2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Functional Safety Verification For ISO 26262

Kevin Rich (krich@nvidia.com), NVIDIA
Shekhar Mahatme (smahatme@synopsys.com),
Meirav Nitzan (meirav@synopsys.com), Synopsys

NVIDIA. SYNOPSYS® Silicon to Software™

2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Agenda

- Emergence of the Self Driving Car
- Functional Safety Primer for Automotive Semiconductors
- Mutation Analysis for Validating the Verification Process
- Functional Safety Verification Flow: FMEA to FMEDA
- Customer Case Study; Q&A

Emergence of the Self Driving Car

2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

U.S. DOT Releases New Automated Driving Systems Guidance

September 12, 2017 | Ann Arbor, Michigan

TRANSPORTATION SECRETARY ELAINE L. CHAO ANNOUNCES VISION FOR AUTOMATED VEHICLE TECHNOLOGY, EMPHASIZES SAFETY BENEFITS AND CONSUMER EDUCATION FOCUS

Ann Arbor, MI - The U.S. Department of Transportation and the National Highway Traffic Safety Administration (NHTSA) today released new federal guidance for *Automated Driving Systems (ADS): A Vision for Safety 2.0*. This is the latest guidance for automated driving systems to industry and States.

"The new Guidance supports further development of this important new technology, which has the potential to change the way we travel and how we deliver goods and services," said U.S. Transportation Secretary Elaine L. Chao. "The safe deployment of automated vehicle technologies means we can look forward to a future with fewer traffic fatalities and increased mobility for all Americans."

A Vision for Safety 2.0 calls for industry, state and local governments,

Levels of Automation in Cars

AUTOMATION LEVELS OF AUTONOMOUS CARS

LEVEL 0

There are no autonomous features.

LEVEL 1

These cars can handle one task at a time, like automatic braking.

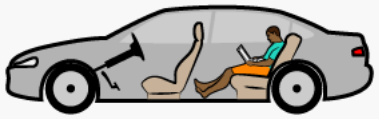
LEVEL 2

These cars would have at least two automated functions.

LEVEL 3

These cars handle "dynamic driving tasks" but might still need intervention.

LEVEL 4



These cars are officially driverless in certain environments.

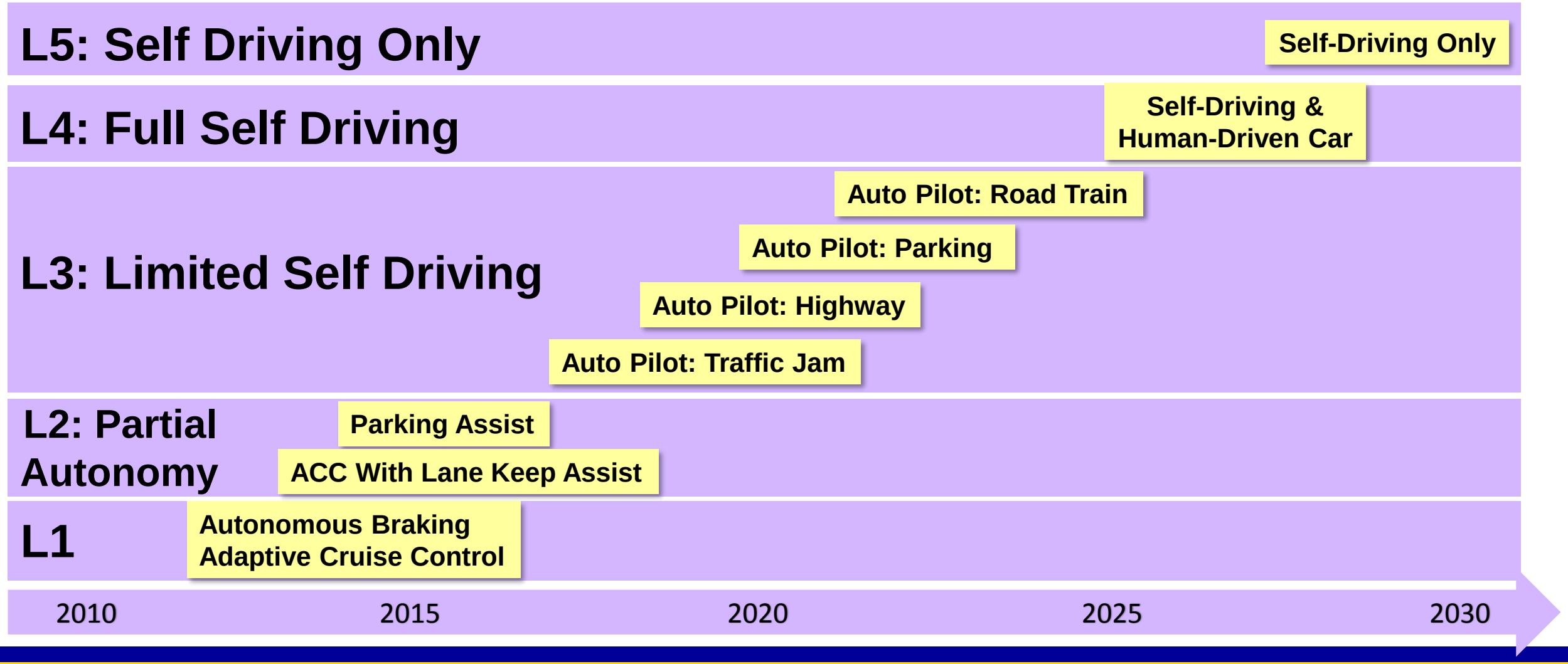
LEVEL 5

These cars can operate entirely on their own without any driver presence.

SOURCE: SAE International

BUSINESS INSIDER

Roadmap of Autonomous Cars



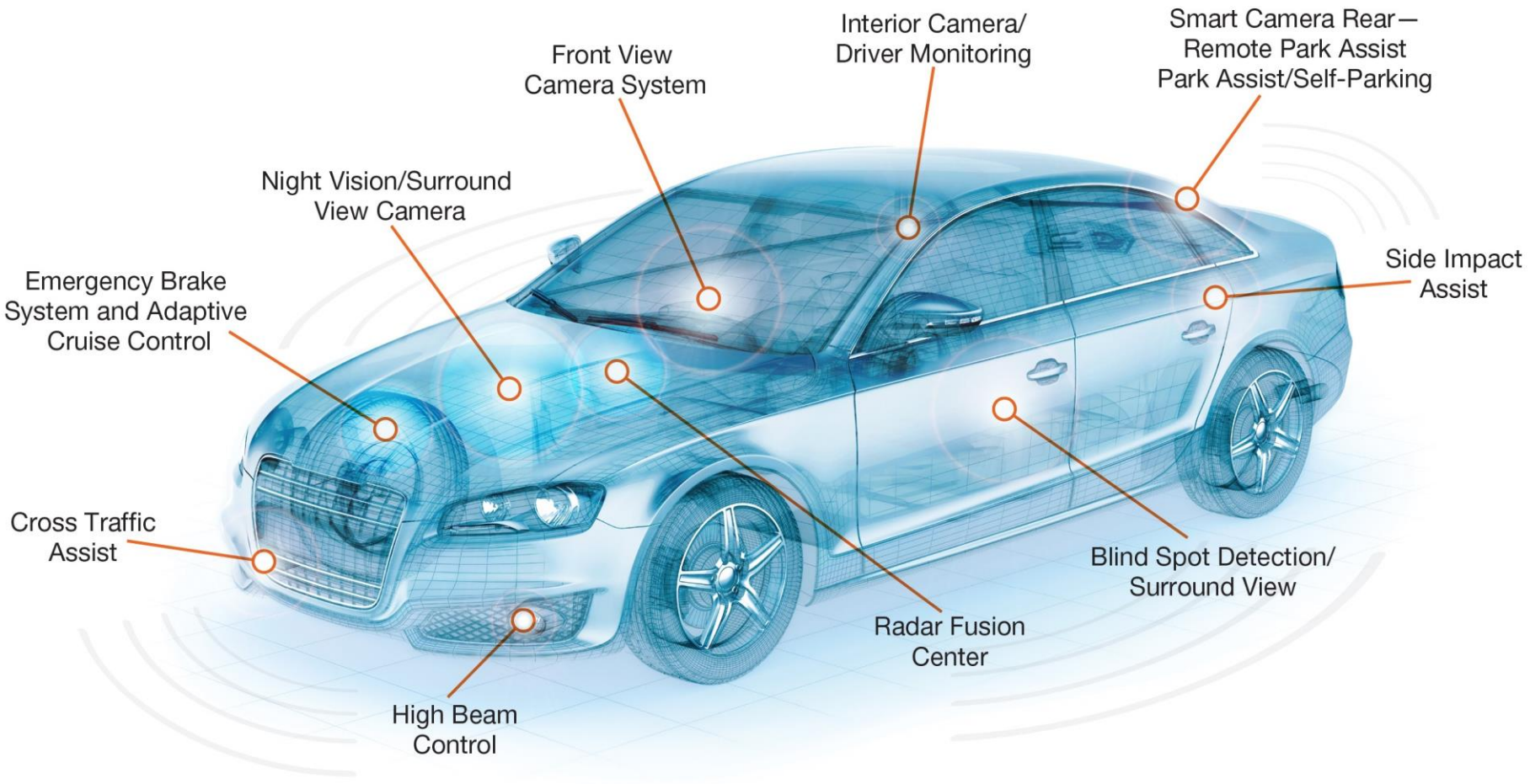
2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

ADAS and Self Driving Car Market Watch

2016	2017	2018	2019	2020	2021	2025	2030
WAYMO	TESLA	GM		HONDA, TOYOTA, RENAULT NISSAN, HYUNDAI, MERCEDES, AUDI	FORD, VOLVO, FCA FIAT CHRYSLER AUTOMOBILES, BMW	RENAULT NISSAN	HYUNDAI, UBER

Source:
https://venturebeat.com/2017/06/04/self-driving-car-timeline-for-11-top-automakers/
http://www.driverless-future.com/?page_id=384
http://mashable.com/2016/08/26/autonomous-car-timeline-and-tech/#C3BDRFPjcEq1

Complex SOCs For ADAS



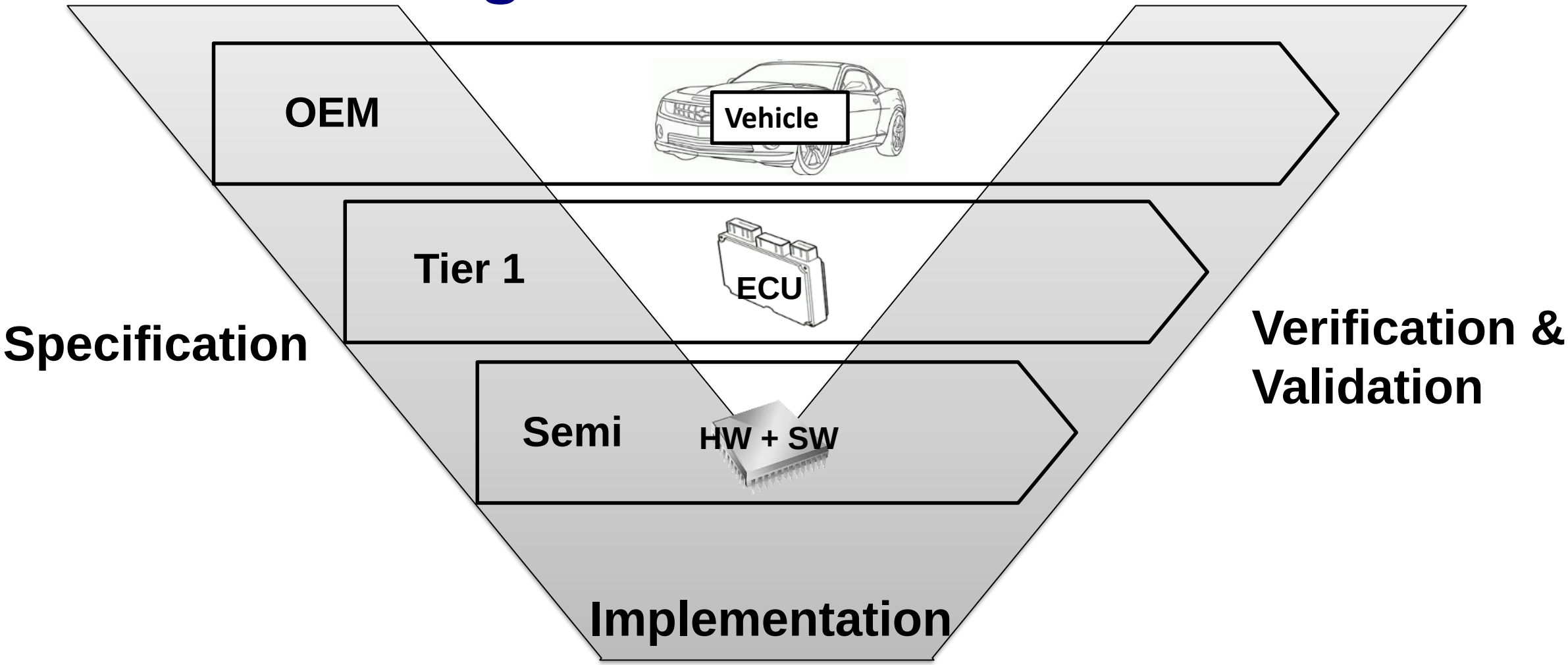
2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Functional Safety Primer for Automotive Semiconductors

What is Functional Safety?

- Functional Safety is the "Absence of unreasonable risk due to *hazards* caused by malfunctioning behavior of Electrical/Electronic systems" [ISO 26262]
- In a nutshell, functional safety is about ensuring the safe operation of systems even when they go wrong
- Functional safety is critical to many markets: Aerospace, Medical, Industrial, Automotive, etc.

V-Diagram: Automotive View of "Design"



Functional Safety Standards

- IEC 61508: Base functional safety standard
- ISO 26262: Automotive functional safety standard
  - Derived from IEC 61508, published 2011
    - Part 1: Vocabulary
    - Part 2: Management of Functional Safety
    - Part 3: Concept Phase
    - Part 4: Product Development: System Level
    - Part 5: Product Development: Hardware Level
    - Part 6: Product Development: Software Level
    - Part 7: Production and Operation
    - Part 8: Supporting Processes
    - Part 9: ASIL Orientated and Safety Oriented Analysis
    - Part 10: Guideline on ISO 26262
    - Part 11: Application of ISOS 26262 to Semiconductors (2nd Edition)

2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Functional Safety Verification for Automotive IPs/SoCs

ISO 26262-5 Product Development: Hardware Level, Part 8, and Part 11

1. Vocabulary

2. Management of functional safety

3. Concept phase
- 3-6 Hazard analysis and risk assessment (HARA)

4. Product development at the system level
- 4-7 System architectural design

7. Production, operation, service

5. Product development at the HW level
- 5-6 Specification of hardware safety requirements
- 5-8 Evaluation of the hardware architectural metrics
- 5-9 Evaluation of the safety goal violations due to random failures

6. Product development at the SW level
- 6-6 Specification of software safety requirements
- 6-10 Software integration and verification
- 6-11 Testing of the embedded software

8. Supporting processes
- 8-6 Specification and management of safety requirements
- 8-7 Configuration management
- 8-8 Change management
- 8-10 Documentation management
- 8-11 Confidence in the use of software tools
- 8-12 Qualification of software components

9. ASIL-oriented and safety-oriented analyses

10. Guideline on ISO 26262

11. Guideline on application of ISO 26262 to semiconductors

Source: ISO/DIS26262:2016: Overview of ISO26262

Safety Goals/Requirements

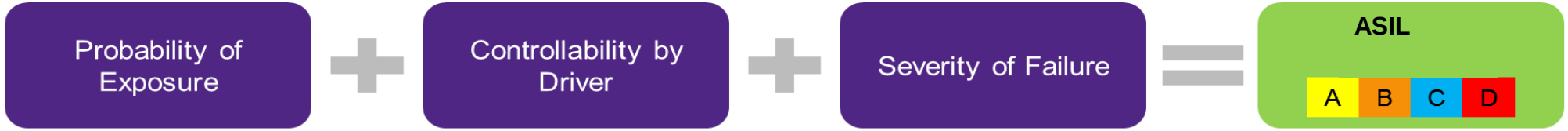
- Safety Goal
  - Top-level safety requirement
  - Derived from Hazard Analysis and Risk Assessment (HARA)

- Example(s)
  - Unintended activation of emergency brake must be prevented
  - Unintended inflation of airbags must be prevented.

2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Hazard Analysis and Risk Assessment (HARA) (OEMs)

- Automotive Safety Integrity Level (ASIL)



E0	Combination of Very low Probabilities
E1	Very Low Probability *(less often than once a year for the great majority of drivers)*
E2	Low Probability *(a few times a year for the great majority of drivers)*
E3	Medium Probability *(once a month or more often for an average driver)*
E4	High Probability *(almost every drive on average)*

C0	Controllable in general
C1	Simply controllable *(99% or more of all drivers are usually able to avoid a harm)*
C2	Normally controllable *(90% or more of all drivers are usually able to avoid a harm)*
C3	Difficult to control or Uncontrollable *(Less than 90% of all drivers are usually able or barely able to avoid a harm)*

S0	No injuries
S1	Light and moderate injuries
S2	Severe and life-threatening injuries (survival possible)
S3	Life threatening injuries (survival uncertain), fatal injuries

Severity	Probability	C1	C2	C3
S1	**E0**	QM	QM	QM
	E1	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	**A**
	E4	QM	**A**	**B**
S2	**E0**	QM	QM	QM
	E1	QM	QM	QM
	E2	QM	QM	**A**
	E3	QM	**A**	**B**
	E4	**A**	**B**	**C**
S3	**E0**	QM	QM	QM
	E1	QM	QM	**A**
	E2	QM	**A**	**B**
	E3	**A**	**B**	**C**
	E4	**B**	**C**	**D**

Safety Element out of Context (SEooC)

- Chips and IPs are usually Safety Elements out of Context

Issue

No/little knowledge of the system in which the design is used

- Hazards
- Safety goals
- Architecture

Resolution

SEooC vendors need to specify Assumptions of Use **(AoU)**

- Safety requirements
- Expected integration environments and requirements

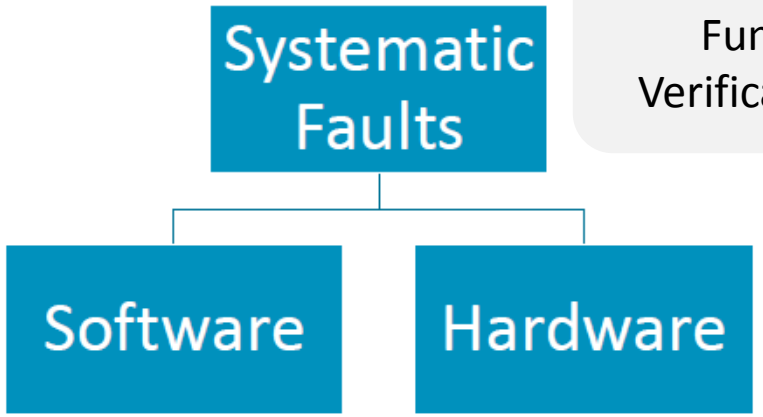
SEooC vendors should aim at highest expected **ASIL**

- Fault avoidance
- Fault control
- Independent confirmation measures

Digital Logic Failure Modes

Systematic failure / fault

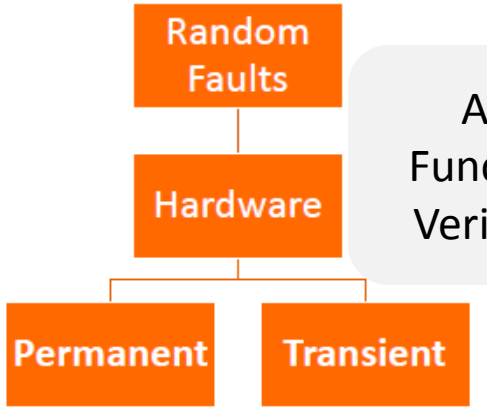
- Failure related in a deterministic way to a certain cause to be eliminated by a modification of the design or of the manufacturing process, operational procedures and documentation



Found/covered by Functional Verification tools

Random hardware failure / fault

- Failure occurring at a random time which results from degradation mechanisms in the hardware



Assessed by Functional **Safety** Verification tools

d.c. faults stuck-at
stuck-open
shorts

"soft errors"
SBU = single bit upset
MBU = multiple bit upset
SET = single event transient

Safety Fault Metrics for ISO 26262 ASIL Ratings

- Fault Injection Testing recommended for ASIL A & B and highly recommended for ASIL C & D

Method	ASIL A	ASIL B	ASIL C	ASIL D
Fault Injection Testing	+	+	+ +	+ +

- Maximize detection of single point and multi-point latent faults

Metric	ASIL B	ASIL C	ASIL D
Single Point Fault Metric	≥ 90%	≥ **97%**	≥ **99%**
Latent Fault Metric	≥ 60%	≥ **80%**	≥ **90%**

- Probabilistic Metric of Hardware Failure (PMHF)

Metric	ASIL B	ASIL C	ASIL D
PMHF (FIT Rate)	**100**	**100**	**10**

ISO 26262 Safety Principles

Prevent / Eliminate Bugs

Avoid Systematic Faults – Design Bugs (Permanent Faults)

Implementation:
Use best practice/certified design flows

Verification & Validation:
Use best-in-class Functional Verification methodology

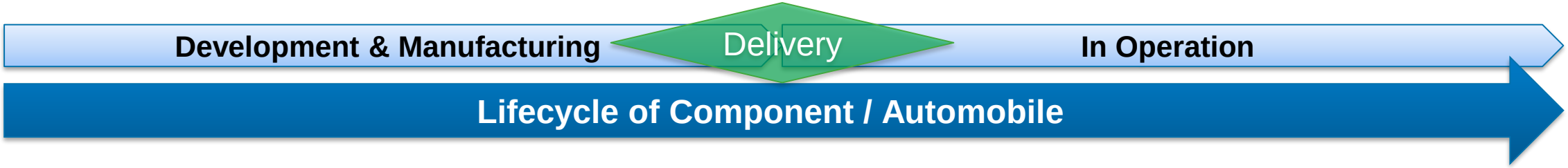
Control Failures

Control of Systematic Faults – Bug Escapes (Permanent Faults)

Control of Random Faults – H/W Failures (Permanent or Transient Faults)

Implementation:
Deploy comprehensive Safety Mechanisms

Verification & Validation:
Follow ISO 26262 recommendations for **ASIL** level

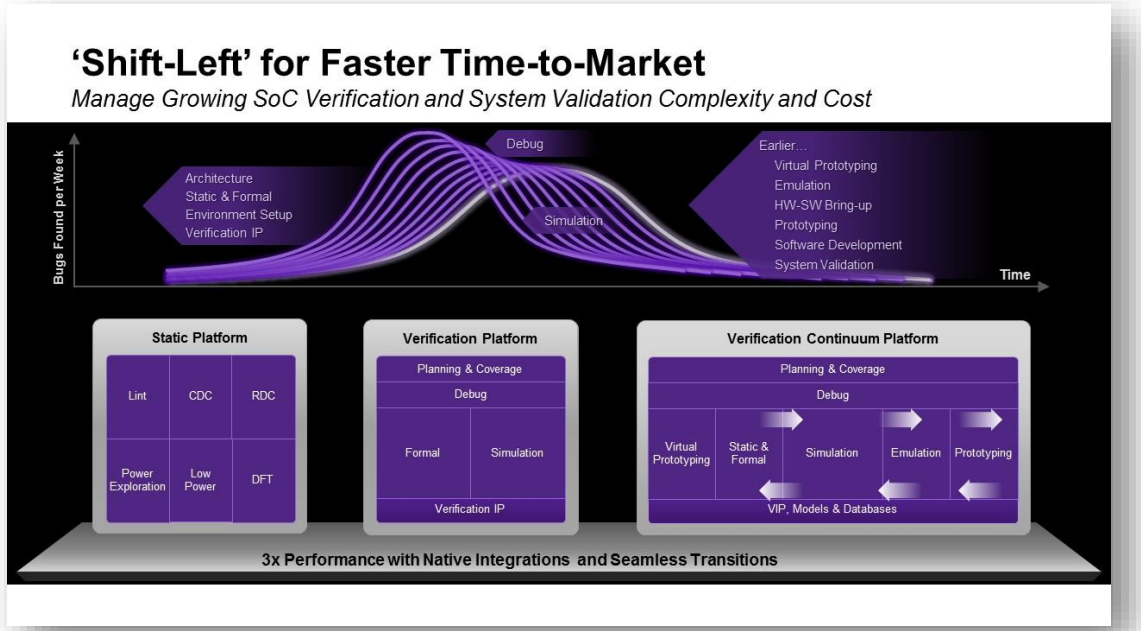


Functional Verification is Essential Starting Point

Prevent / Eliminate Bugs

Avoid Systematic Faults – Design Bugs (Permanent Faults)

Verification & Validation:
Use best-in-class Functional Verification methodology



Functional Verification Technology Platforms

- Many technologies must be used to ensure the highest functional verification quality
- Early software bring-up enables faster and more complete verification
- Verification quality analysis provides objective measure of functional verification effectiveness

Functional Safety Verification – Verify Control of Hardware Failures

- Hardware failures are modeled as both systematic and random faults (which may be permanent or transient)
- ISO 26262 recommends fault injection testing to verify the effectiveness of the Safety Mechanisms
- Results and reports from fault injection testing are essential for ISO 26262 FMEDA work product

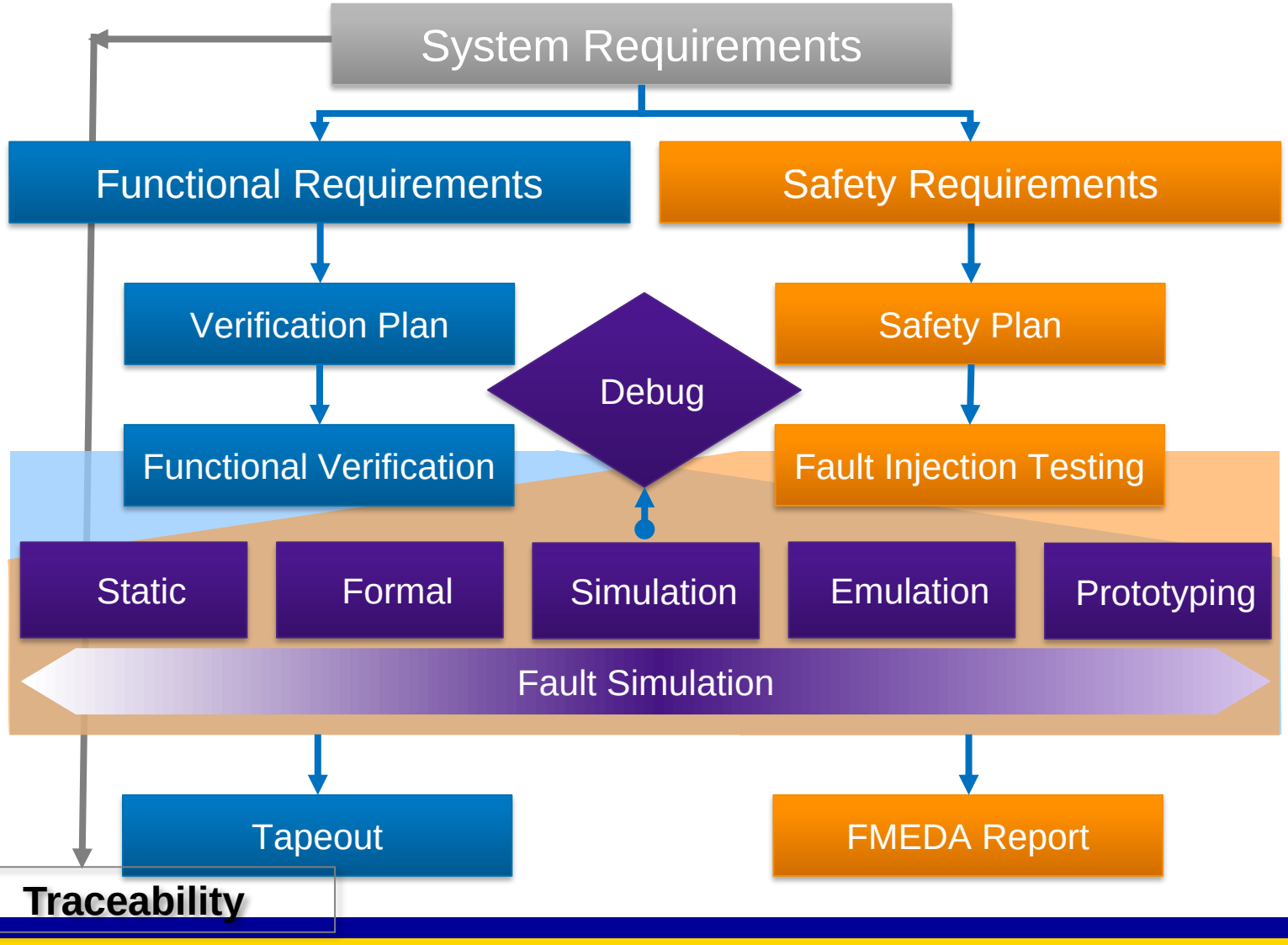
Control Failures

Control of Systematic Faults – Bug Escapes (Permanent Faults)

Control of Random Faults – H/W Failures (Permanent or Transient Faults)

Verification & Validation:
Follow ISO 26262 recommendations for **ASIL** level
Determine Diagnostic coverage by fault simulation

Verification Flow Alignment



- **Alignment of requirements for functional and safety verification**
  - **Accelerate complete verification process**
  - **Requires solution for systematic and random fault testing**
- **Integrated with ISO 26262 Flows**
  - **Failure mode effects analysis**
  - **Safety plan traceability and results**

Verification Goal Comparison

Functional Verification
Prevent / Eliminate Bugs

- Validate functional correctness of design
- Unified verification technologies with fastest engines
- Development and manufacturing testing
- Avoid Systematic Faults

Functional Safety Verification
Control Failures

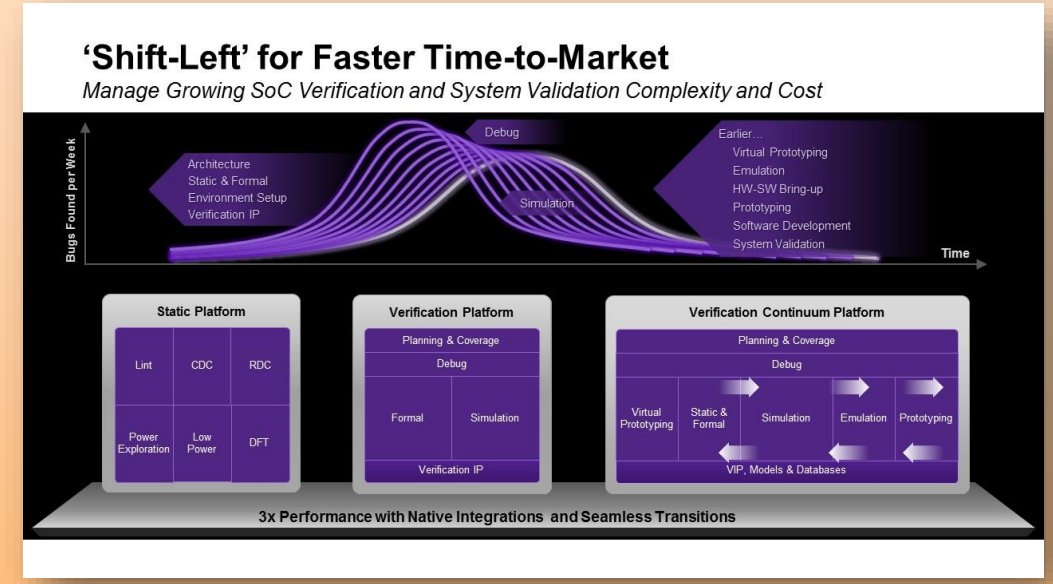
- Confirm robustness of safety mechanisms
- Confidence in tool chain
- “In Operation” testing
- Control of Random Faults

2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Verification Goal Comparison

Functional Verification
Prevent / Eliminate Bugs

Unified verification technologies with fastest engines



Functional Safety Verification
Control Failures

Certified tool chain

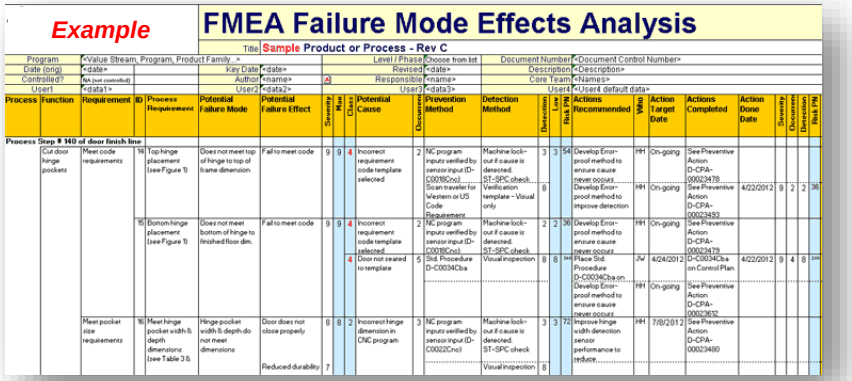
SGS TÜV SAAR
ASIL D READY
Functional Safety
www.sgs-tuev-saar.com

2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

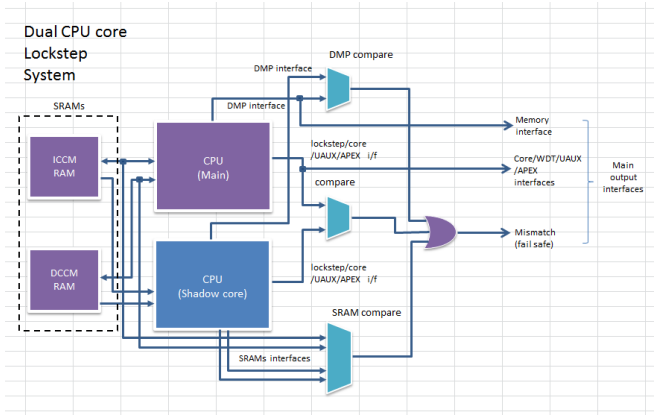
Functional Safety Process

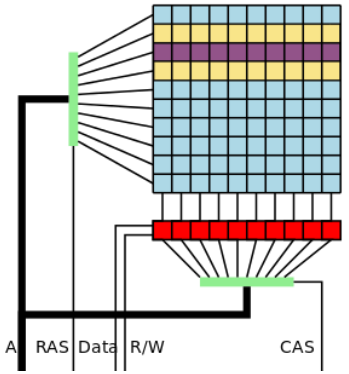
Implement and Confirm Quality of Safety Mechanisms (SM)

- Define Failure Mode and Effects Analysis (FMEA) for device
- Implement Safety Mechanisms to protect against failures
- Run fault injection to get ISO 26262 metrics
- Generate FMEDA report, Safety manual



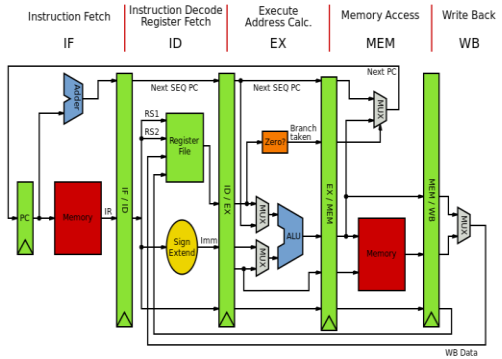
Dual-Core Lockstep





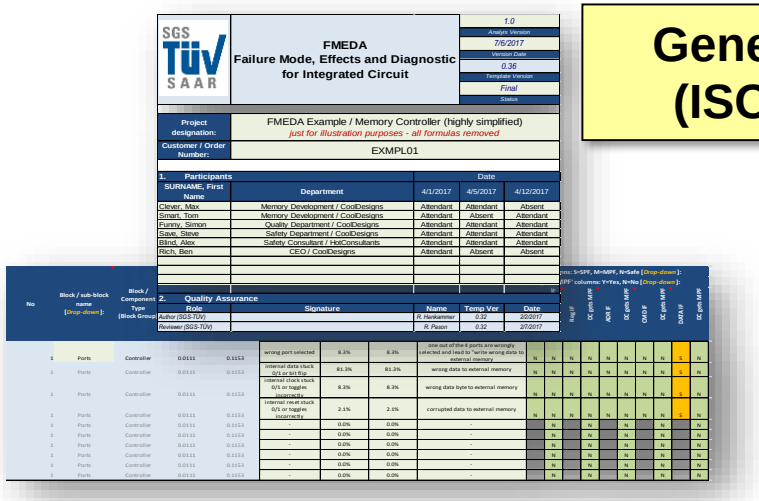
ECC Memory Protection

Software Test Libraries



Custom Safety Mechanisms

Unique Functional Safety Needs - Summary


FMEDA
Failure Mode, Effects and Diagnostic for Integrated Circuit

Project designation: FMEDA Example / Memory Controller (highly simplified)
just for illustration purposes - all formulas removed
Customer / Order Number: EXMPL01

1. Participants

SURNAME, First Name	Department	Date 4/1/2017	4/5/2017	4/12/2017
Clever, Max	Memory Development / CoolDesigns	Attendant	Attendant	Absent
Smart, Tom	Memory Development / CoolDesigns	Attendant	Absent	Attendant
Funny, Simon	Quality Department / CoolDesigns	Attendant	Attendant	Attendant
Save, Steve	Safety Department / CoolDesigns	Attendant	Attendant	Attendant
Blind, Alex	Safety Consultant / HotConsultants	Attendant	Attendant	Attendant
Rich, Ben	CEO / CoolDesigns	Attendant	Absent	Absent

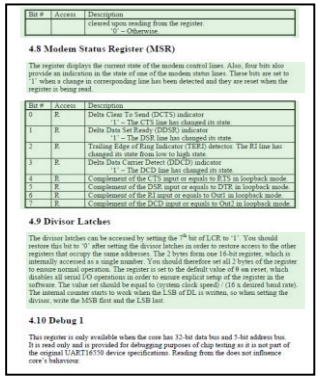
2. Quality Assurance

Role	Signature	Name	Temp Ver	Date
Author (SGS-TÜV)		R. Hankammer	0.32	2/2/2017
Reviewer (SGS-TÜV)		R. Pisani	0.32	2/5/2017

Generate FMEDA reports (ISO-26262 deliverable)

Certified toolchain

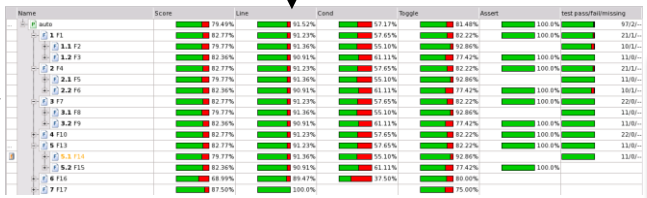
Traceability


4.8 Modem Status Register (MSR)

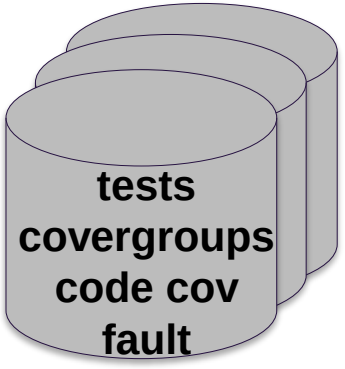
4.9 Divisor Latches

4.10 Debug 1

Safety Requirements



Safety Verification Plan



Safety Coverage

2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Mutation Analysis for Qualifying Verification Process

ISO 26262 Functional Safety Principles

Prevent / Eliminate Bugs

Avoid Systematic Faults – Design Bugs (Permanent Faults)

Implementation:
Use best practice/certified design flows

Verification & Validation:
Use best-in-class Functional Verification methodology

Control Failures

Control of Systematic Faults – Bug Escapes (Permanent Faults)

/W Failures (Permanent or Transient Faults)

Implementation:
Deploy comprehensive Safety Mechanisms

Verification & Validation:
Follow ISO 26262 recommendations for **ASIL** level

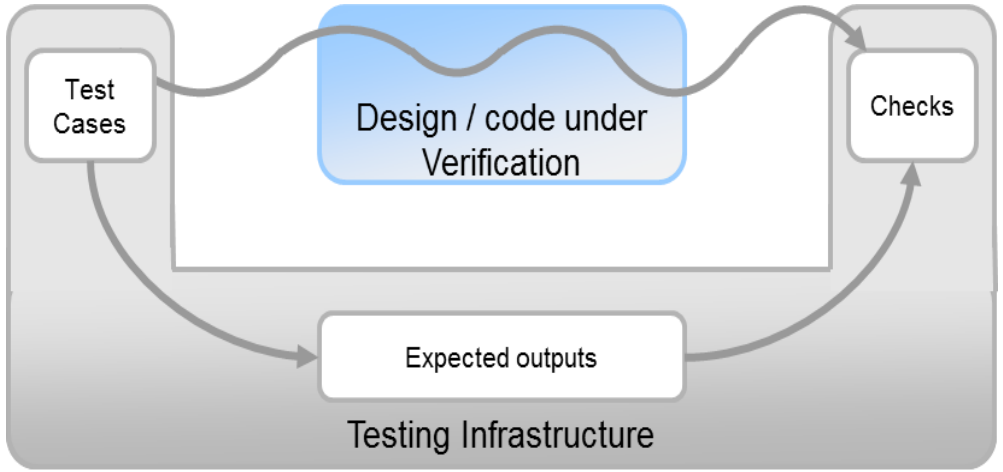
Ensure Avoidance of Systematic Faults

Development & Manufacturing — Delivery — **In Operation**

Lifecycle of Component / Automobile

Quality of Verification

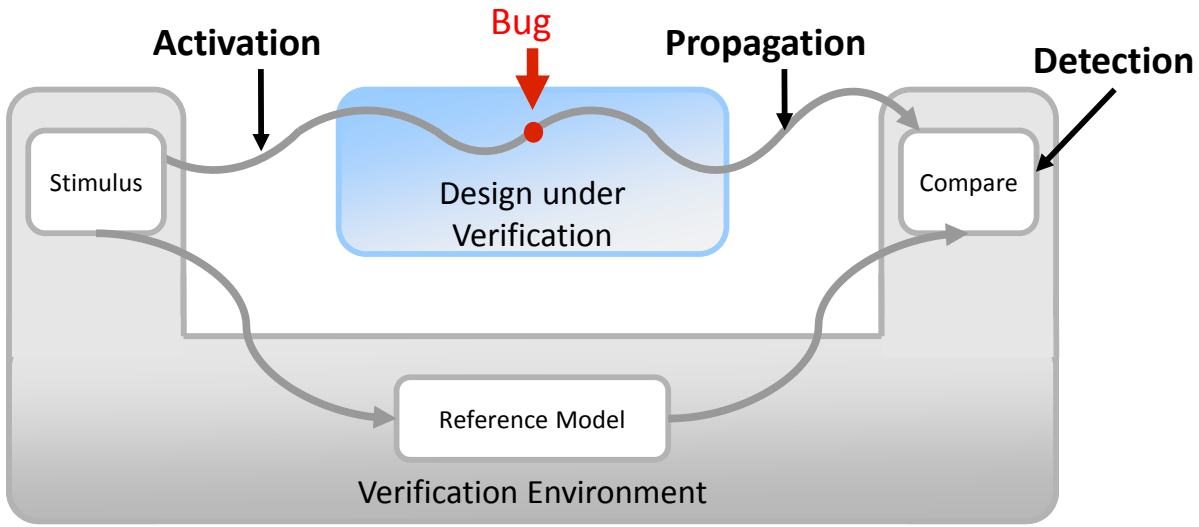
- Failing tests are debugged
- False Positives are silent
- False Negatives and False Positives are bugs in *verification*



		Design has bug(s)	
		Yes	**No**
Verification result is	**Pass**	False Positive	Ok (done)
	Fail	Ok (debug design)	False Negative

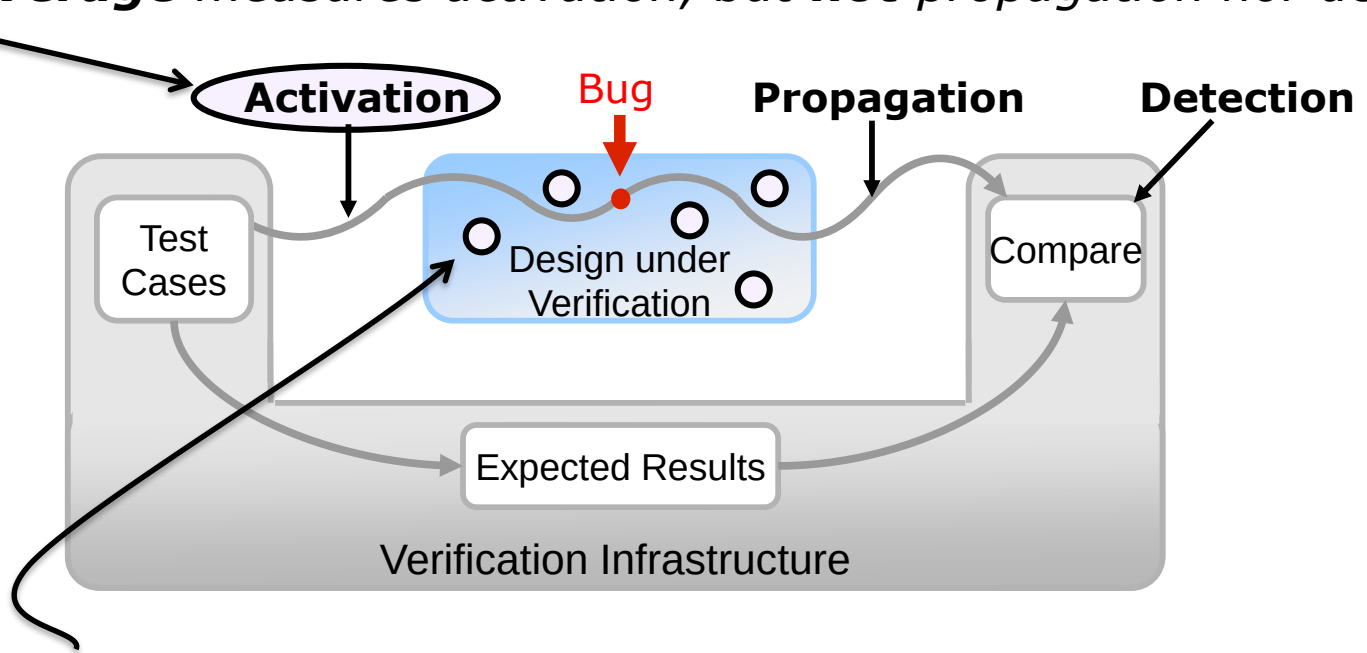
Effective Verification

- Applies universally



Assessing Verification Effectiveness

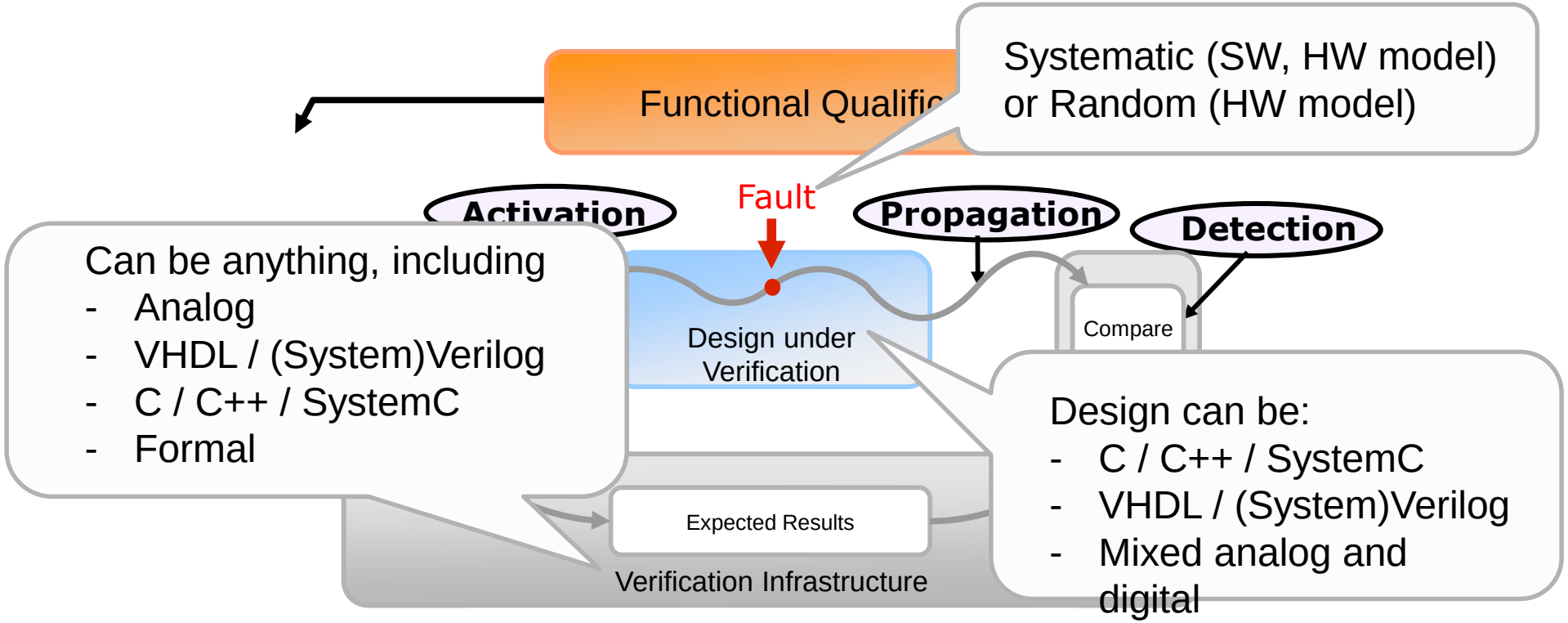
Code coverage *measures activation, but* ***not*** *propagation nor detection*



Functional coverage *checks "important" functional points, however comprehensiveness of functional points is unknown*

Mutation-Based Analysis Concept

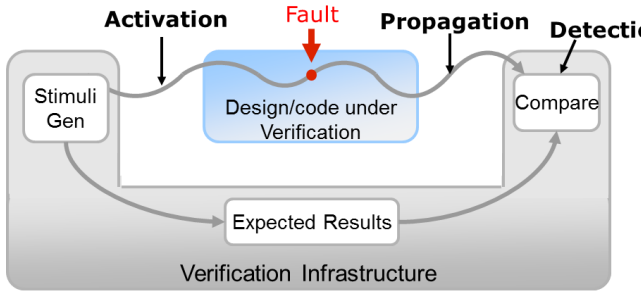
- **Automatically inserts** “artificial bugs” called **faults** into the design
- Runs verification process on “broken” design
- Measures the ability of the environment to activate, propagate, and detect faults



How Does Mutation Work?

- **Modifies design code to insert defects**

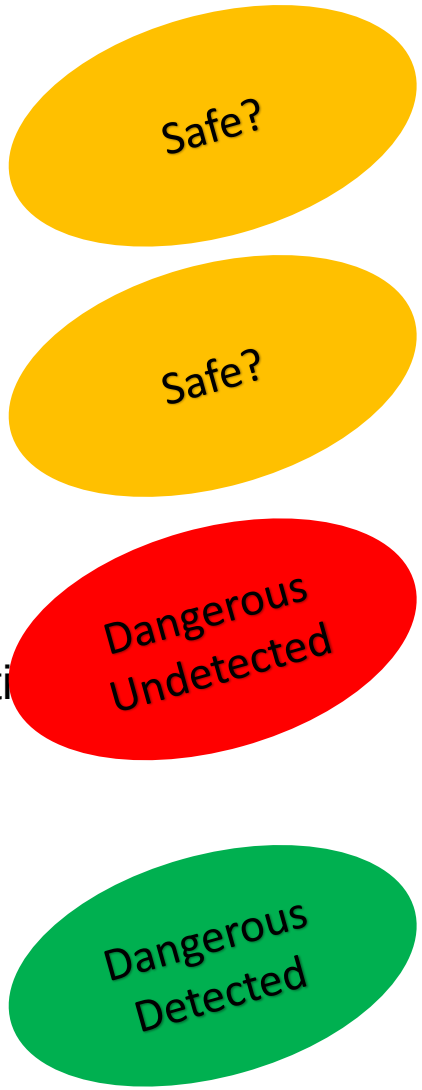
```
o1 = f(i1)      → o1 = 1'b0          // tie to constant

if (a)          →  if (TRUE)         // force execution
  f1();              f1();  // of "if" branch
else                 else
  f2();              f2();

a = b | c       →  a = b & c         // change operator
```



- **Pass the broken design to the verification**
  - Does at least one test fail? Environment is robust
  - Do all tests pass? Problem with verification environment

Interpreting Results

- Non-activated (NA)
  - Stimulus does not exercise the fault
    - Similar to code (line) coverage
- Non-propagated (NP)
  - Stimulus exercises the fault
    - But no difference seen at observation points vs. passing simulation
- Non-detected (ND)
  - Stimulus exercises the fault
  - A difference(s) propagates to observation point(s) vs. passing simulati
    - But all tests PASS
- Detected (D)
  - Stimulus exercises the fault
  - At least one test FAILs
    - OK



Systematic Failure Methodology

- **Fault Reduction Technology**
  - **Remove Equivalent Faults**
    - Faults which do not change the design due to dead logic or redundant code
  - **Prioritize fault injection**
    - Top 2 fault classes can expose big problems quickly
  - **Drops related faults when a fault is non-detected**
    - May also be non-detected and would point to same weakness

- **Methodology**
  - **Leverage the verification infrastructure**
    - Submit multiple test runs in parallel
  - **Start with a small set of tests/seeds**
    - A “smoke suite” will quickly find missing checkers/assertions that won’t “appear” if you simply add more tests
  - **Iterate**
    - Fix problems as they are found and then continue

Phases of Functional Qualification

Model

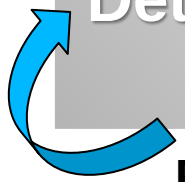
- Parse the design to determine faults to insert
- Search for unreachable faults
- Determine cones of influence
- Create Instrumented Files for the next 2 phases

Activate

- Run every test once
- Determine which tests activate each fault
- Determine which faults are not activated

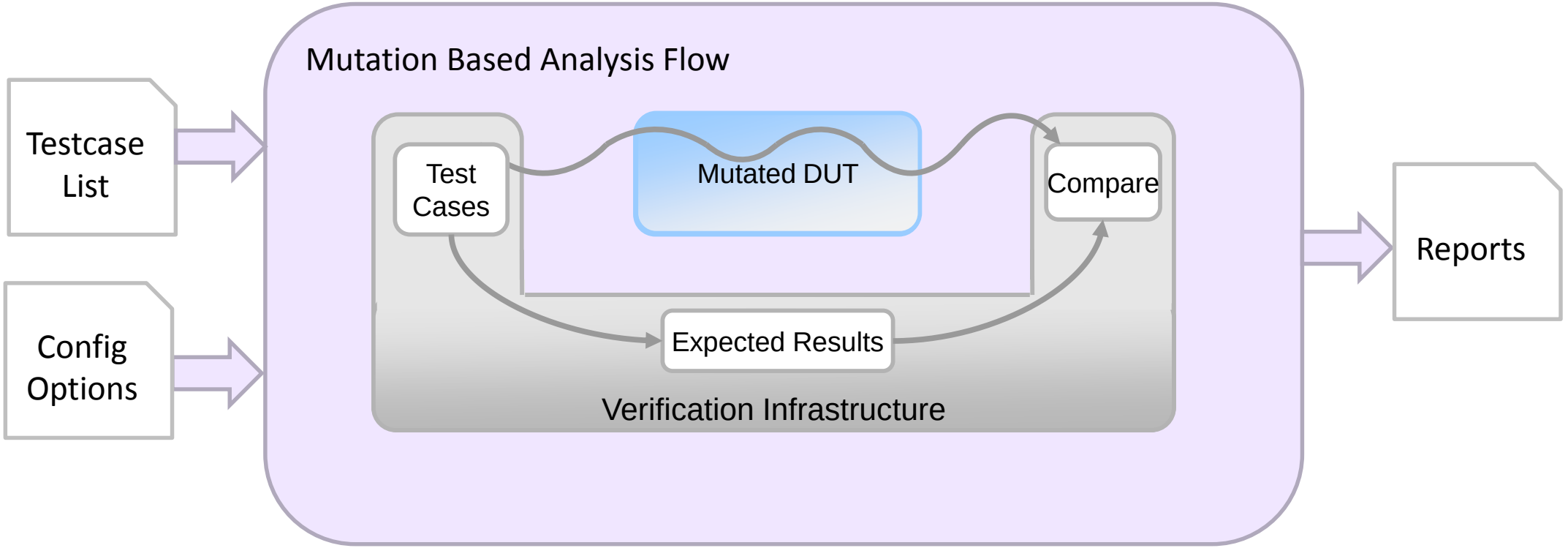
Detect

- Iterate enabling a fault in the Design
- Run tests activating the fault
- Determine if any test is capable of propagating and detecting the fault

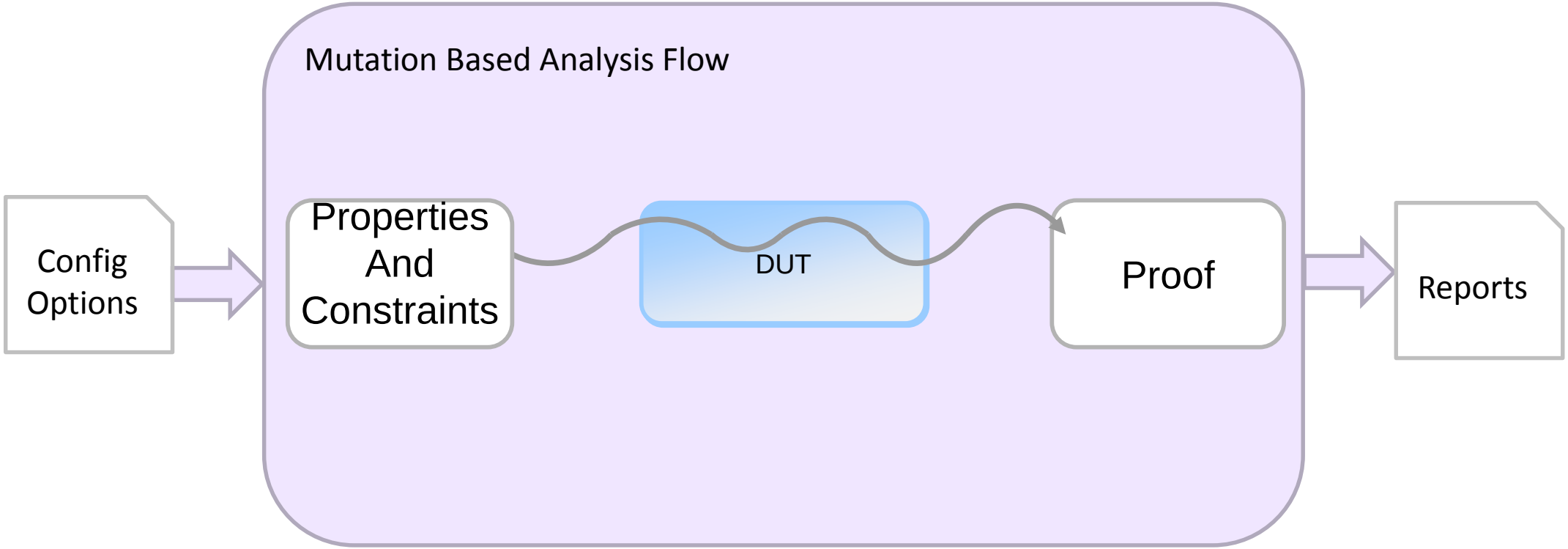


Fix and iterate as problems are found

Easy Integration Within Existing Environments



Formal Verification



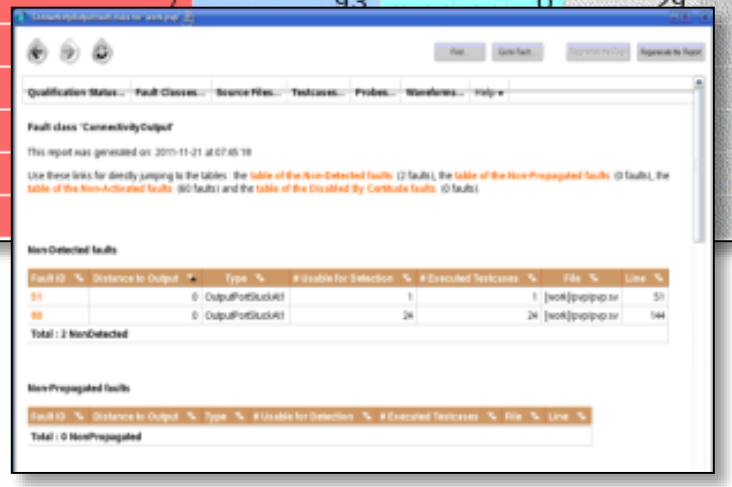
Detailed Fault Reports

Qualification Status... | Fault Classes... | Source Files... | Testcases... | Probes... | Waveforms... | Help

Fault classes for 'dag_top'

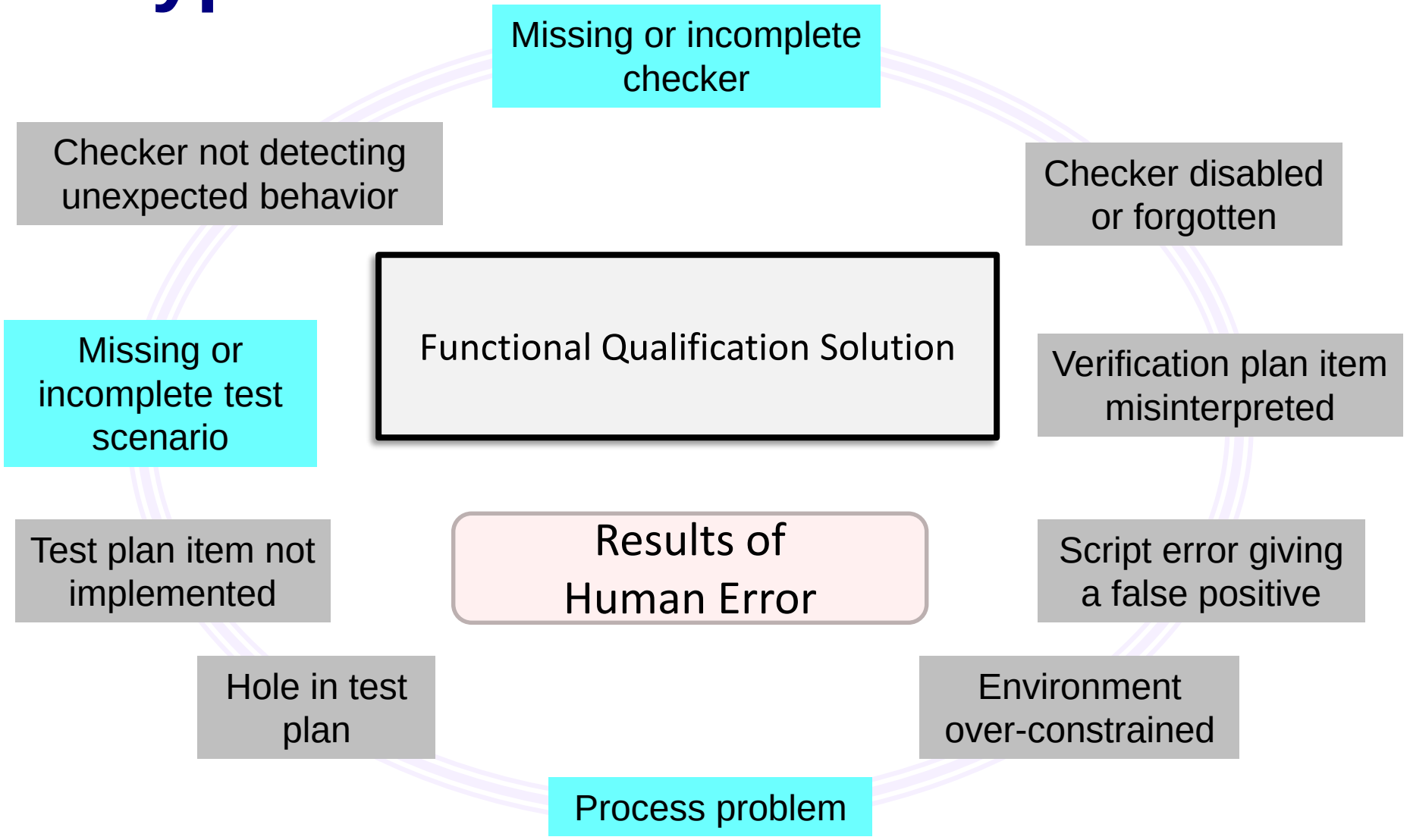
This report was generated on: 2011-02-10 at 15:50:50

Class name	Faults in Design	Faults in List	Non-Activated	Non-Propagated	Detected	Non-Detected	Disabled By Certitude	Disabled By User	Dropped	Not Yet Qualified
ConnectivityOutput	183	183	16	0	107	19	5	0	36	0
ResetConditionTrue	261	261	0	89	149	11	4	0	0	8
SynchronousControlFlow	797	797	2	29	370	4	48	0	338	6
ConnectivityInput	1596	1596	103	18	527	10	98	0	840	0
SynchronousDeadAssign	331	331	0	19	183	7	93	0	29	0
ComboLogicControlFlow	762	762	23	25	253					0
SynchronousLogic	532	532	3	13	246					0
ComboLogic	8958	8958	14	101	584					428
OtherFaults	1869	3	0	0	0					0
All Fault Classes (9)	**15289**	**13423**	**161**	**294**	**2419**					**442**



Results by Fault Class

Typical Problems Found



2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Functional Safety Verification Flow
FMEA to FMEDA

ISO 26262 Work Products

- FMEA, FMEDA
  - **F – Failures of a given component** Consider a component in a system
  - **M – Mode** Look at one of the ways in which it can fail
  - **E – Effects** Determine the effects this failure mode will cause to the system we are examining
  - **D – Diagnostic** Determine the coverage
  - **A – Analysis** Analyze how much impact the symptom will have on the environment/people/ the system itself

Source:https://about.brighton.ac.uk/cem/research/seminars/2011/fmea_pres.pdf

Failure Mode Effect Analysis (FMEA)

- Systematic method of failure analysis
  - **For each element**:
    - Identify the manner in which a **failure can occur**
    - Identify the **consequences of the failure**
    - Identify the **probability/severity of the failure**

- Common entry systems
  - Excel spreadsheet
  - Commercial tools

FMEA Components

- Checkbox of items in an FMEA
  - Block Diagram
  - Block List
  - Failure Modes
  - Potential Cause of Failure
  - Safety Mechanism
  - RPN (Risk Priority Number)
  - Estimated Coverage

FMEA Inputs example

- Design block level list and diagram.

Reset Logic

Flag Logic

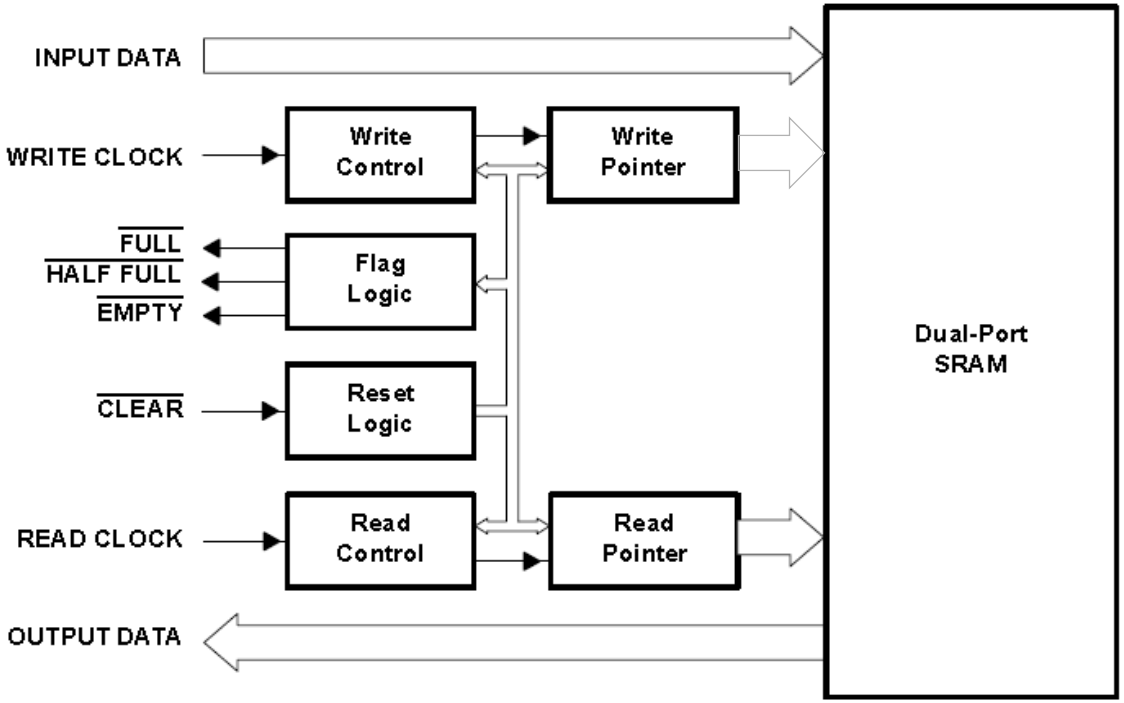
Read Control

Read Pointer

Write Control

Write Pointer

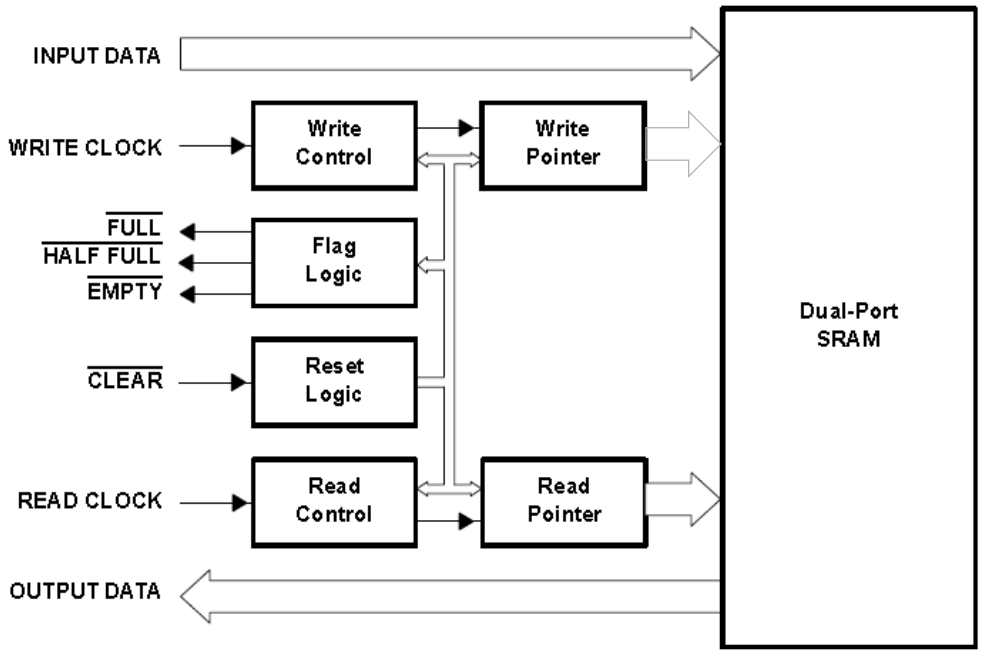
SRAM



Block Diagram of FIFO with Static Memory

FMEA Failure Mode analysis example

- Failure Mode 1:
  - Failure: Full signal is not raised when FIFO is full
  - Effect: Data will be overwritten
  - Safety Mechanism: Redundant read/write pointers
- Failure Mode 2:
  - Failure: Data in FIFO is corrupted
  - Effect: Invalid data
  - Safety Mechanism: ECC



Block Diagram of FIFO with Static Memory

FMEA Work product example:

PRIMARY SAFETY MECHANISMS

Element	Unique ID	Safety Mechanism	Diagnostic or Avoidance?	Type	Requirements ID	Periodicity	Execution Time	Error Response	Error Reporting Time	Equivalent ISO 26262 Diagnostic	ISO 26262 DC	Estimated DC
Host	HOST_PSM_1	Host Safety 1	Avoidance	HW (internal)		continuous	Real-time	Interrupt	1ms	hardware consistency monitoring	High	Medium
Host	HOST_PSM_2	Host Safety 2	Diagnostic	HW (internal)		continuous	Real-time	Interrupt	1ms	Processing units: Other sub-elements::Parity bit	Low	Medium
Host	HOST_PSM_3	Host Safety 3	Avoidance	HW (internal)		continuous	Real-time	Interrupt	1ms	hardware consistency monitoring	High	Medium
Host	HOST_PSM_4	Host Safety 4	Diagnostic	HW (internal)		continuous	Real-time	Interrupt	1ms	over/under flow detection	Low	Medium

MAIN FMEA

Unique ID	Top Design Element	Element -1	Element -2	Potential Faults	Potential Errors (as seen at top design element boundary)	Potential Effect(s) of Failure (visible to system)	System-Level Potential Effect Class	ISO 26262 Equivalent Fault/Error/Failure	Severity [Optional]	R
HOST_FM_1	MEM_CTRL			corrupted CPU com	Memory content corruptio	Wrong coding, wrong or no execution	CPU/GPU::Unintended instruction(s) flow executed	rocessing units: Other sub-elements::d.c. fault mode	5	
HOST_FM_2	MEM_CTRL			corrupted CPU com	Memory content corruptio	Wrong coding, wrong or no execution	CPU/GPU::Unintended instruction(s) flow executed	units: ALU - Data Path::Soft error model (for seque	5	
HOST_FM_3	MEM_CTRL			corrupted CPU write	Memory content corruptio	Wrong coding, wrong or no execution	CPU/GPU::Unintended instruction(s) flow executed	rocessing units: Other sub-elements::d.c. fault mode	5	
HOST_FM_4	MEM_CTRL			corrupted CPU write	Memory content corruptio	Wrong coding, wrong or no execution	CPU/GPU::Unintended instruction(s) flow executed	units: ALU - Data Path::Soft error model (for seque	5	
HOST_FM_5	REG_UNIT			incorrect registers r	Memory content corruptio	Processor architectural state/control corrup	CPU/GPU::Unintended instruction(s) flow executed	rocessing units: Other sub-elements::d.c. fault mode	5	
HOST_FM_6	REG_UNIT			incorrect registers r	Memory content corruptio	Processor architectural state/control corrup	CPU/GPU::Unintended instruction(s) flow executed	units: ALU - Data Path::Soft error model (for seque	5	
HOST_FM_7	REG_UNIT			incorrect registers v	Memory content corruptio	Processor architectural state/control corrup	CPU/GPU::Unintended instruction(s) flow executed	rocessing units: Other sub-elements::d.c. fault mode	5	
HOST_FM_8	REG_UNIT			incorrect registers v	Memory content corruptio	Processor architectural state/control corrup	CPU/GPU::Unintended instruction(s) flow executed	units: ALU - Data Path::Soft error model (for seque	5	

Failure Mode Effect &Diagnostic Analysis (FMEDA)

- A detailed analysis technique to obtain:
  - Design failure rates
  - Failure Modes diagnostic capability
- FMEDA is an extension of the FMEA analysis
  - Assessing the **Safety Metrics** for the given Failure Mode

FMEDA Inputs

- Technology Information for Failure In Time (FIT)
  - Needed to compute Failure Rates

ISO 26262 acceptable technology standards:
- IEC TR 62380
- SN 29500
- FIDES Guide

- Design information
  - Digital logic and analog area, flop/latch, RAM/ROM counts
    - Needed to compute Failure Mode Distribution

- Safety Mechanism (if exists) for the Failure Modes

Failure Mode (FM) Distribution

- Each FMEDA needs to have a base Failure Rate assigned to it
- Possible distributions:
  - **Uniform**: Each FM has a failure rate equal to the overall failure rate divided by the number of failure modes
    - Reasonable assumption for initial analysis; assumes highly symmetrical design
  - **Area**: Each FM's failure rate depends on its relative portion of the design area
    - Similarly, it may depend on the number of gates/flops
  - **Number of outputs affected**
    - Considers their cone of influence

FMEDA Diagnostic Coverage Components

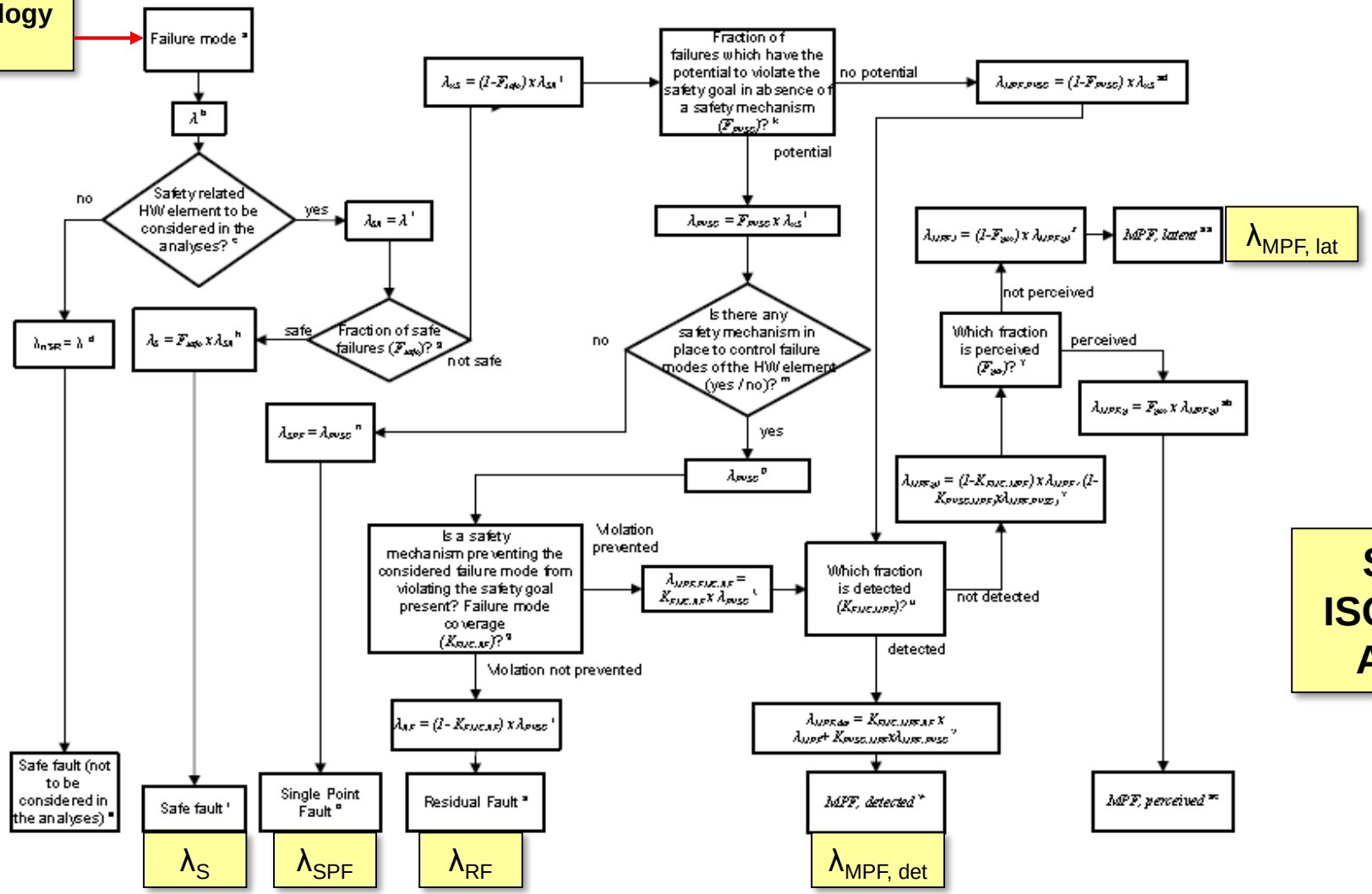
- **Fault list** – a list of design locations with potential random failures
  - Based on FMEA potential cause of failure
  - Generated from block level or elementary sub parts
- **Observation Points**
  - Design points in which the effect of an injected fault should be observed
    - Normally –at the boundary of a block in which the fault is injected
- **Diagnostic Points**
  - Design points which are activated when the safety mechanism detects the injected fault
    - e.g.: safety_alarm IO pin, interrupt to interrupt controller etc.

FMEDA Diagnostic Coverage Components – cont.

- **Workloads**
  - These are sets of tests which stimulate the area of the injected fault
  - Types of workloads:
    - **Representative**: follow normal use cases, do not necessarily activate all signals in the relevant block
    - **Exhaustive**: provide 100% toggle coverage of the relevant block

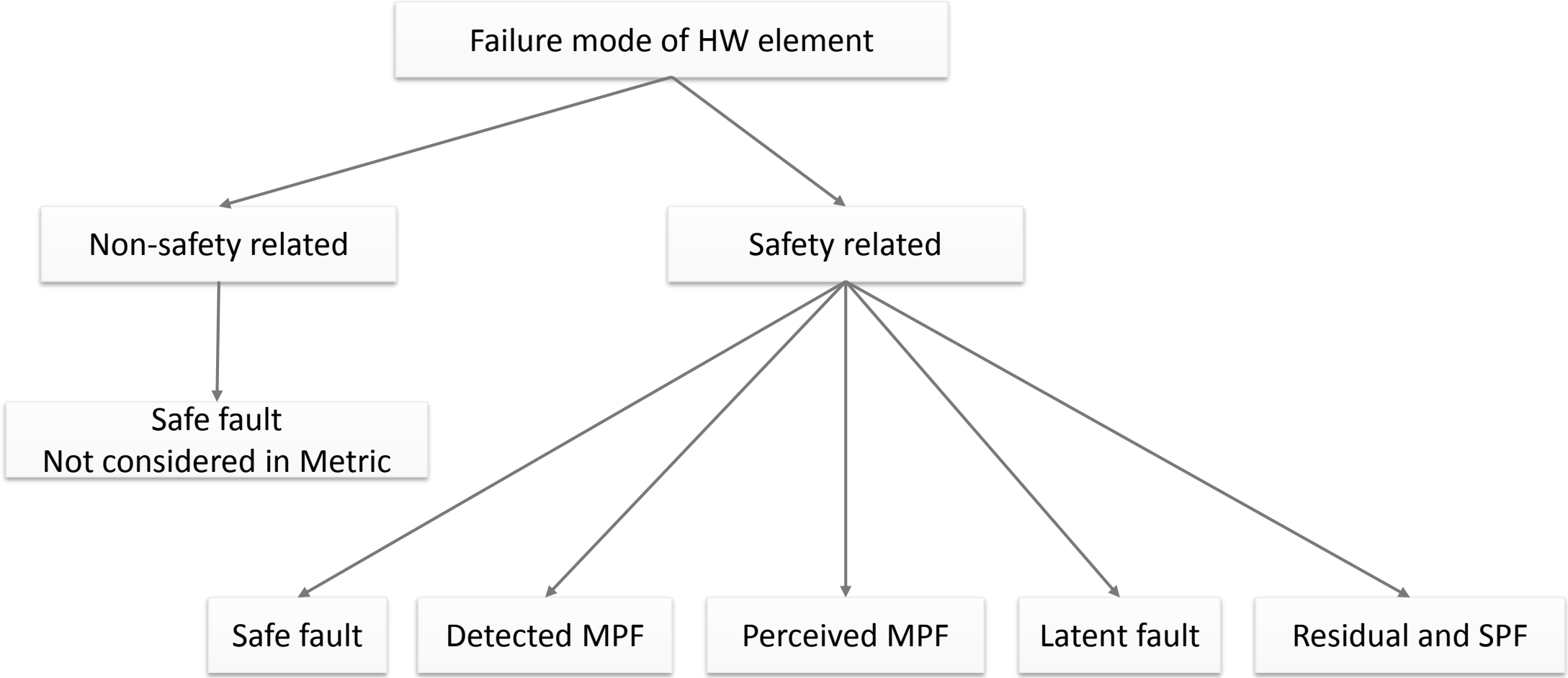
ISO 26262 Fault Classification

Technology FIT



Source: ISO 26262-5 Annex B

Fault Classification Simplified



Faults Classification (1)

- Safe Faults (for calculating λ_S)
  - Faults which will not violate a safety goal
  - Example:
    - Faults in CPU debug logic
- Single Point Faults (for calculating λ_{SPF})
  - A single fault which can lead to a violation of a safety goal
  - Not protected by a SM
  - Example:
    - Interconnect with no protection for data of address buses

Faults Classification (2)

- Residual Faults (for calculating λ_{RF})
  - A single fault which can lead to a violation of a safety goal
  - Not detected by a SM (SM does not have 100% coverage)
  - Example:
    - A memory fault which is not detected by memory diagnostics (e.g. checkerboard test)

Faults Classification (3)

- Detected Dual (Multi) Point Fault ($\lambda_{MPF,det}$)
  - A fault in combination with another fault which leads to a violation of a safety goal
  - Detected by the SM
  - Example:
    - A memory bit with a permanent fault which is protected by parity and activates a warning light
    - A fault in the parity logic leads to a violation of the safety goal
    - Self Test of the parity logic can detect the fault in it

Faults Classification (4)

- Latent Dual (Multi) Point Fault ($\lambda_{MPF,l}$)
  - A fault in combination with another fault which leads to a violation of a safety goal
  - Is not detected by the SM
  - Example:
    - A memory bit with a permanent fault which is corrected by ECC but does not activate a warning light
    - A fault in the ECC would lead to a violation of the safety goal

ISO 26262 Metric (part 5 Annex C)

- Failure Rate:

$$\lambda = \lambda_{SPF} + \lambda_{RF} + \lambda_{MPF} + \lambda_{S} \quad \text{(C.1)}$$

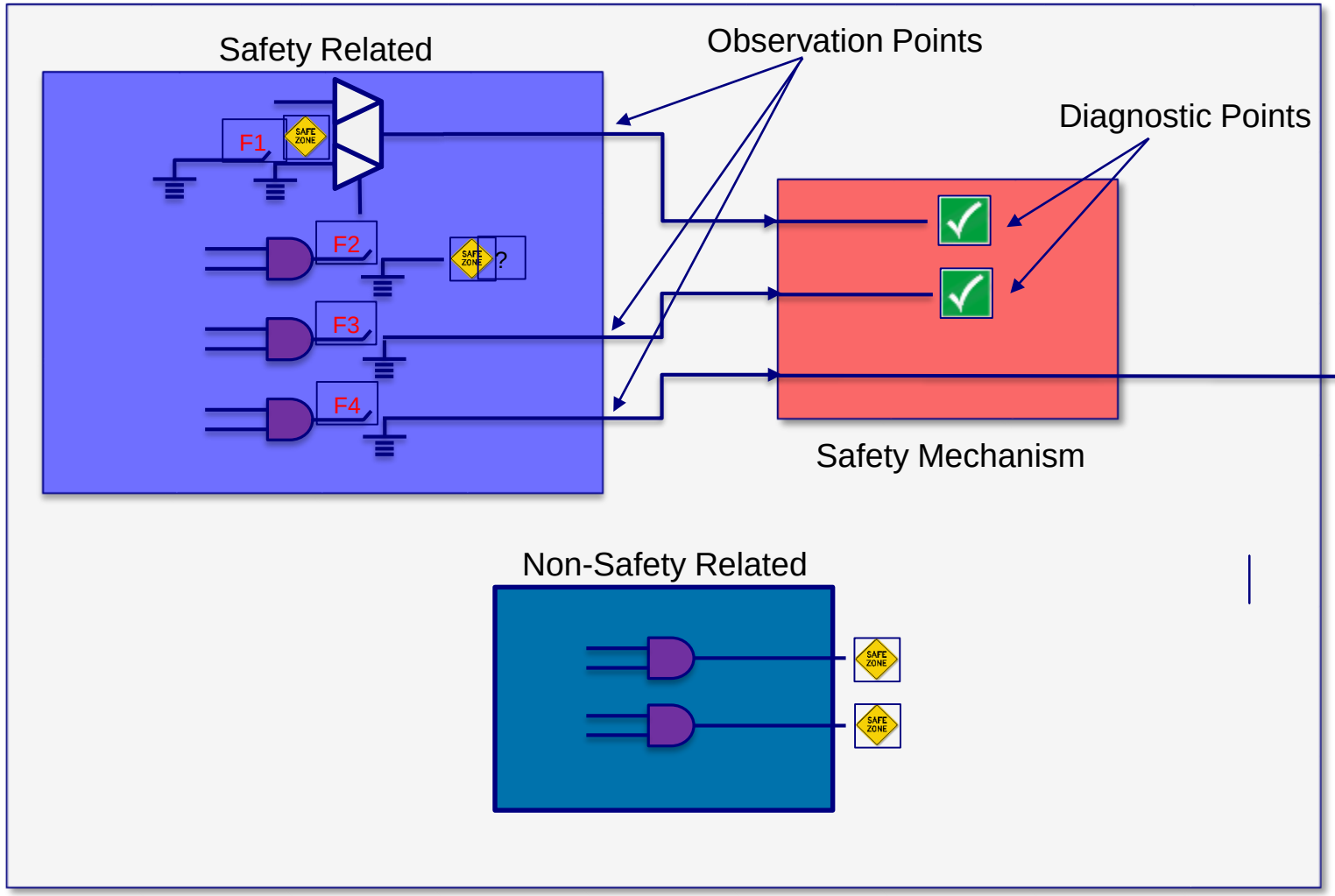
- SPFM

$$1 - \frac{\sum_{SR,HW} (\lambda_{SPF} + \lambda_{RF})}{\sum_{SR,HW} \lambda} = \frac{\sum_{SR,HW} (\lambda_{MPF} + \lambda_{S})}{\sum_{SR,HW} \lambda} \quad \text{(C.5)}$$

- LFM

$$1 - \frac{\sum_{SR,HW} (\lambda_{MPF,latent})}{\sum_{SR,HW} (\lambda - \lambda_{SPF} - \lambda_{RF})} = \frac{\sum_{SR,HW} (\lambda_{MPF,perceived\ or\ detected} + \lambda_{S})}{\sum_{SR,HW} (\lambda - \lambda_{SPF} - \lambda_{RF})} \quad \text{(C.6)}$$

Fault Classification Through Simulation



F1 – Safe
F2 – Assumed Dangerous
F3 – Dangerous Detected
F4 – Dangerous Undetected

If a fault was not observed and/or detected (F2), it can be:

1. A safe fault
2. A dangerous fault which did not propagate due to insufficient stimulus

Fault Injection Campaign

- Goal: determine Diagnostic Coverage of the SM by injecting faults in the design, and checking if they are detected by it
  - Fault simulators
    - Can use existing verification tests
    - Can run concurrently, handling many faults at a time
    - Stimulus may not be sufficient to cause all dangerous faults to propagate
  - Formal tools
    - Can determine which faults are uncontrollable from the inputs
    - Can check for Observation points Cone Of Influence (COI) – observability of faults

Fault Simulation Strategies

- At the beginning of the fault campaign – sample low percentage (e.g. 2%)
  - Check that your safety mechanism coverage matches expectations
- Full fault campaign –use Expert Judgement for sampling size
  - well-known Safety Mechanisms vs. “home grown” ones
    - E.g.: Covering a safety critical processor by creating a **lock-step** with a redundant copy of the processor is a well known SM in the industry
      - » In this case it may be enough to fault simulate 5-10% of the faults
    - Other SMs need 100% fault simulations

Add Observation (Strobe) Points

- When it comes to strobing, three things are important:
  - Location (where), Location (when), Location (what)!
  - Strobing affects not only how many faults will detected, it will affect performance at well.
  - Use $fs_strobe to add observation and diagnostic points

Syntax:

```
$fs_strobe(<list_of_hierarchical_signal_names>) or
$fs_strobe(<instance_path>)
```

Automatically strobes all outputs of a Verilog instance

Example:

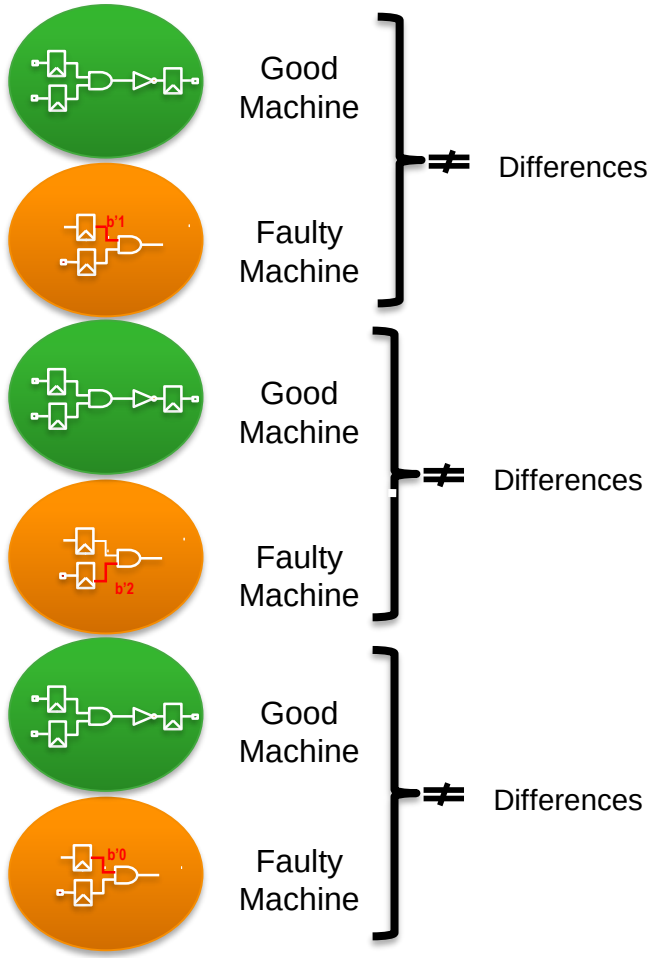
```
initial begin
wait (reset===1);
$display ("reset completed injecting faults now");
$fs_inject;
forever @(posedge testclk)
  if (faultSenseOn === 1'b1)
    #99 $fs_strobe(TPAD1, TPAD2, TPAD3);
end
```

Delaying fault injection until after reset typically gives higher test coverage due to more detected faults and fewer potential faults

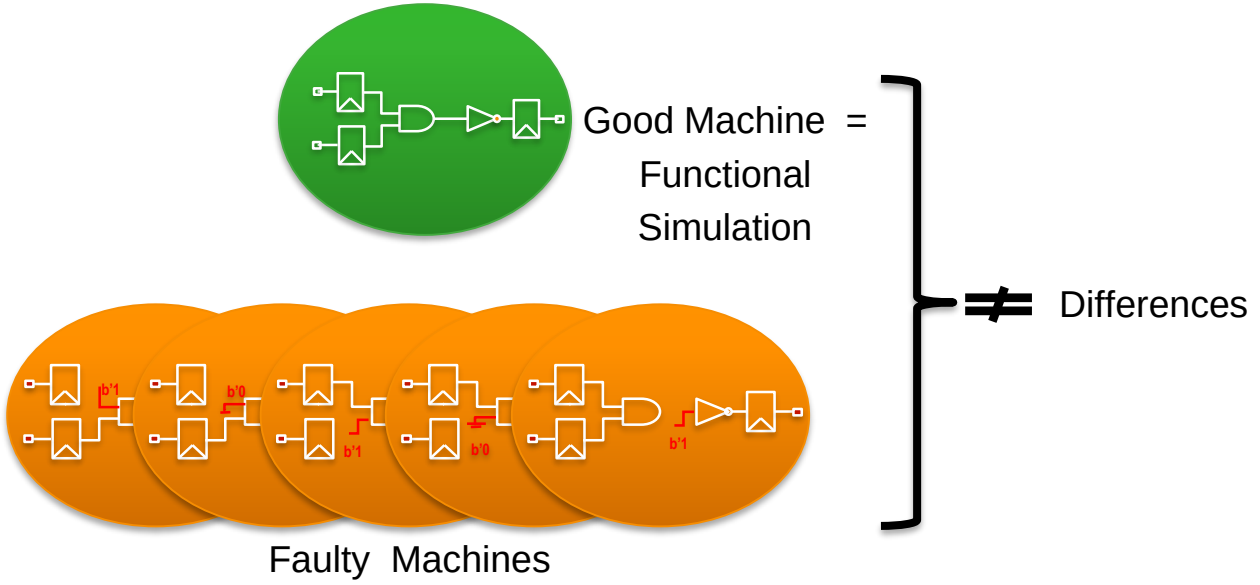
Generate Faults

- Many methods available to generate faults
  - Let tool generate faults
  - Import faults from 3rd party tools
  - Specify faults using a proprietary Standard Fault Format
- Advantage of using Standard Fault Format
  - Can specify user defined fault status
  - Can specify regions to generate faults and also regions to exclude
  - Extremely compact representation for transient faults
  - Can use wildcards
  - Can specify sampling methods during fault generation
  - Can specify user defined coverage metrics

Use Concurrent Fault Simulator



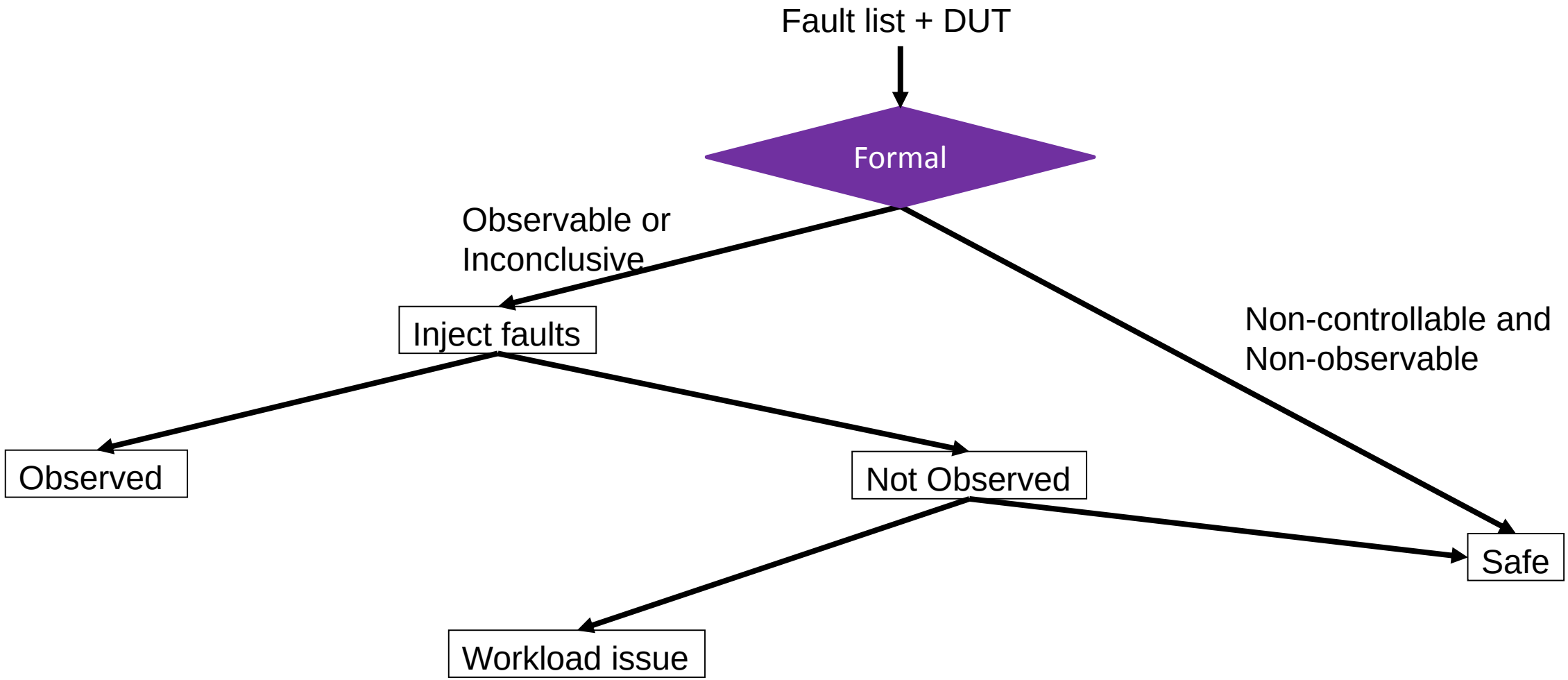
Legacy Parallel Simulation Technology



Thousands of faults in a single simulation
<u>Orders of magnitude</u> faster than parallel simulation

Concurrent Fault Simulation Technology

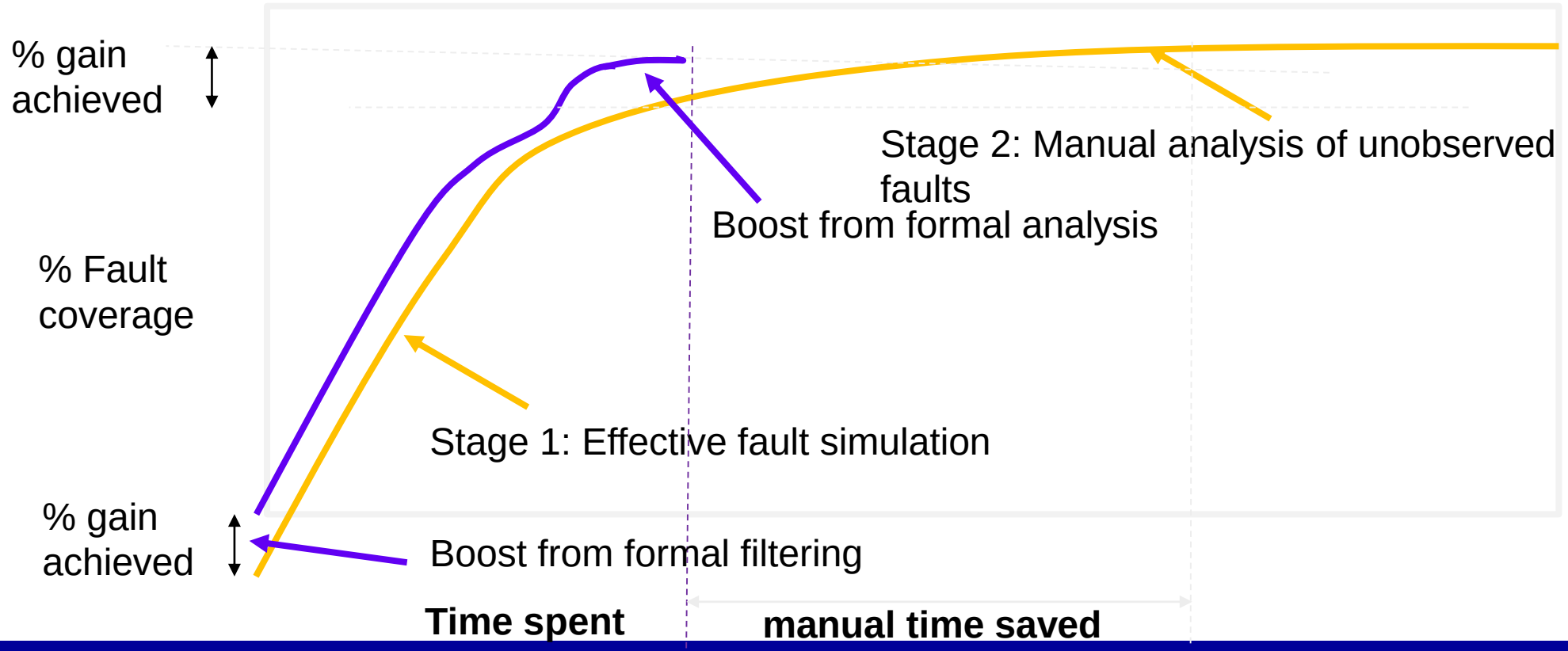
Use Formal Technology To Analyze Safe/Unsafe Faults



Benefits of Formal Fault Analysis

- Formal filtering of faults can provide a boost to fault coverage % by eliminating safe faults
- Formal analysis of unobserved faults can help in creating better stimulus

The fastest way to fault simulate is NOT to fault simulate …



2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

FMEDA calculation & Report

MAIN FMEDA

Unique ID	Top Design Element	Element-1	Element-2	Potential Faults	Potential Errors (as seen at the top design element boundary)	Potential Effects of Failure (visible to the system)	System level potential effect class	Safety Related?
HOST_FM_1	MEM_CTRL			corrupted CPU command	Memory content corruption	Wrong coding, wrong or no execution	CPU/GPU::Uni	Yes
HOST_FM_2	MEM_CTRL			corrupted CPU command	Memory content corruption	Wrong coding, wrong or no execution	CPU/GPU::Uni	Yes
HOST_FM_3	MEM_CTRL			corrupted CPU write data	Memory content corruption	Wrong coding, wrong or no execution	CPU/GPU::Uni	Yes
HOST_FM_4	MEM_CTRL			corrupted CPU write data	Memory content corruption	Wrong coding, wrong or no execution	CPU/GPU::Uni	Yes
HOST_FM_5	REG_UNIT			incorrect registers read	Memory content corruption	Processor architectural state/control corrupt	CPU/GPU::Uni	Yes
HOST_FM_6	REG_UNIT			incorrect registers read	Memory content corruption	Processor architectural state/control corrupt	CPU/GPU::Uni	Yes
HOST_FM_7	REG_UNIT			incorrect registers write	Memory content corruption	Processor architectural state/control corrupt	CPU/GPU::Uni	Yes
HOST_FM_8	REG_UNIT			incorrect registers write	Memory content corruption	Processor architectural state/control corrupt	CPU/GPU::Uni	Yes

Permanent Fault Model

Unique ID	D_{FMi}	$\lambda_{intrinsic}$	λ_{nSR}	λ_{SR}	F_{safe} Device	F_{safe} Application	F_{safe}	λ_S	λ_{nS}	F_{PVSG}	λ_{PVSG}	SoC built-in Diagnostic	built-in Diagnostic ID	SoC built-in Diagnostic $K_{FMC,RF}$	Application Diagnostic	Application Diagnostic ID	Application Diagnostic $K_{FMC,RF}$	$K_{FMC,RF}$	λ_{SPF}	λ_{RF}	$\lambda_{MPF,primary}$	$\lambda_{MPF,secondary}$	λ_{MPF}
HOST_FM_1	9.13%	5.81E+00	0.00E+00	5.81E+00	75%	0%	75%	4.36E+00	1.45E+00	41%	5.96E-01			30%			0%	30.0%	0.00E+00	4.17E-01	8.57E-01	1.79E-01	1.04E+00
HOST_FM_2																							
HOST_FM_3	3.91%	2.49E+00	0.00E+00	2.49E+00	96%	0%	96%	2.39E+00	9.96E-02	43%	4.28E-02	Host Safety 2	PSM_2	98%			0%	98.3%	0.00E+00	7.49E-04	5.68E-02	4.21E-02	9.89E-02
HOST_FM_4																							
HOST_FM_5	77.33%	4.92E+01	0.00E+00	4.92E+01	79%	0%	79%	3.89E+01	1.03E+01	16%	1.65E+00			30%			0%	30.0%	0.00E+00	1.16E+00	8.68E+00	4.96E-01	9.17E+00
HOST_FM_6																							
HOST_FM_7	9.63%	6.12E+00	0.00E+00	6.12E+00	68%	0%	68%	4.16E+00	1.96E+00	45%	8.82E-01	2,Host Safety	PSM_4	70%			0%	70.0%	0.00E+00	2.64E-01	1.08E+00	6.17E-01	1.69E+00
HOST_FM_8																							

Transient Fault Model

Unique ID	$\lambda_{MPF,p}$	D_{FMi}	$\lambda_{intrinsic}$	λ_{nSR}	λ_{SR}	F_{safe} Device	F_{safe} Application	F_{safe}	λ_S	λ_{nS}	F_{PVSG}	λ_{PVSG}	SoC built-in Diagnostic	SoC built-in Diagnostic ID	SoC built-in Diagnostic $K_{FMC,RF}$	Application Diagnostic	Application Diagnostic ID	Application Diagnostic $K_{FMC,RF}$	$K_{FMC,RF}$	λ_{SPF}	λ_{RF}
HOST_FM_1	8.94E-01																				
HOST_FM_2		9.13%	3.88E-01	0.000	0.388	32%	0%	32%	1.24E-01	2.64E-01	92%	2.43E-01	Host Safety 2	_2	98%			0%	97.8%	0.000	0.005
HOST_FM_3	4.64E-05																				
HOST_FM_4		3.91%	1.66E-01	0.000	0.166	57%	0%	57%	9.48E-02	7.15E-02	15%	1.07E-02	Host Safety 4	_4	97%			0%	97.4%	0.000	0.000
HOST_FM_5	7.02E+00																				
HOST_FM_6		77.33%	3.29E+00	0.000	3.287	73%	0%	73%	2.40E+00	8.87E-01	82%	7.28E-01	Host Safety 4	_4	97%			0%	96.7%	0.000	0.024
HOST_FM_7	1.69E-01																				
HOST_FM_8		9.63%	4.09E-01	0.000	0.409	61%	0%	61%	2.50E-01	1.60E-01	45%	7.18E-02	Host Safety 4	_4	90%			0%	90.0%	0.000	0.007

ISO 26262 Metric report

- Probabilistic Metric for random Hardware Failures (PMHF)
- Single-point fault metric (SPFM)
- Latent-fault metric (LFM)

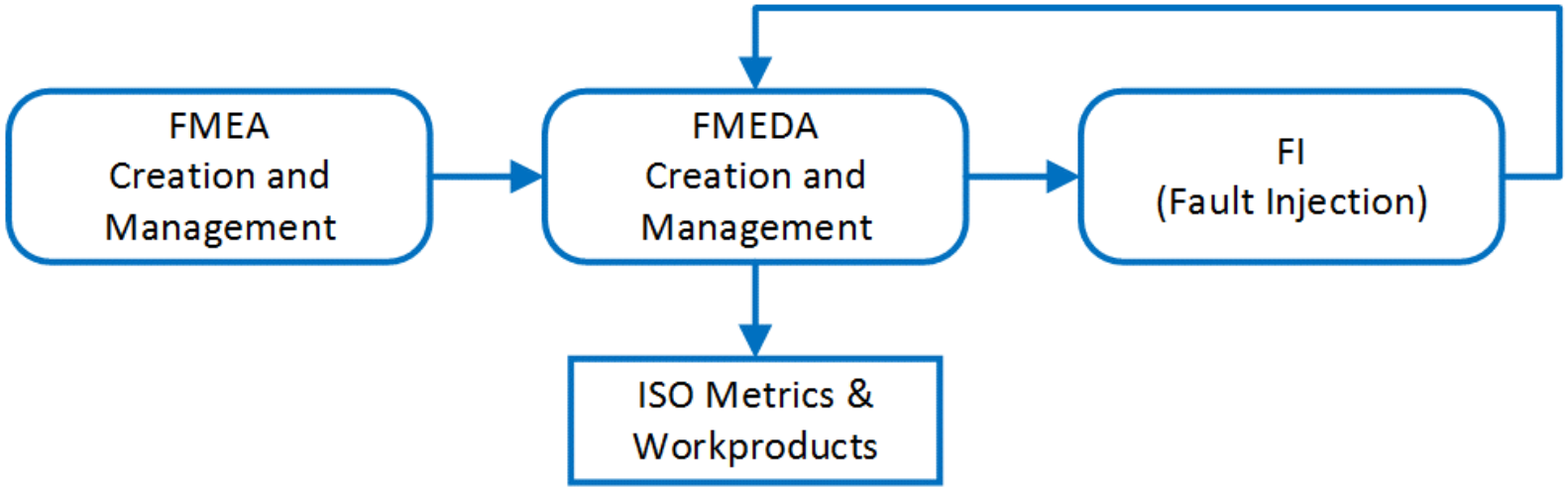
D	E	F	G	H
METRICS DASHBOARD				
	Permanent	Transient	Total	
PMHF (Failures per 10^9 hours	1.84E+00	3.69E-02	1.88E+00	
SPFM	97.1%	99.1%	97.2%	
	Permanent			
LFM	88.8%			
Part	(P&T combined)			
HOST	0.972356059			>= 90%
				< 90%

2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

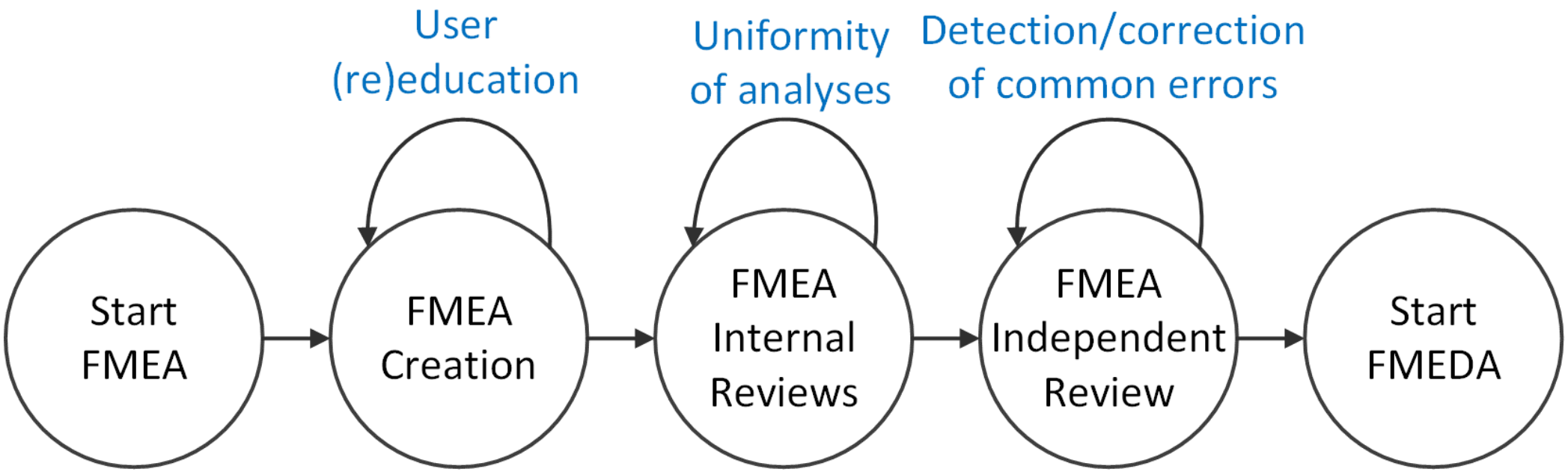
NVIDIA ISO 26262 Methodology Case Study

NVIDIA Case Study

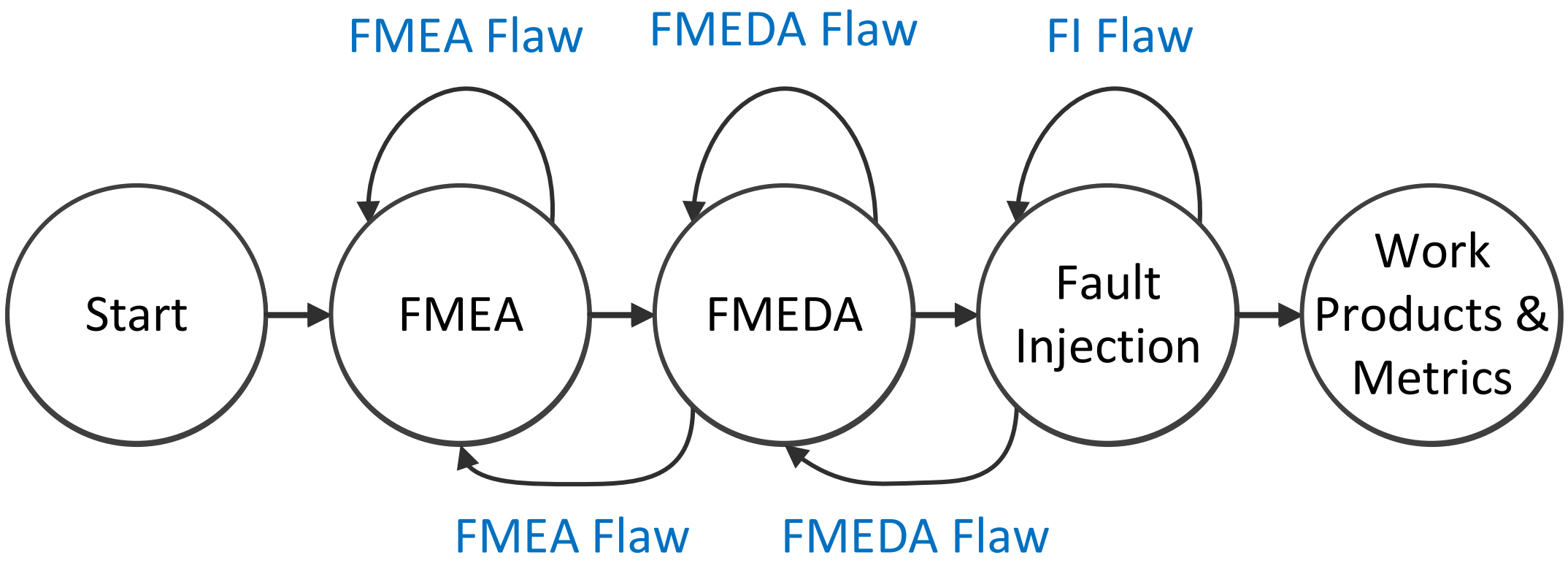
- Focus on FMEA to Metrics process for HW
- Big Picture
- FMEA Challenges
- FI Challenges
- Mindset Challenges
- Conclusions



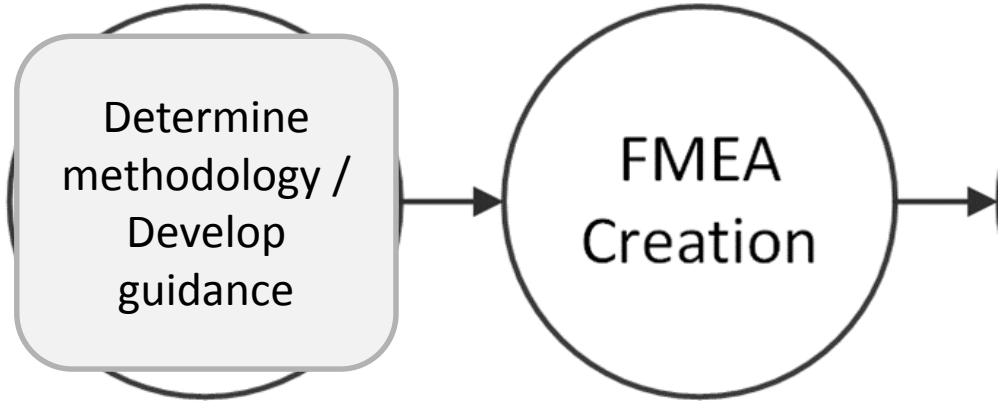
Stating the Obvious : Speed Matters



Scaling the Obvious



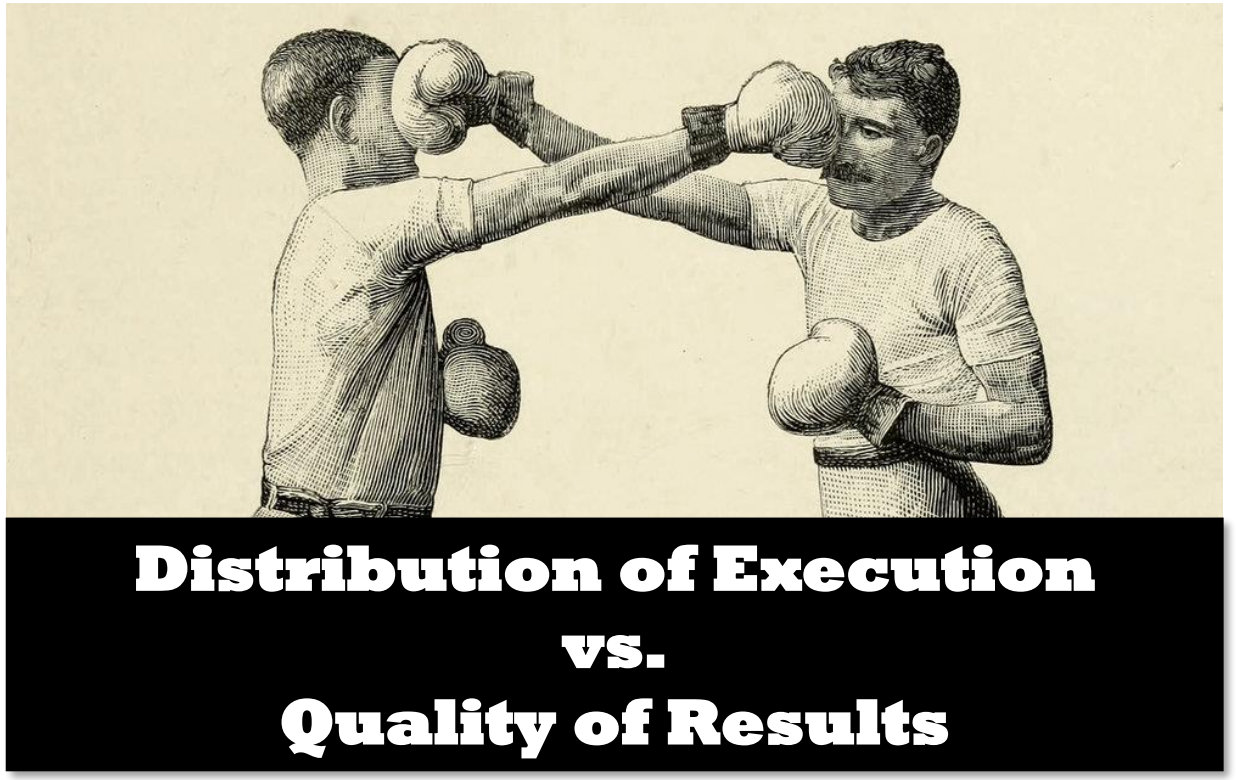
Nobody Wins a Marathon in the 1st Mile



- How to interpret and apply ISO 26262?
- How to communicate that guidance?

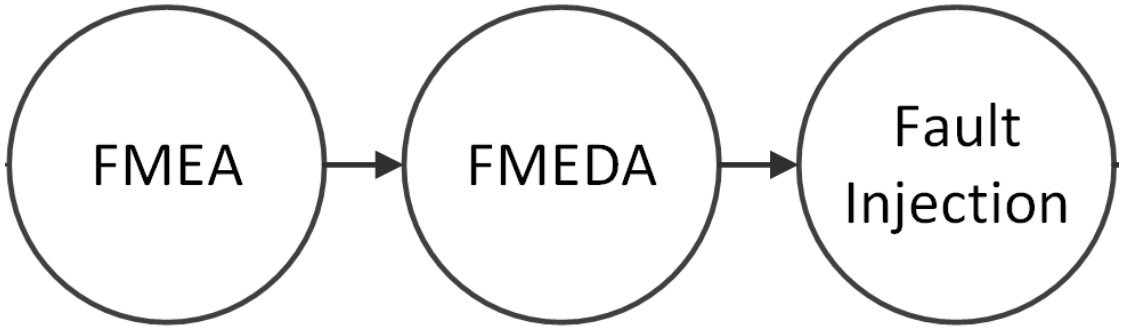
FME(D)A: Distribution vs. Quality

- Distribution of execution
- Quality of results



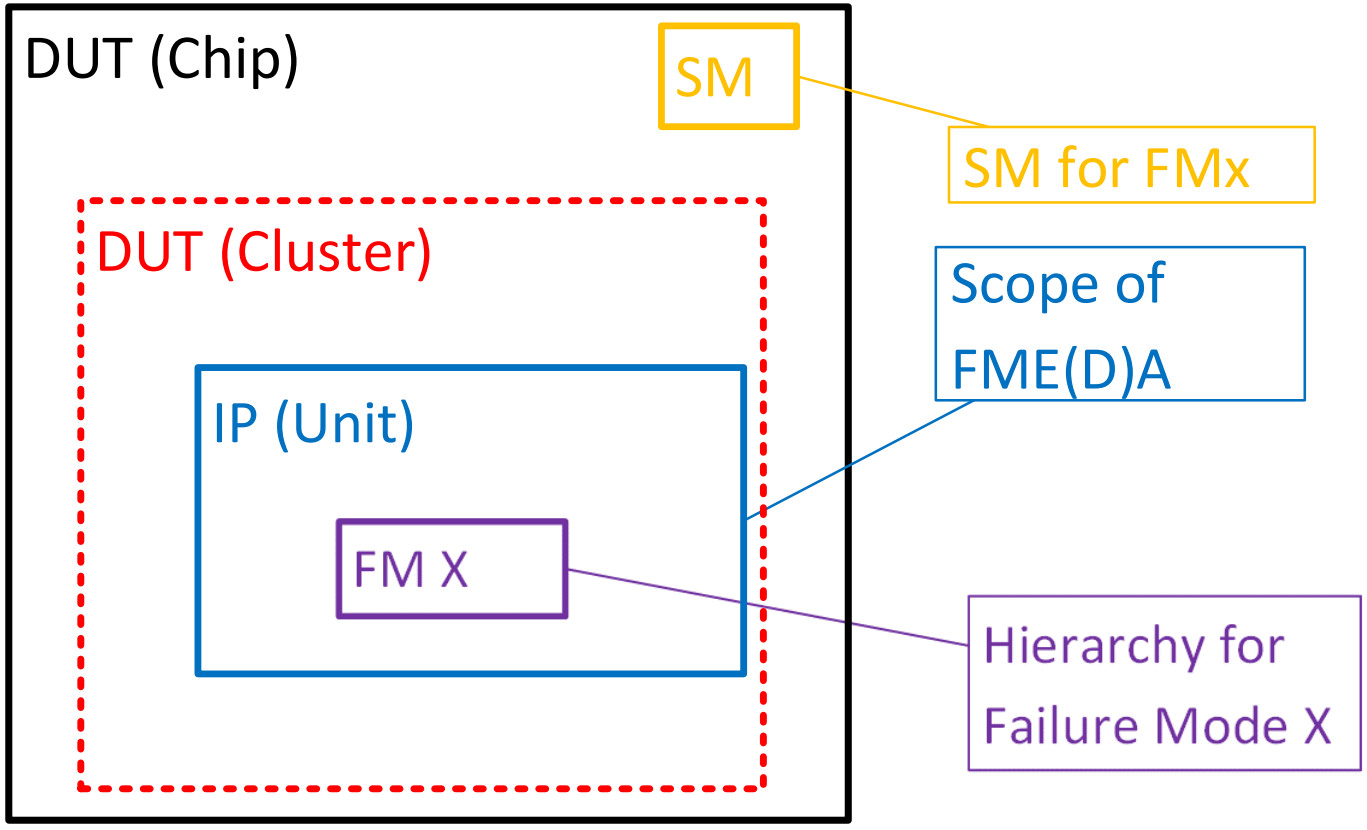
FMEA Execution Issues

- FMEA template format
- Scope of an individual FMEA
- Granularity of analysis within an FMEA
- Uniform application of the standard
- FMEA is just the start



Z01X is a Tool, How Will You Use It?

- What IPs, FMs?
- DUT selection
  - Where does the FM live?
  - Available DUTs?
  - Where does SM live?



FI : No Shortage of Questions

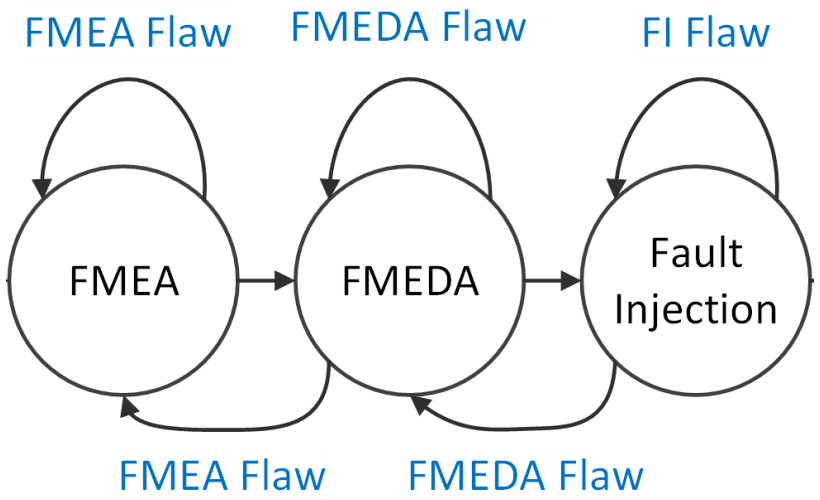
- Workload selection
- RTL vs Gates
  - Transients can reasonably use RTL or Gates
  - Permanents need Gates
- DUT, Workload, RTL vs. Gates interact

FuSa Requires Mindset Change

- DV is used to thinking about systematics (“bugs”)
  - DV: Assume functionality is buggy, expose the bugs
  - FI for FMEDA: Assume functionally correct, measure efficacy of SM
- Arch, design are not used to thinking about random faults

Conclusions

- Specialized tools are necessary
  - 100 Excels will not suffice
- FuSa methodology must be carefully defined
- FuSa methodology != DV methodology
- Phase rollout to avoid churn
  - Single pilot
  - 1 pilot per category/type of IP
  - Full rollout



2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Thank You!