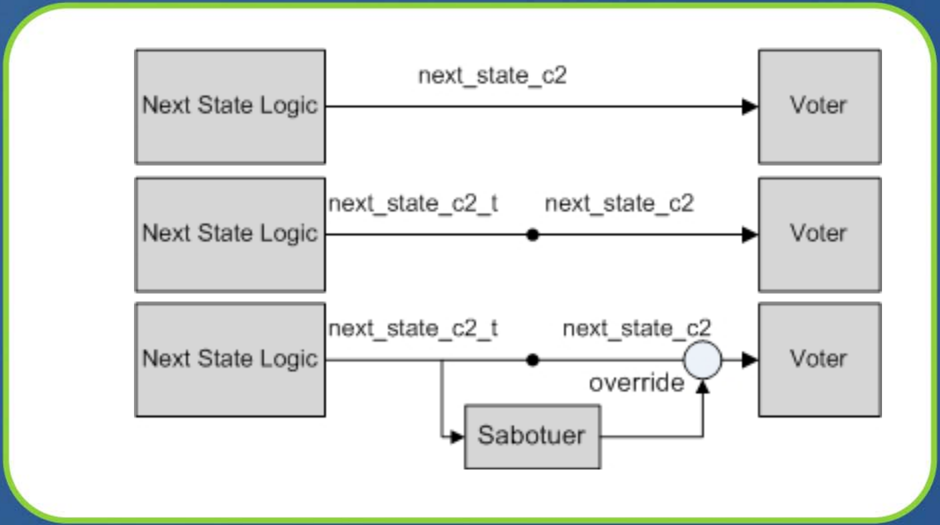
Fault Injection Mechanism

- Minor changes are made to the DUT
- Saboteurs are included in the environmental model



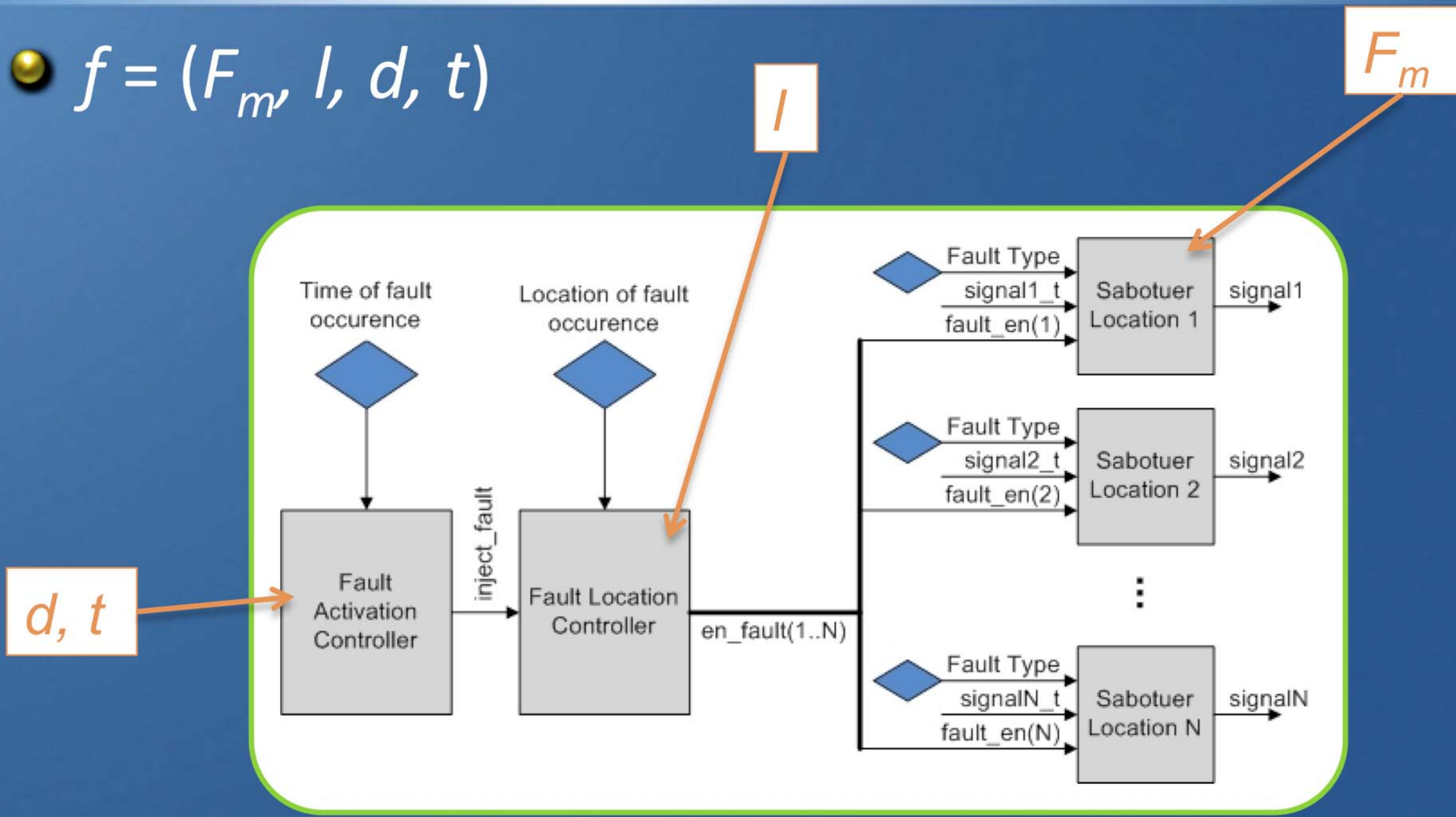
Design Instrumentation

- For combinational logic:
  - Saboteurs intercept design signals
    1. Identify fault location
    2. Split signal into two pieces
    3. The saboteur overrides the original signal value
  - Only requires minimal design changes
    - Easier to justify that design behavior has not changed



The Flexible Fault Framework

- $f = (F_m, l, d, t)$



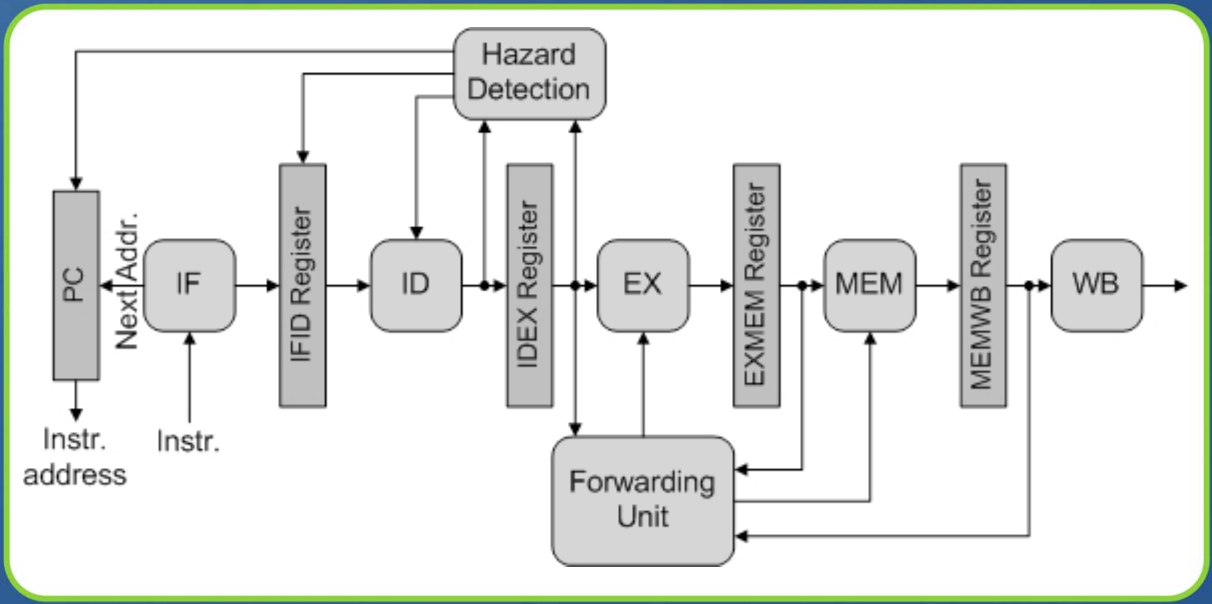
ABVFI uses non-determinism to model the random behavior of faults

Outline

- Overview of model checking and ABV
- The ABVFI methodology
- ABVFI implementation
  - Fault Injection Mechanism
    - Design instrumentation
  - The Flexible Fault Framework
- Case Study – The PHFT processor
- Conclusion

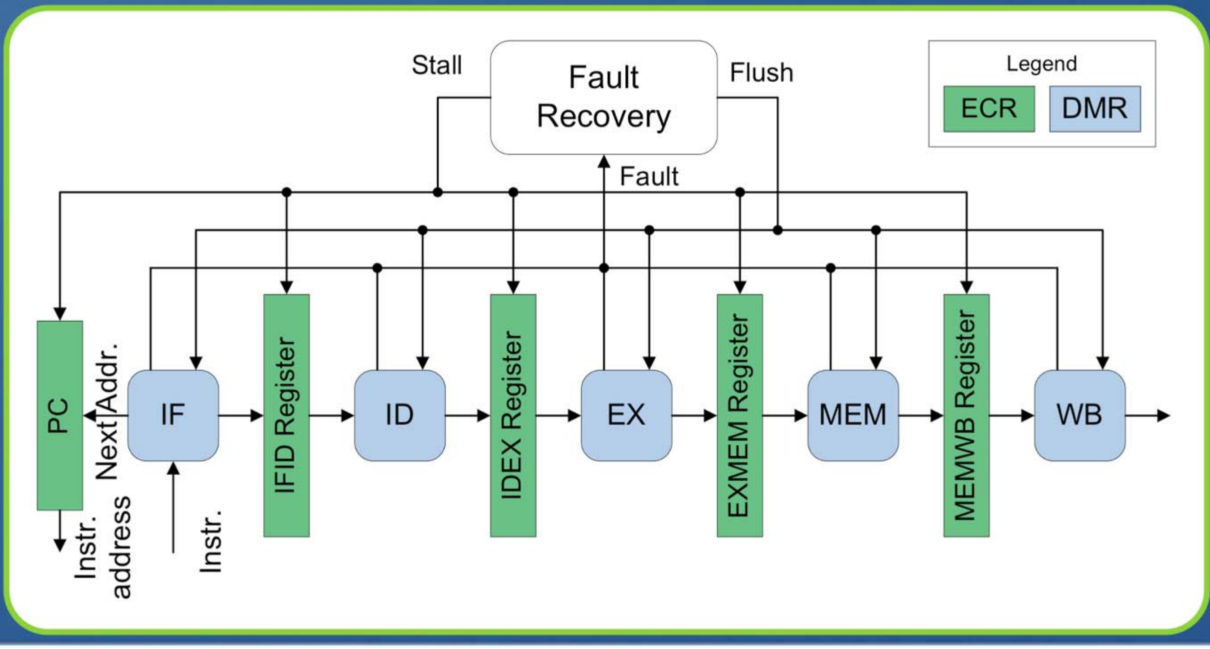
Case Study – PHFT Processor

- Requirements
  - Develop a 32-bit, 5 stage, pipelined processor
  - The processor should be capable of handling an SEU
- The design is based on the RISC-style processors developed by Patterson & Hennessy
  - It includes hazard detection and data forwarding
  - Study does not include register file or main memory
- Used IBM's Rulebase ABV toolset



The PHFT Processor

- Fault tolerance in the PHFT processor is:
  - Inter-stage registers – error correcting registers
    - Capable of detecting and correcting an SEU
  - Pipeline stages – DMR
    - Capable of detecting an SEU
  - Detected faults in the DMR stages stalls the pipe while the fault propagates out of the logic



Experimental Setup

- The modularity and fault tolerance mechanisms included provided a natural decomposition
  - Demonstrates how ABVFI can be used on large, complex designs
- Analysis took place in three stages
  ① Inter-stage registers
  ② DMR stages
  ③ Fault recovery

PHFT Results

- The ABVFI analysis showed that the PHFT processor provides total fault coverage

Processor Stage	Number of Assertions	Fault Locations (bits)	Total Analysis Time (minutes)
Fetch	3	192	1.97
Decode	8	48	2.92
Execute	7	230	3.12
Memory	2	64	7.23
Write back	1	64	0.4
PCDMR register (8 bit)	2	18	0.43
Fault recovery	6	30	4.50
Total	29	646	20.57

Case Study Summary

- The analysis results show
  - For all faults locations and under the SEU fault model, the PHFT processor covers all faults
  - ABVFI is feasible for real-world systems
    - A piecewise approach can be taken for large, complex designs
    - ~20 minutes total computation time for analysis
    - Took much less time than simulation would take for an exhaustive test (days, weeks, months?)

Other case studies

- Two other case studies revealed partial fault coverage
  - The distance kernel
    - demonstrated ABVFI in an assessment/verification role
  - The RGL design
    - demonstrated ABVFI is an enhanced design process for a safety-critical system

Outline

- Overview of model checking and ABV
- The ABVFI methodology
- ABVFI implementation
  - Fault Injection Mechanism
    - Design instrumentation
  - The Flexible Fault Framework
- Case Study – The PHFT processor
- Conclusion

Summary

- Manufacturing trends indicate that IC designs are going to become less reliable
- In order to deal with faults, designers need to address dependability during the development process
- Current practices in IC fault injection either rely on statistical methods or provide results that are incomplete
- ABVFI is a methodology that aims to address these issues
- Through ABVFI, IC designers can assess the fault coverage of a design using an exhaustive and accurate fault injection technique

Contributions of ABVFI

- Uses formal verification for an exhaustive analysis
- The Flexible Fault Framework provides the flexibility to model applicable faults
- An analysis considers faults in both sequential and combinational logic for a more accurate analysis of error propagation
- Includes a toolset that
  - Eases the adoption into existing practices
  - Provides objectivity
- Can be applied in multiple ways
  - Coverage-aware design, safety assessment, enhanced design

Questions?