INFRASTRUCTURE FOR AVF MODELING

MARK WILKENING

JUNE 14, 2015

OVERVIEW

AMD

- Motivation
- Architectural Vulnerability Factors
  - Program Vulnerability Factor
  - Hardware Vulnerability Factor
  - Spatial Multi-bit Architectural Vulnerability Factor
- Infrastructure for AVF Measurement
  - Infrastructure Overview
  - API Overview

MOTIVATION

MICROPROCESSOR RELIABILITY AND TRANSIENT FAULTS

AMD

- Reliability is a major constraint in the design of microprocessors
- Transient faults are a primary concern in designing reliable microprocessors
  - These faults occur when a particle strike deposits enough charge to flip state in storage elements
- Microprocessor vendors set a failure rate (FIT) target and perform significant analysis to efficiently design for and validate against this target
  - Pre-RTL
    - Architectural Vulnerability Analysis
  - Pre-Silicon
    - Statistical Fault Injection
  - Post-Silicon
    - Particle Beam Testing

AMD

Architectural Vulnerability Factors

ARCHITECTURAL VULNERABILITY FACTORS (AVF)
[MUKHERJEE03]

AMD

- Not all faults (bit flips) become errors (incorrect program output)
- The Architectural Vulnerability Factor (AVF) of a hardware structure is defined as the probability that a fault in the structure becomes an error
- AVF can be conservatively estimated through ACE Analysis
  - Determines during each cycle which bits are required for Architecturally Correct Execution (ACE)

$$AVF_H = \frac{\sum_{n=0}^{N} ACE\ bits\ in\ H\ at\ cycle\ n}{B_H \times N}$$

B_H: Size in bits
N: Number of cycles

  - Multiple variants of AVF for multiple structures can be estimated during a single fault-free simulation run
  - Well suited to early stage (pre-RTL) design exploration
  - Much faster than software fault injection

ARCHITECTURAL VULNERABILITY FACTORS CONT.

AMD

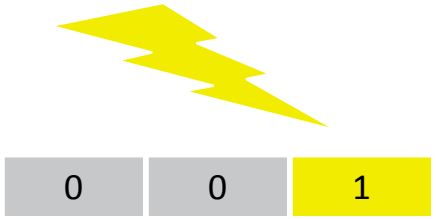
HARDWARE VULNERABILITY FACTORS AND PROGRAM VULNERABILITY FACTORS [SRIDHARAN09, SRIDHARAN10]

- Faults which do not propagate into errors are called masked
- Faults can become masked at many different levels in a system
  - Device level masking
    - Circuit level timing
  - Microarchitecture level masking (Hardware Vulnerability Factor)
    - Register lifetimes
    - Performance enhancing state (branch prediction tables)
  - Architecture level masking (Program Vulnerability Factor)
    - Dynamically dead state
    - Logically masked state
  - Application level masking
    - Approximately correct state
- AVF considers both microarchitectural and architectural masking
  - AVF = HVF x PVF

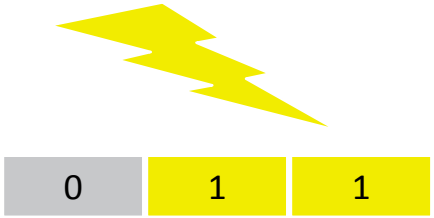
ARCHITECTURAL VULNERABILITY FACTORS CONT.

SPATIAL MULTI-BIT ARCHITECTURAL VULNERABILITY FACTORS [WILKENING14]

- As process technology scales downward it becomes more likely a single particle strike will simultaneously flip multiple bits
  - This is known as a spatial multi-bit transient fault
- The effects of spatial multi-bit faults are non-trivial, and spatial multi-bit AVF (MBAVF) can vary significantly from single-bit AVF
- MBAVF can give insight into reliability behavior involving
  - Various multi-bit patterns (fault modes)
  - Different interleaving schemes and error correcting codes
  - Both microarchitectural and architectural masking



Single-bit Fault

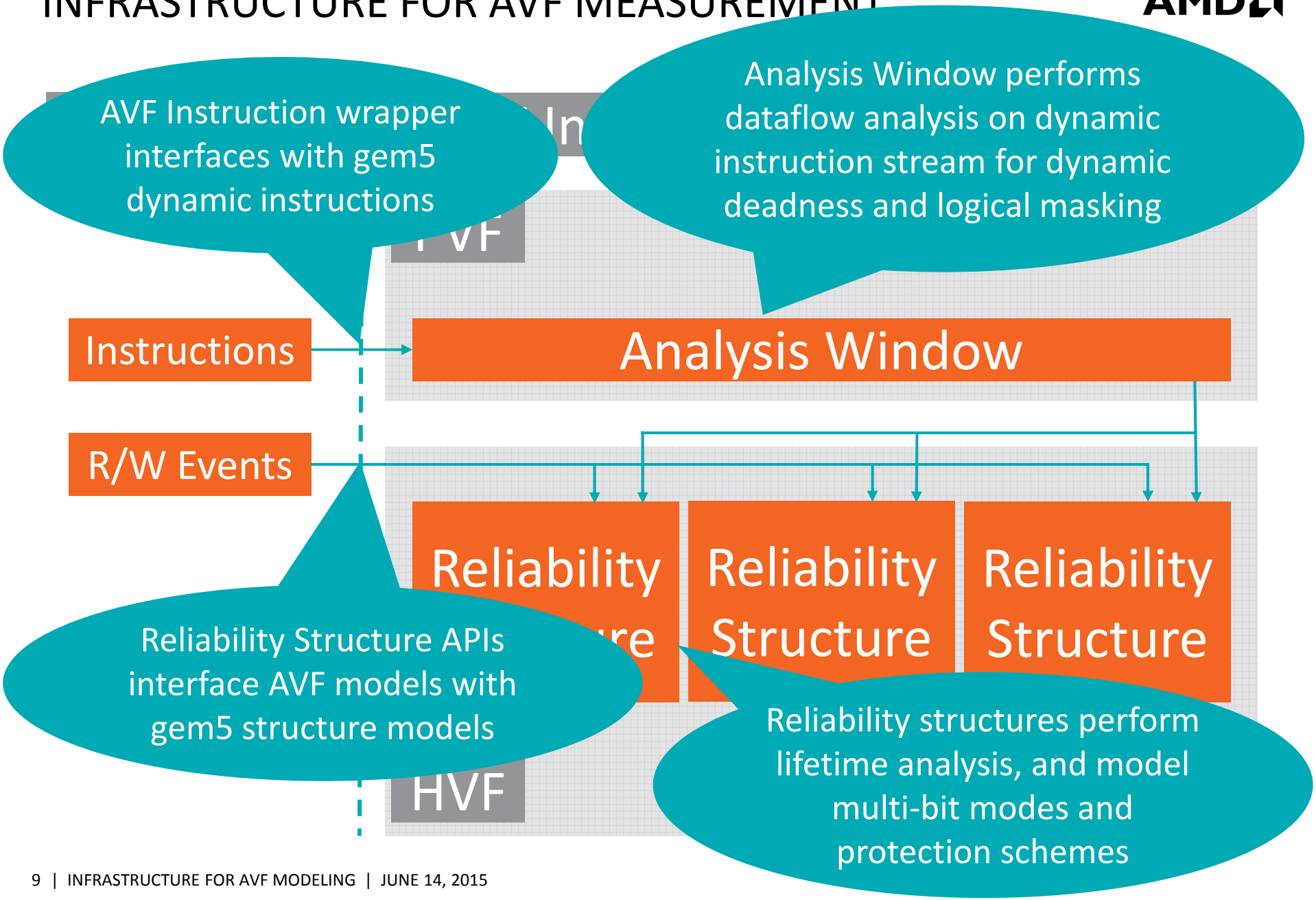


Multi-bit Fault

AMD

Infrastructure for AVF Measurement

INFRASTRUCTURE FOR AVF MEASUREMENT



INFRASTRUCTURE FOR AVF MEASUREMENT

API OVERVIEW

- Register committed instructions
  - PostCommitBuffer->insert(

```
instruction,                                  // dynamic instruction
curTick())                                    // commit cycle
```

- Register read and write events to modeled state
  - Pass cycle, identifying information for associated instruction, location in modeled structure, and associated architectural state
  - CPURegisterFile->read_with_association(

```
curTick(),                                    // current cycle
device_id, context_id, thread_id, dynamic_id, // instruction identifiers
2, true, 3,                                   // info needed for variable length arguments
reg_num, byte,                                // register #, byte – physical register location
thread_id, reg_num, byte)                     // thread, register #, byte – architectural state
```

AMD

Questions?

DISCLAIMER & ATTRIBUTION

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

ATTRIBUTION