

Error Injection in a Subsystem Level Constrained Random UVM Testbench

Jeremy Ridgeway and Hoe Nguyen
Broadcom, Inc.

October 23, 2018
Austin



Agenda

Introduction

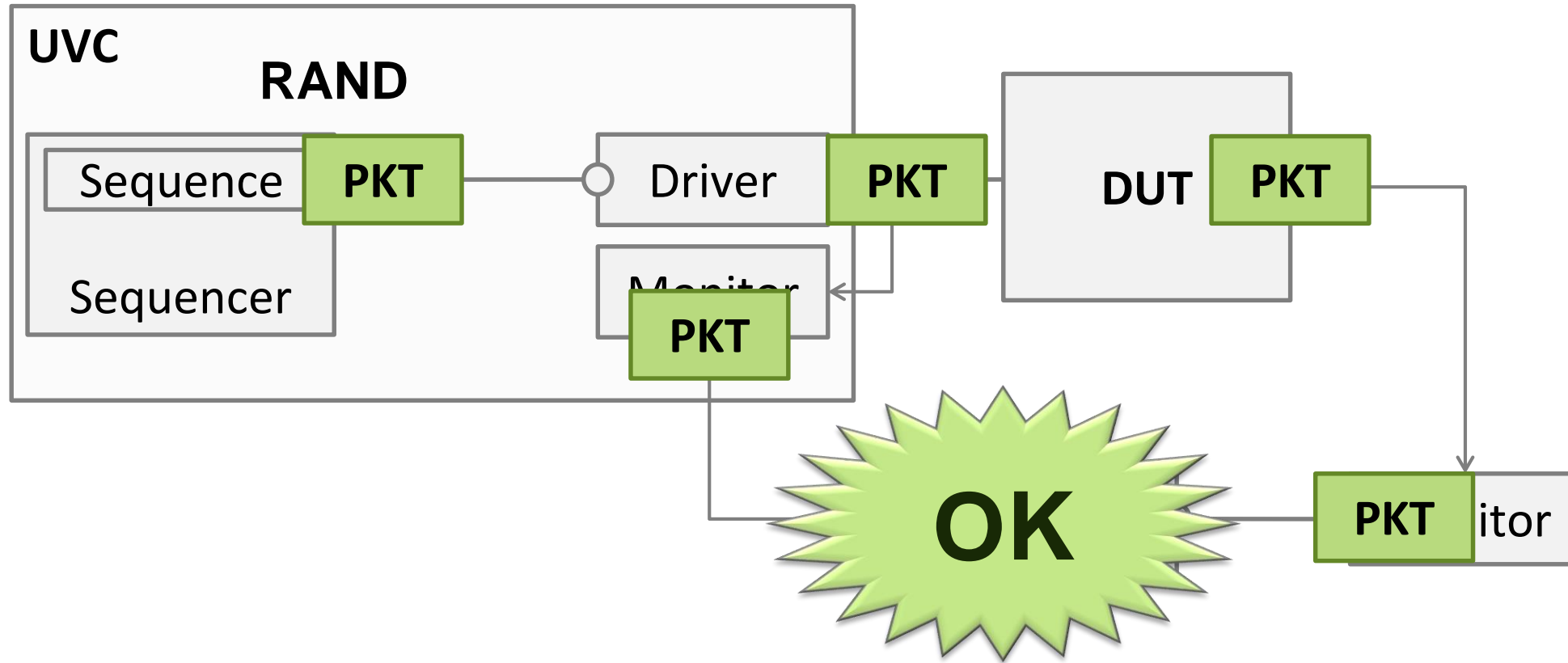
Selective Error Report Demotion

Managing Error Injection

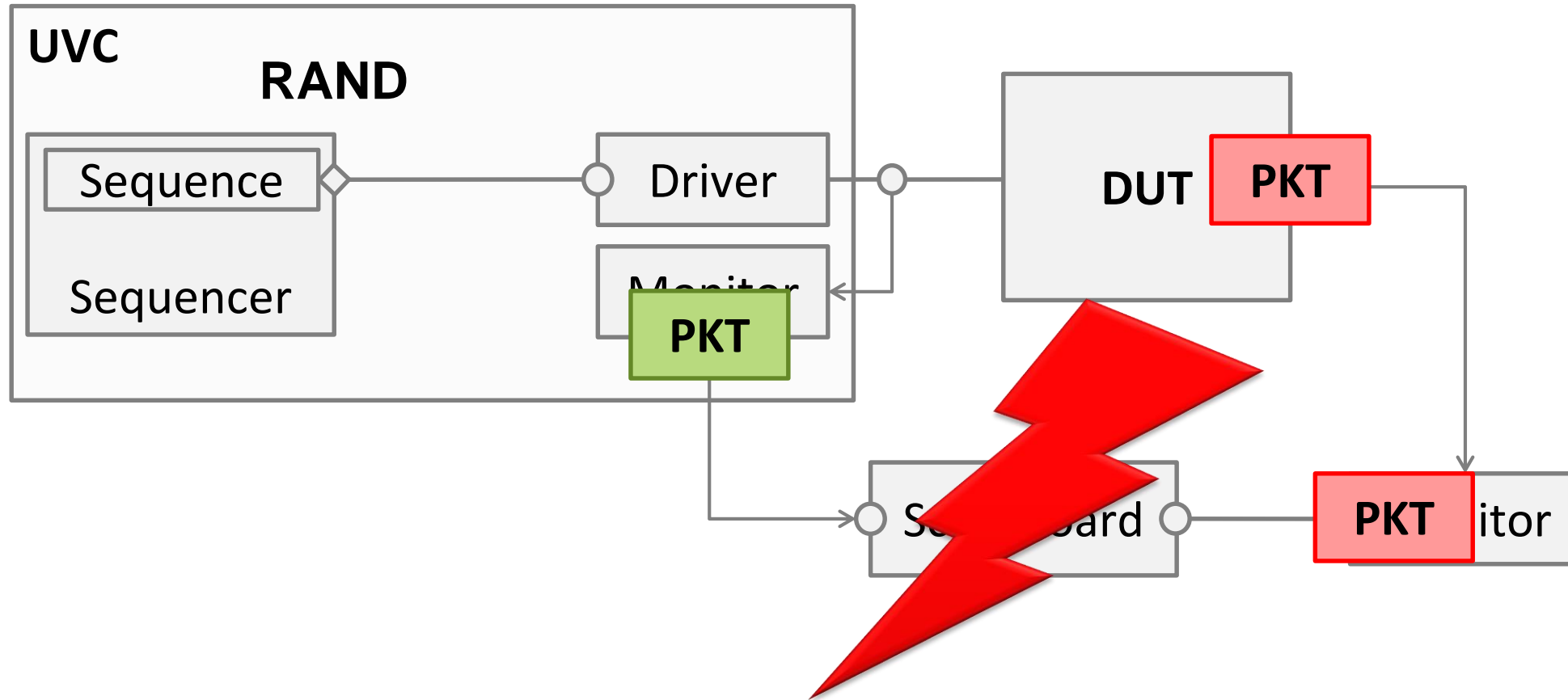
Expecting Errors

Responding to Interrupts

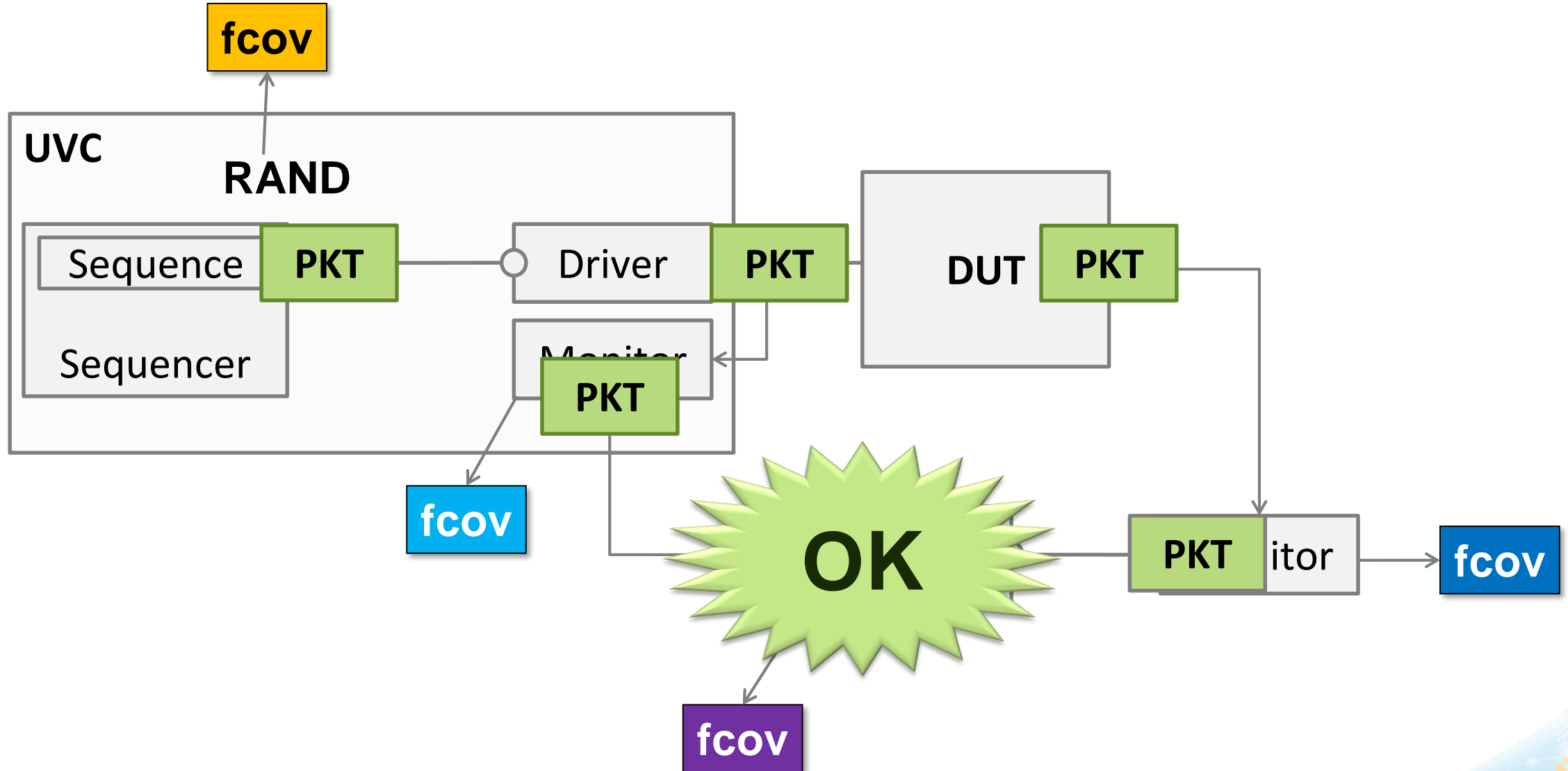
Constrained Random UVM Testbench: Good Data



Constrained Random UVM Testbench: Error Detected



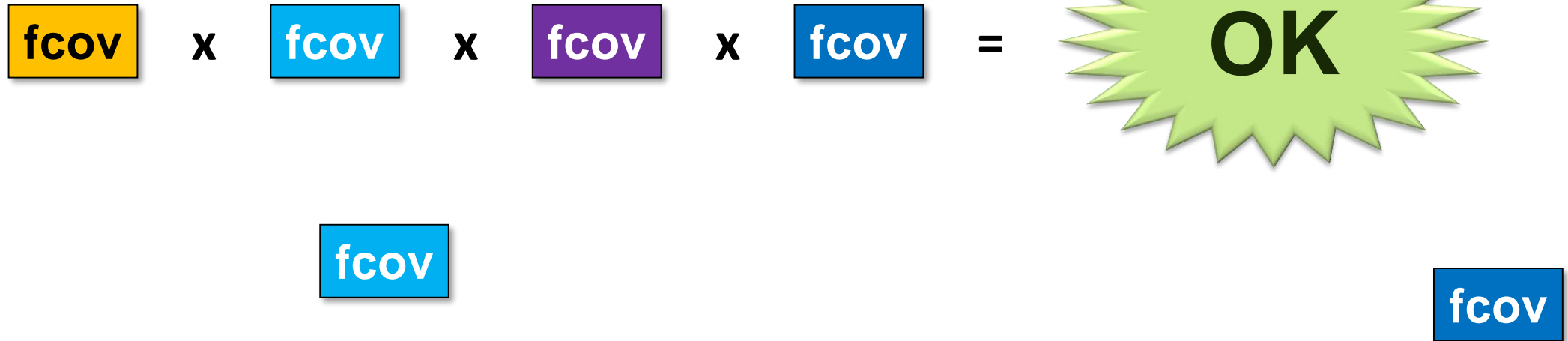
UVM Testbench Reporting Layer: Functional Coverage of Good Data



Random Scenario

Cross Coverage of Good Data

fcov



- Important for verification status

fcov

Random Scenario

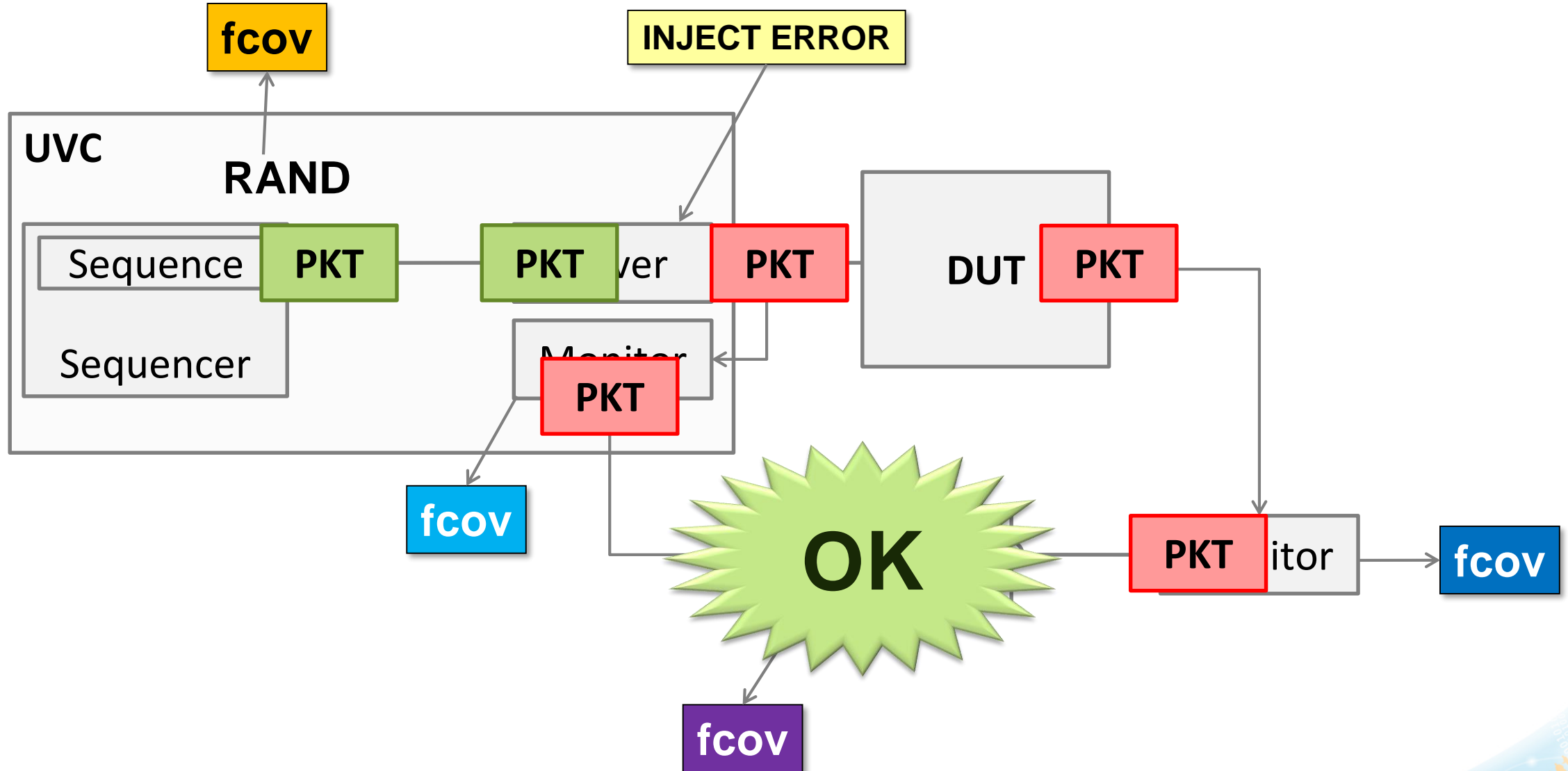
Cross Coverage: Error Detected

$$\boxed{\text{fcov}} \times \boxed{\text{fcov}} \times \boxed{\text{fcov}} \times \boxed{\text{fcov}} =$$

FAIL

- Equally important

Random Scenario Testing WITH Error Injection



Agenda

Introduction

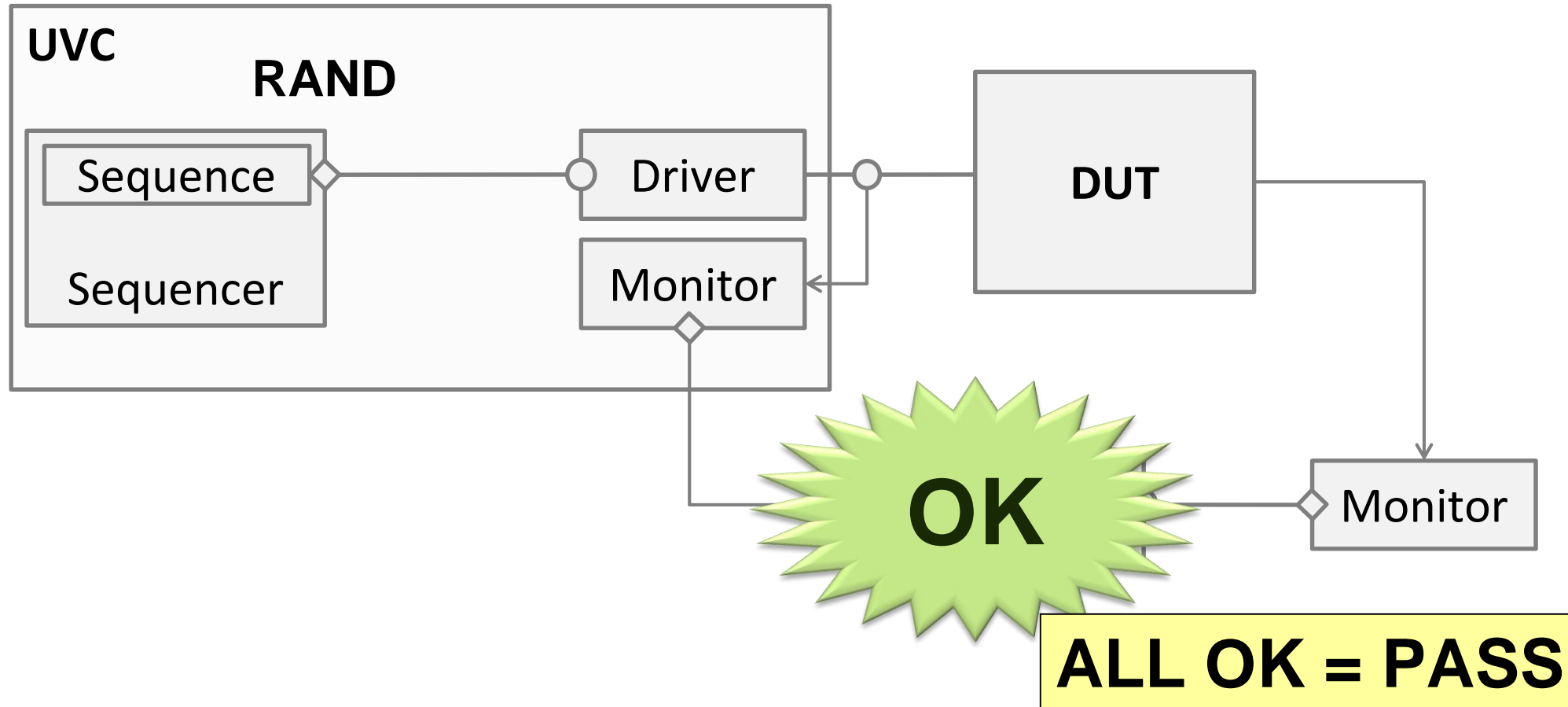
Selective Error Report Demotion

Managing Error Injection

Expecting Errors

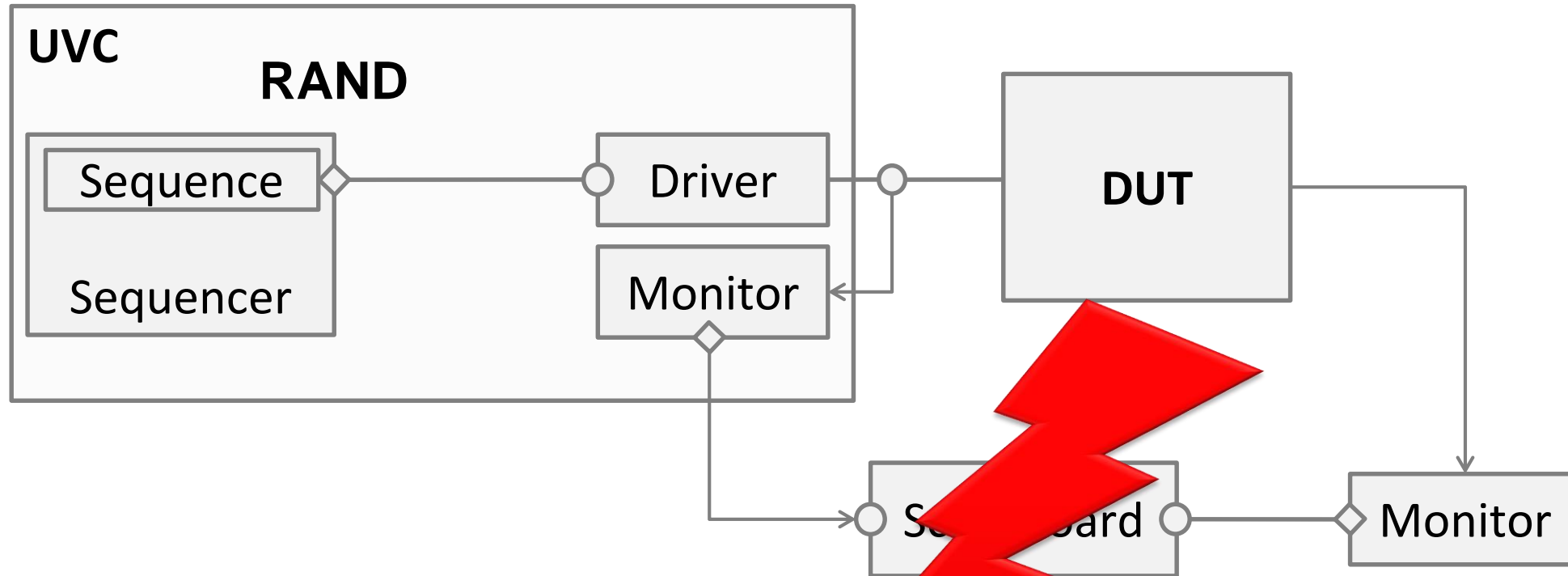
Responding to Interrupts

Reporting Layers: Test PASS/FAIL



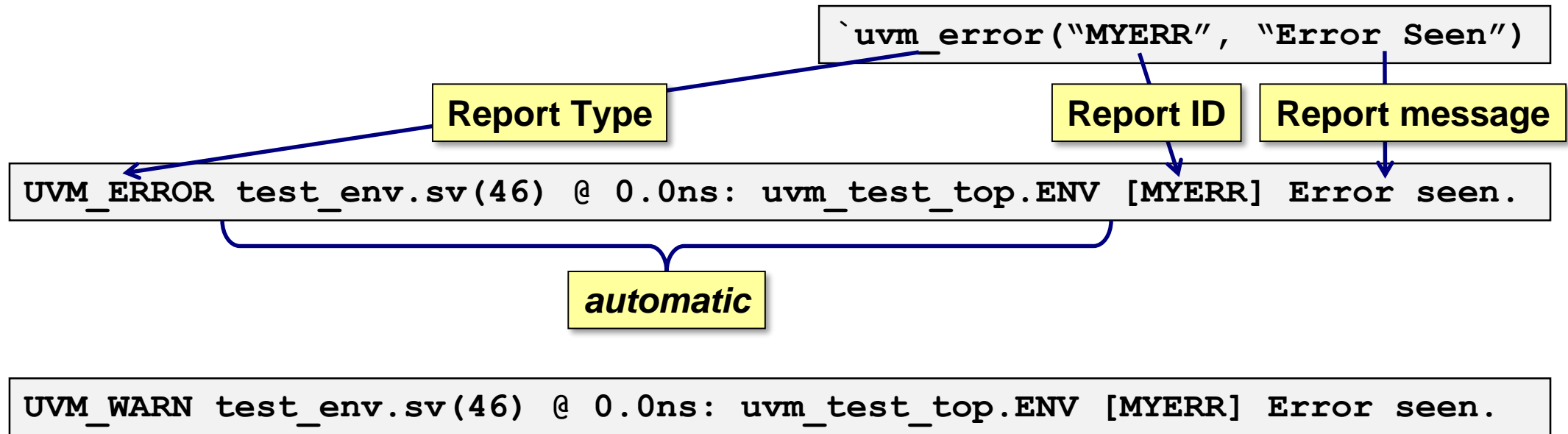
Reporting Layers: Test PASS/FAIL

Need to Ensure Error Reports are accurate



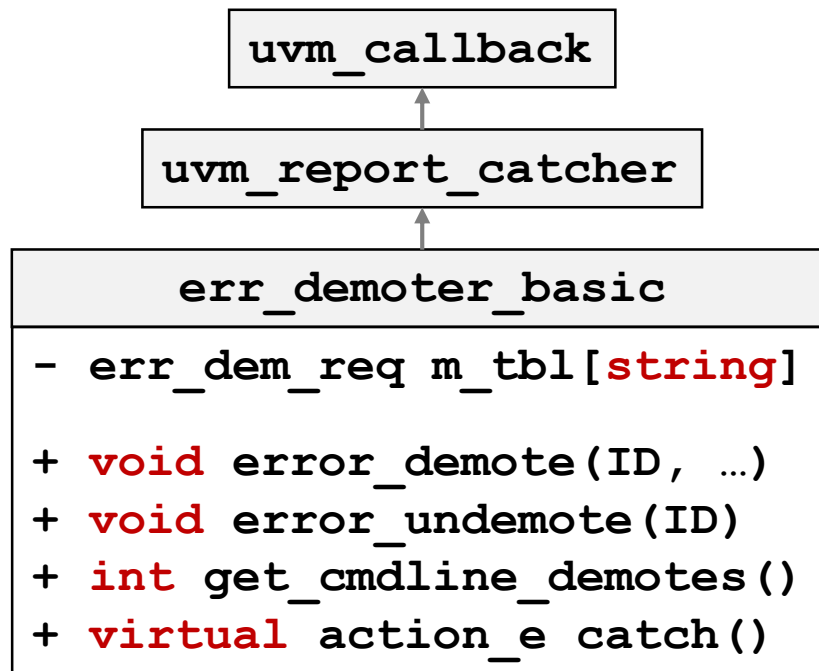
JUST 1 ERROR = FAIL

Error Report Message Demotion



- Focus on the Report Type and Report ID
- Catch all report messages
- Demote known Report Error Report ID messages to WARNING

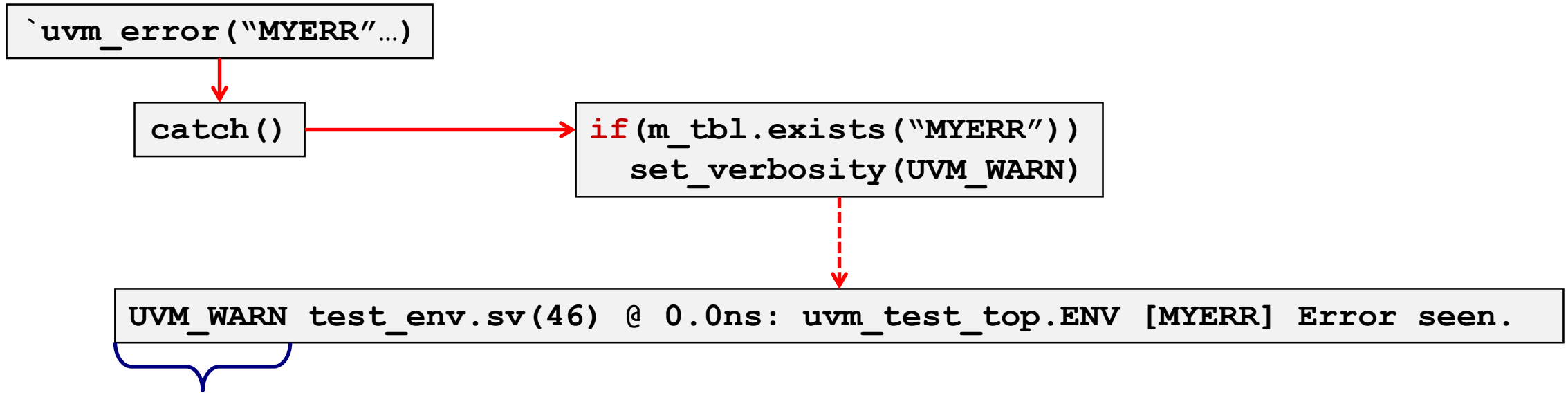
UVM Report Catcher Extension



```
function new (...);  
    super.new (...);  
    uvm_report_cb::add(.obj(null),  
        .cb(this), ordering(UVM_PREPEND));  
endfunction
```

- Error Demoter maintains an array keyed by Report Message ID
- Demote when caught report message matches and:
 - Always, Within (absolute/relative) time window, Count

UVM Report Catcher Flow



- Error demotion becomes warning when expected
- Error demotion becomes info when managed by error injection services

Agenda

Introduction

Selective Error Report Demotion

Managing Error Injection

Expecting Errors

Responding to Interrupts

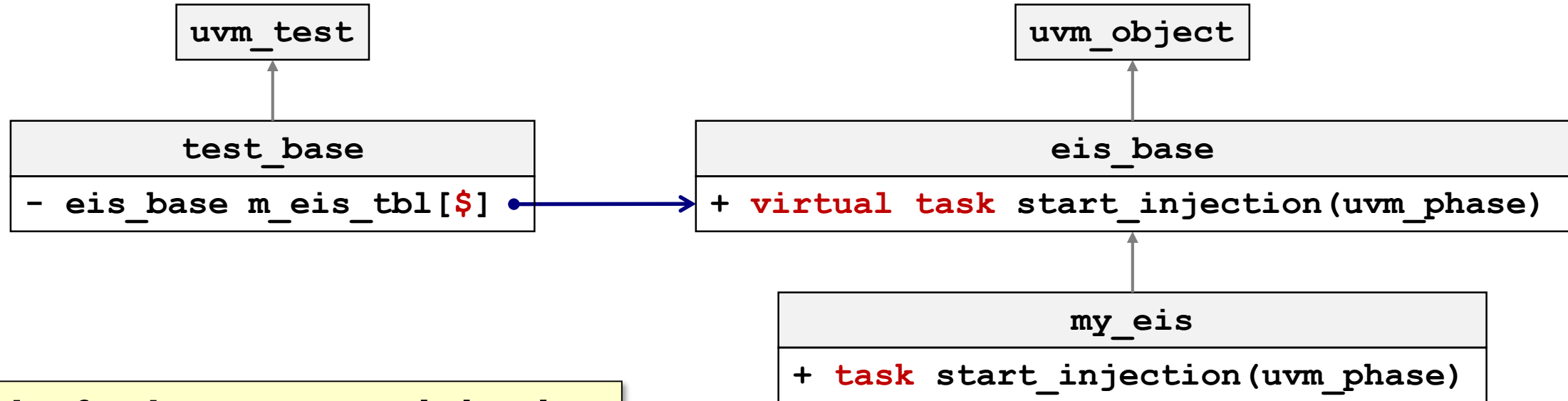
Error Injection Services (EIS)

- Single object to manage everything*
 - Inject the error
 - Expect the error
 - Recover from the error – reset?
- May be instantiated any time == uvm_object
- Randomly selected and started

***As with most functional verification, there are no absolutes.**

Error Injection Services (EIS) Object

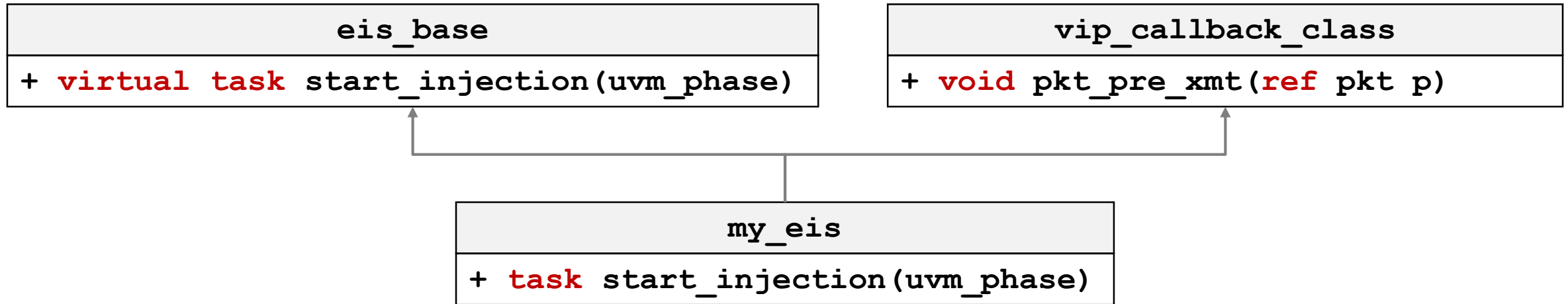
- Contain common functionality in base class
- Extend for specific error injection type
- Randomly select and start error injections



Works for homegrown injection

VIP Callbacks

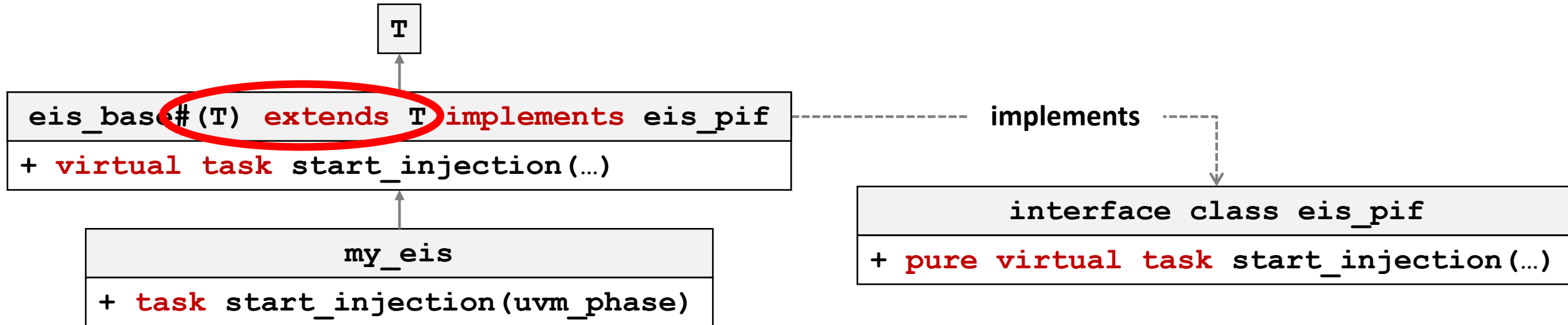
Inheritance is ... difficult



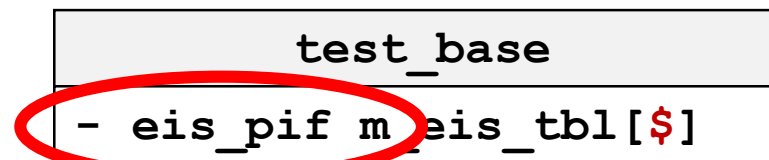
- C++-style multiple inheritance is not supported in SystemVerilog

interface classes **opaque type-parameters**

Opaque Type Parameter Extension with an Interface Class

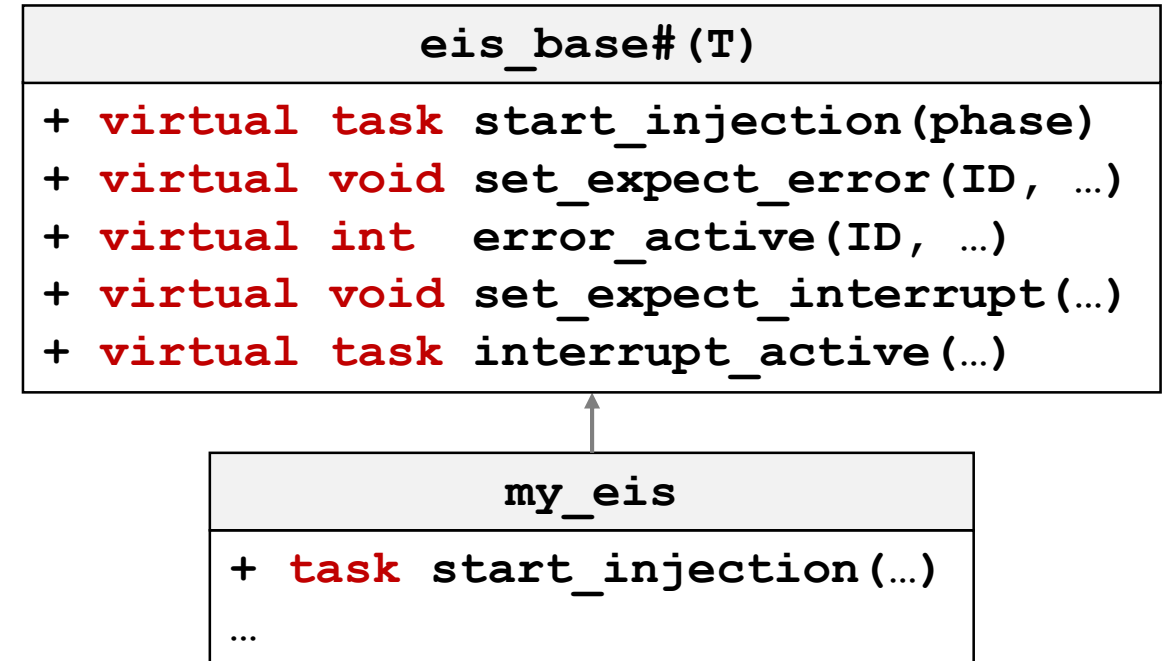


- Opaque type param extension won't solve the problem of maintaining a table of EIS objects:



Managing Error Injection: Common API

- Start injection – task taking uvm_phase allows for blocking if necessary
- Instrumentation
 - Set expected errors
 - Set expected interrupts
- Response via callback access
 - Error report ID is active
 - Contextual Interrupt is active
- Infrastructure to handle connections



Agenda

Introduction

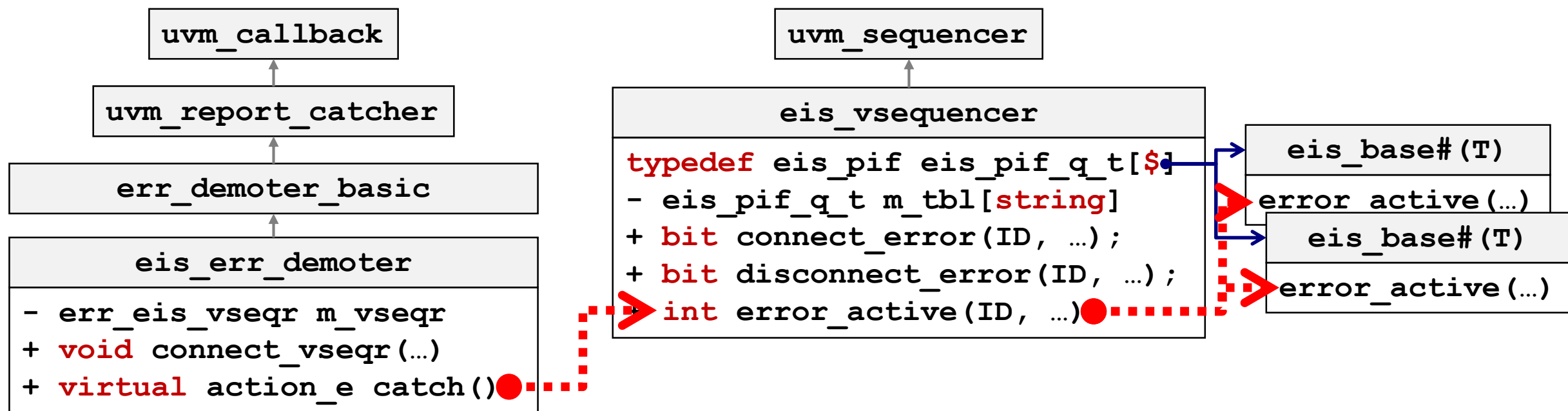
Selective Error Report Demotion

Managing Error Injection

Expecting Errors

Responding to Interrupts

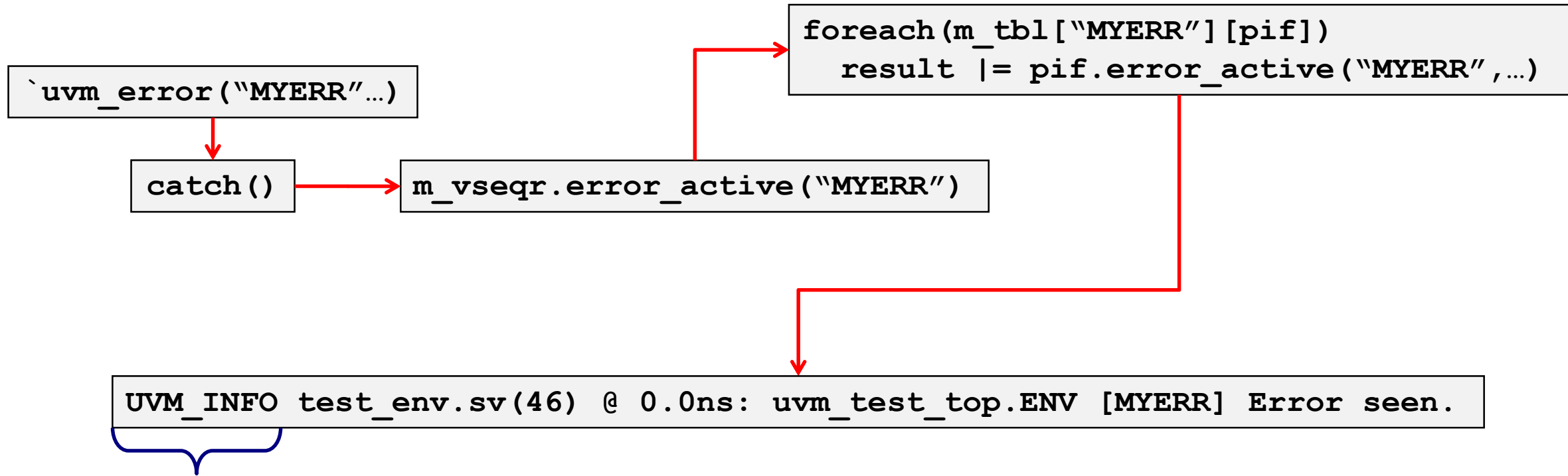
Errors Expected during Error Injection



- Virtual sequencer:
 - Acts as intermediary between active EIS objects and demoter
 - Allows EIS object to start a virtual sequence

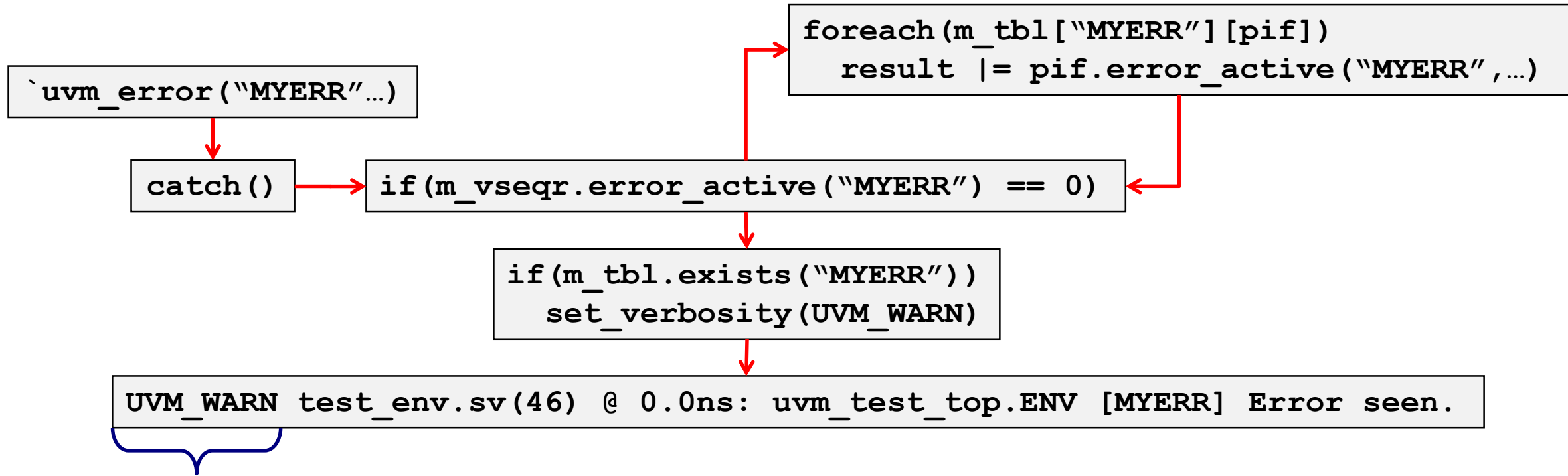
Expecting 1 or 2 EIS objects
to be active at one time

Expected Error Caught



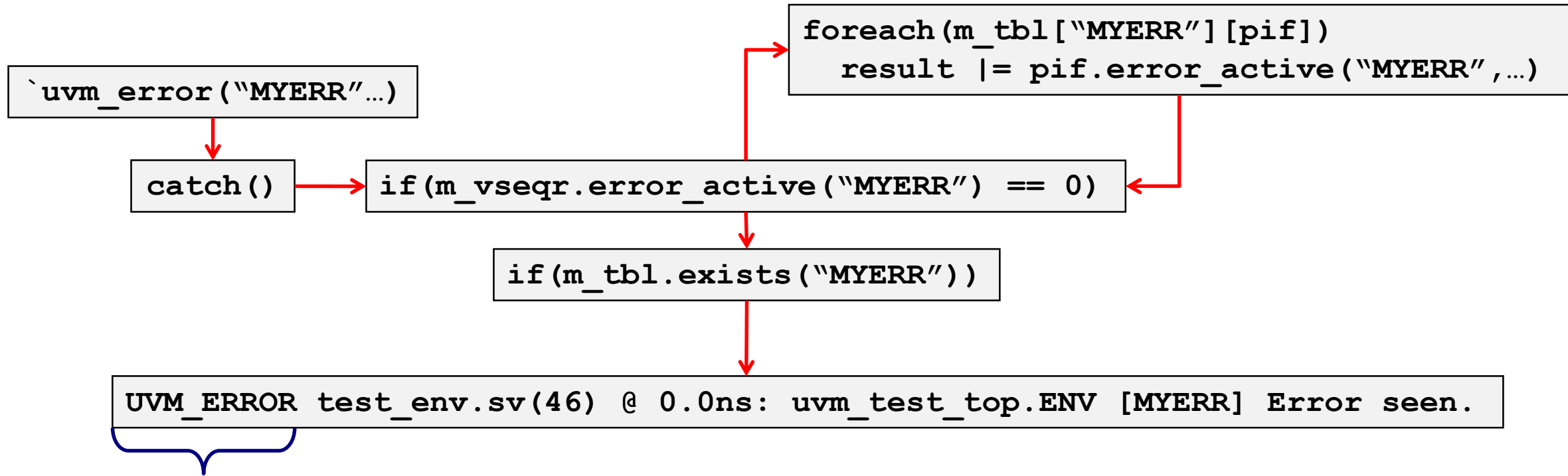
- Expected errors are not warnings – they are demoted to INFO

Demoted Error Caught



- Unexpected but demoted errors are still warnings

Unexpected Error Reported



- Unexpected errors are still errors

Agenda

Introduction

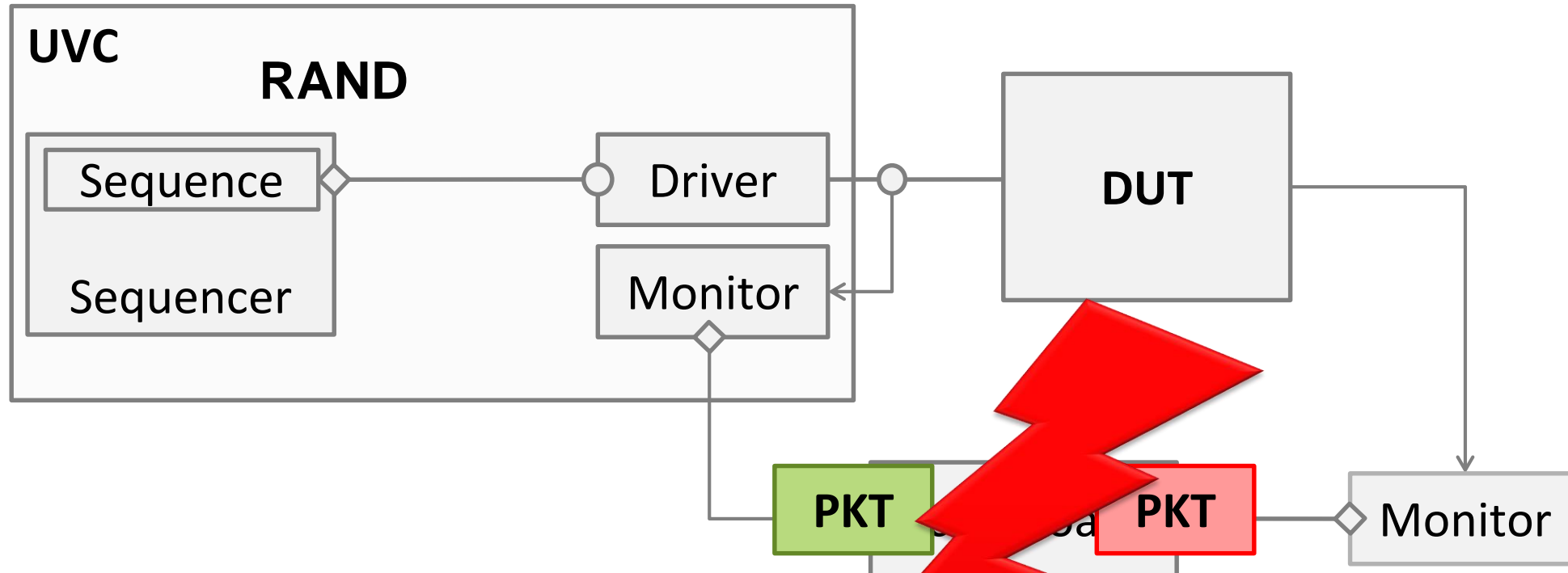
Selective Error Report Demotion

Managing Error Injection

Expecting Errors

Responding to Interrupts

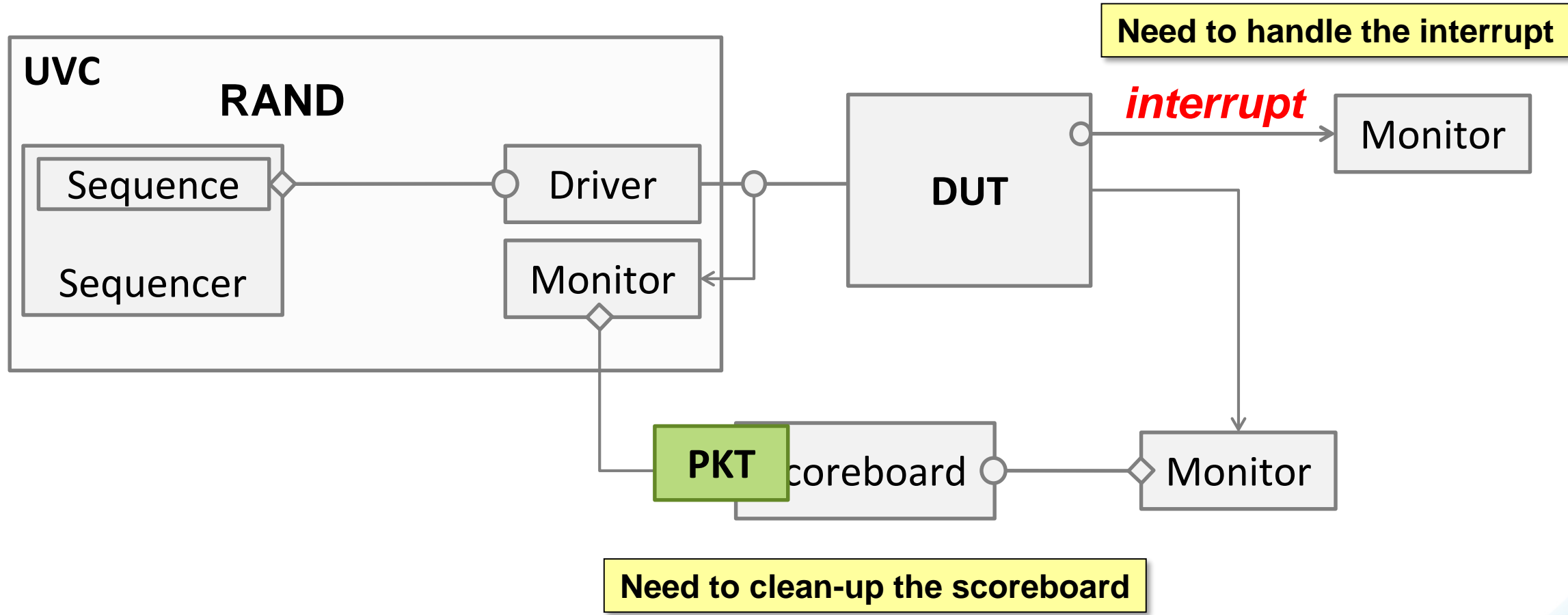
Error Injected at Driver



FAIL

Probably not going fail in this manner

Error Injection Cleanup

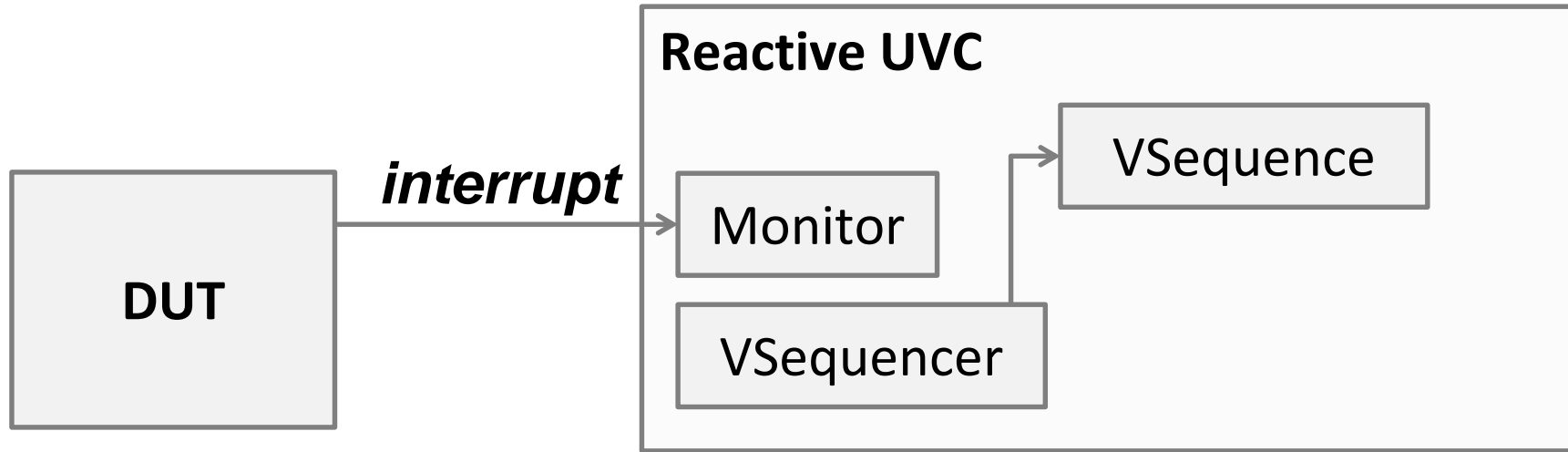


Example Register Description

Register	Field	[msb:lsb]	Default	Access
top_int	rsvd	[31:2]	0	RO
	RXPATH	[1]	0	W1C ←
	RXPKT	[0]	0	RO ←
pkterr	rsvd	[31:1]	0	RO
	CRC	[0]	0	W1C ←

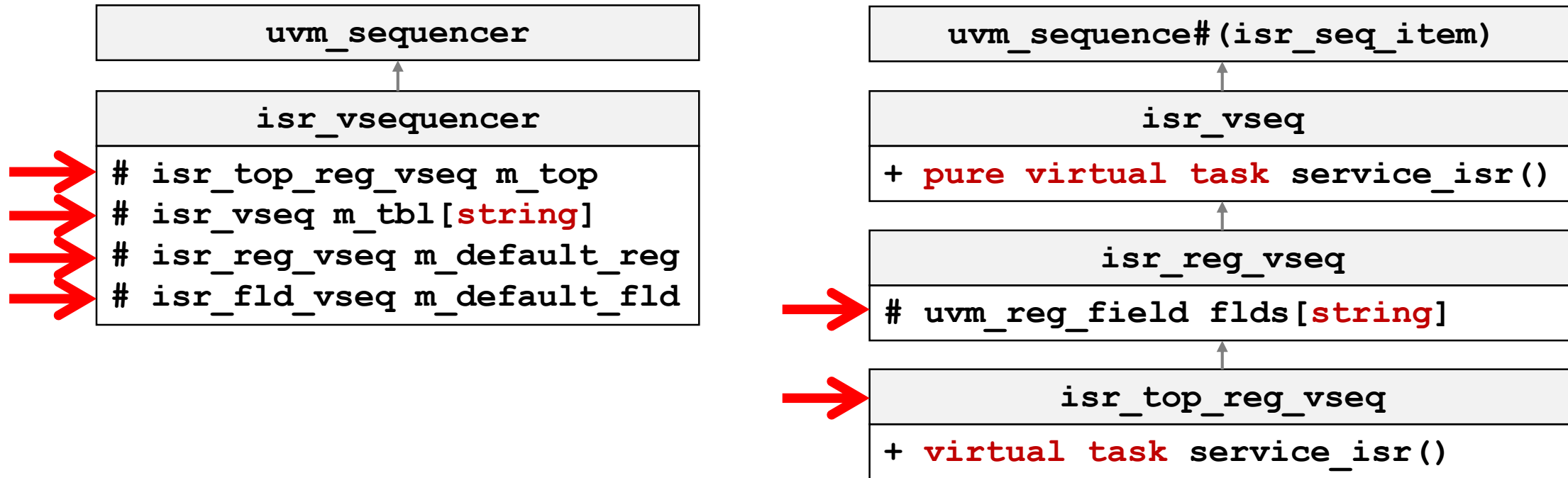
- RXPATH interrupt is autonomous and handled directly
- RXPKT interrupt is dependent on pkterr.CRC register field

Interrupt Service Routine




- At interrupt:
 - Monitor starts virtual sequence on virtual sequencer

ISR Vseqr and Register Vseq

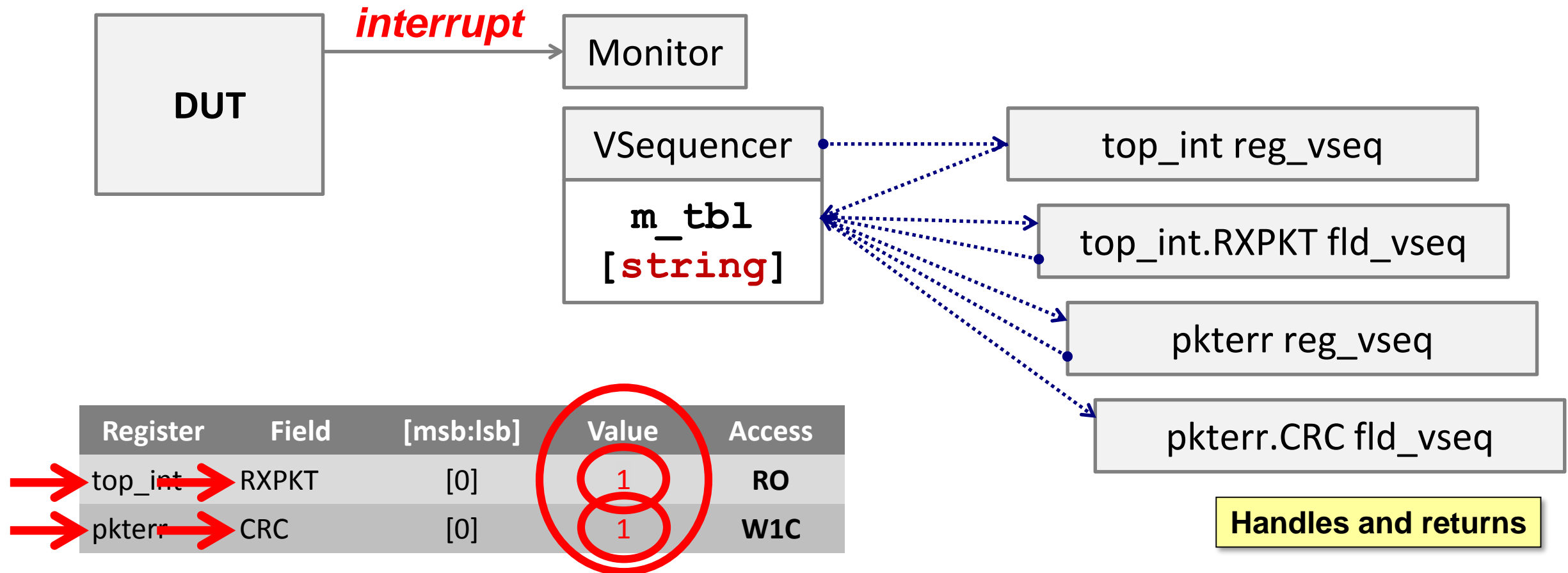


- Single top-level vseq exists *per* interrupt
- isr_reg_vseq table contains this register's field names
- vseqr table is full string register path

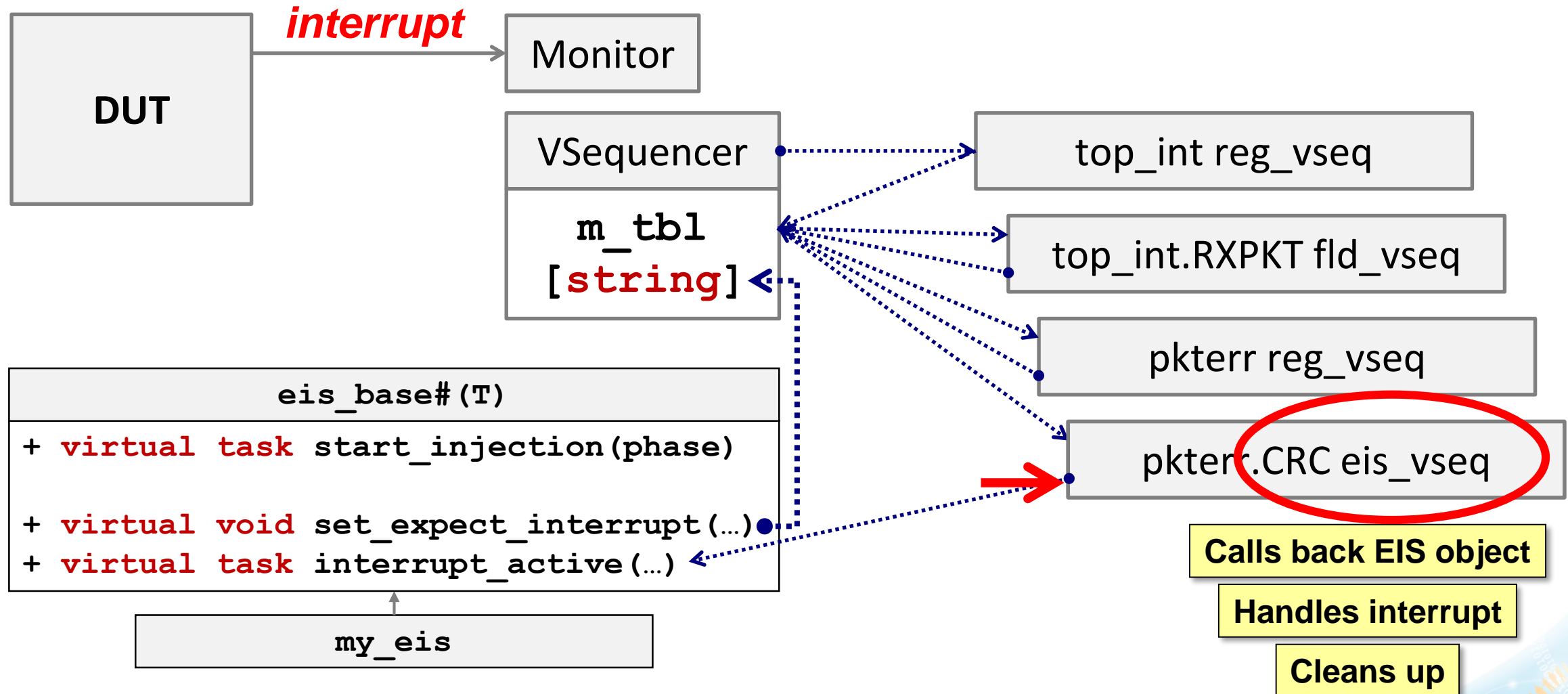


Register	Field	[msb:lsb]	Default	Access
top_int	rsvd	[31:2]	0	RO
	RXPATH	[1]	0	W1C
	RXPKT	[0]	0	RO
pkterr	rsvd	[31:1]	0	RO
	CRC	[0]	0	W1C

Example ISR: RXPKT=1/pkterr.CRC=1



Example ISR: RXPKT=1/pkterr.CRC=1



Summary

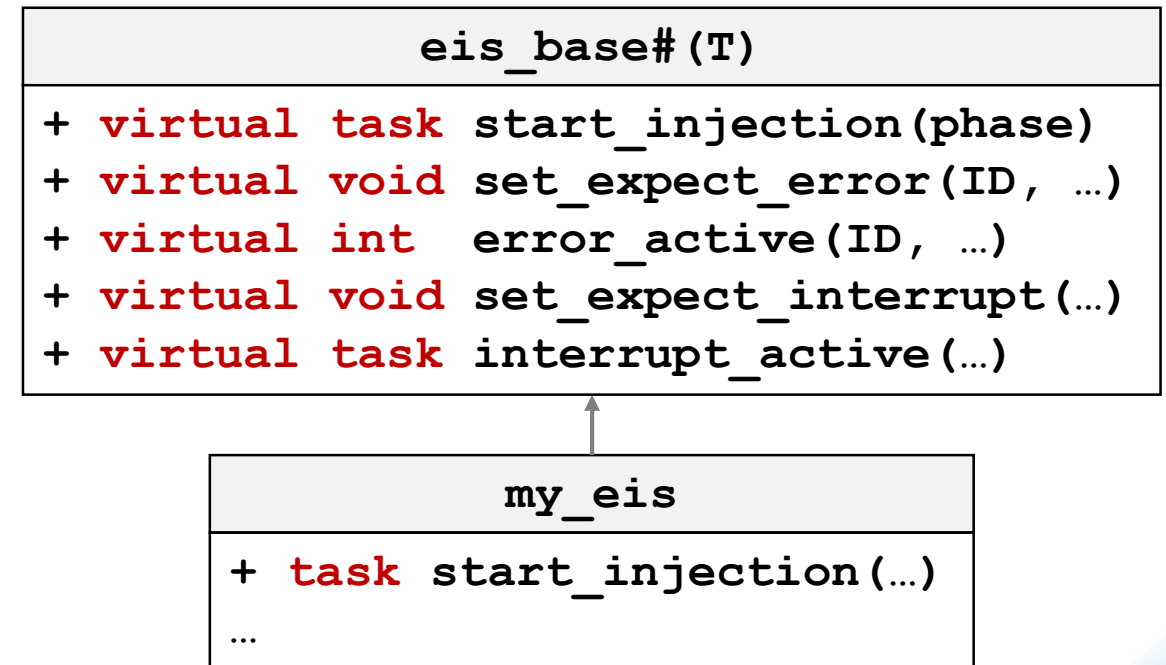
Random Error Injection Selection

Manage Error Injection

Manage Expected Errors

Manage and Respond to Interrupts

See Paper for [more Information](#)

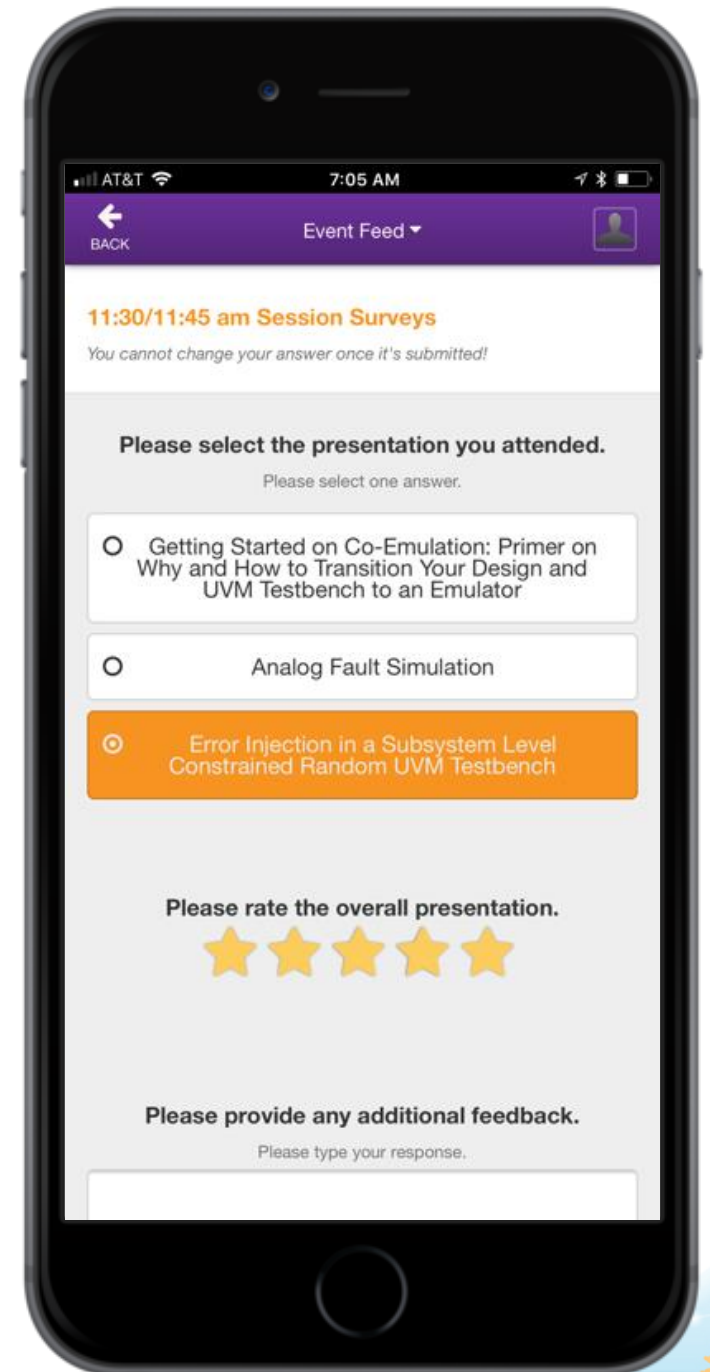


Thank You



Complete The Survey For This Presentation

- From the App, access from the “All Surveys” widget on the home screen or listed on the left navigation
- Select **11:30/11:45 am Session Surveys**
- Select this presentation title and rate using a 5-star scale
- Enter any additional feedback and **hit submit**



Extra Slides

Constrained Random UVM Testbench: Error Detected

