Functional Safety Verification for ISO 26262-Compliant Automotive Designs

JM Forey FSCP TÜV SÜD FSProfessional Automotive ID. TX13050139 www.tuv-sud.com - Synopsys

Werner Kerscher - Synopsys

SYNOPSYS®

Silicon to Software™

accellera SYSTEMS INITIATIVE

2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Agenda

- Emergence of self-driving cars
- ISO 26262 Primer for Semiconductor
- ISO 26262 Requirements – Hardware Development
- Functional Safety Verification Flow, from FMEA to FMEDA

EMERGENCE OF THE SELF DRIVING CAR

Autonomous Vehicles Are Taking Over The World

September 12, 2017 | Ann Arbor, Michigan
TRANSPORTATION SECRETARY ELAINE L. CHAO A
TECHNOLOGY, EMPHASIZES SAFETY BENEFITS AN
Ann Arb
today rel
guidance
"The nev

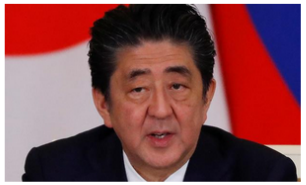
Décret n° 2018-211 du 28 mars
https://www.legifrance.gouv.fr/eli/decret/2018/3/28/TRER1717809D/jo/texte
Legifrance.gouv.fr
LE SERVICE PUBLIC DE LA DIFFUSION DU DROIT
jeudi 7 juin 2018
Informations de mise à jour
Accueil | Droit français | Droit européen | Droit international | Traductions | Bases de données
Décret n° 2018-211 du 28 mars 2018 relatif à l'expérimentation de véhicules à délégation de conduite sur les voies publiques
Imprimer

FINAN

Heise-Foren: Anmelden | Registrieren
heise online
IT Mobiles Entertai
TOPTHEMEN: DSGVO COMPUTEX 2
heise online > News > 10/2016 > Autonom
Autonomes Fahren
Millionen Euro für T
18.10.2016 13:50 Uhr – Andreas Wilkens

SPIEGEL ONLINE SPIEGEL+
Anmelden
Menü | Politik Meinung Wirtschaft Panorama Sport Kultur Netzwelt Wissenschaft mehr
MOBILITÄT
Schlagzeilen | Wetter | DAX 12.717,18 | TV-Programm | Abo
Nachrichten > Mobilität > Aktuell > Autonomes Fahren > Karlsruhe: Testfeld für autonomes Fahren eröffnet
Autonomes Fahren
Karlsruhe wird Testgebiet für Roboter-Autos
Baden-Württemberg will zum Vorreiter beim autonomen Fahren werden. Deshalb dürfen Robo-Autos jetzt auf ausgewählten Straßen rund um Karlsruhe fahren.

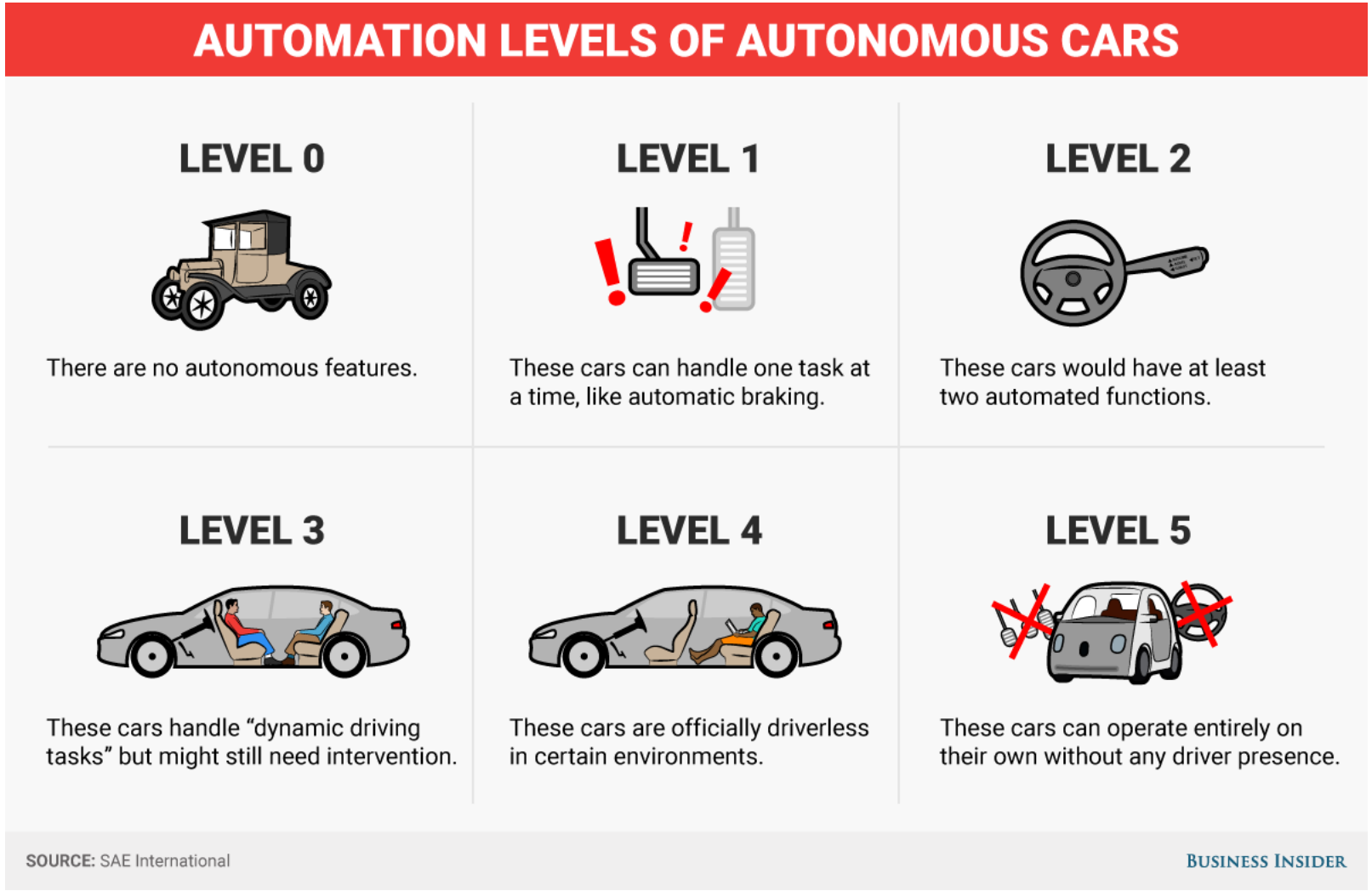
Member sign
TH
News
HOME CARS & CONCEPT
You are here: Home » Mobi
Japan l
system
June 4, 2018 @ 6:33 am
Stanley White
Reuters



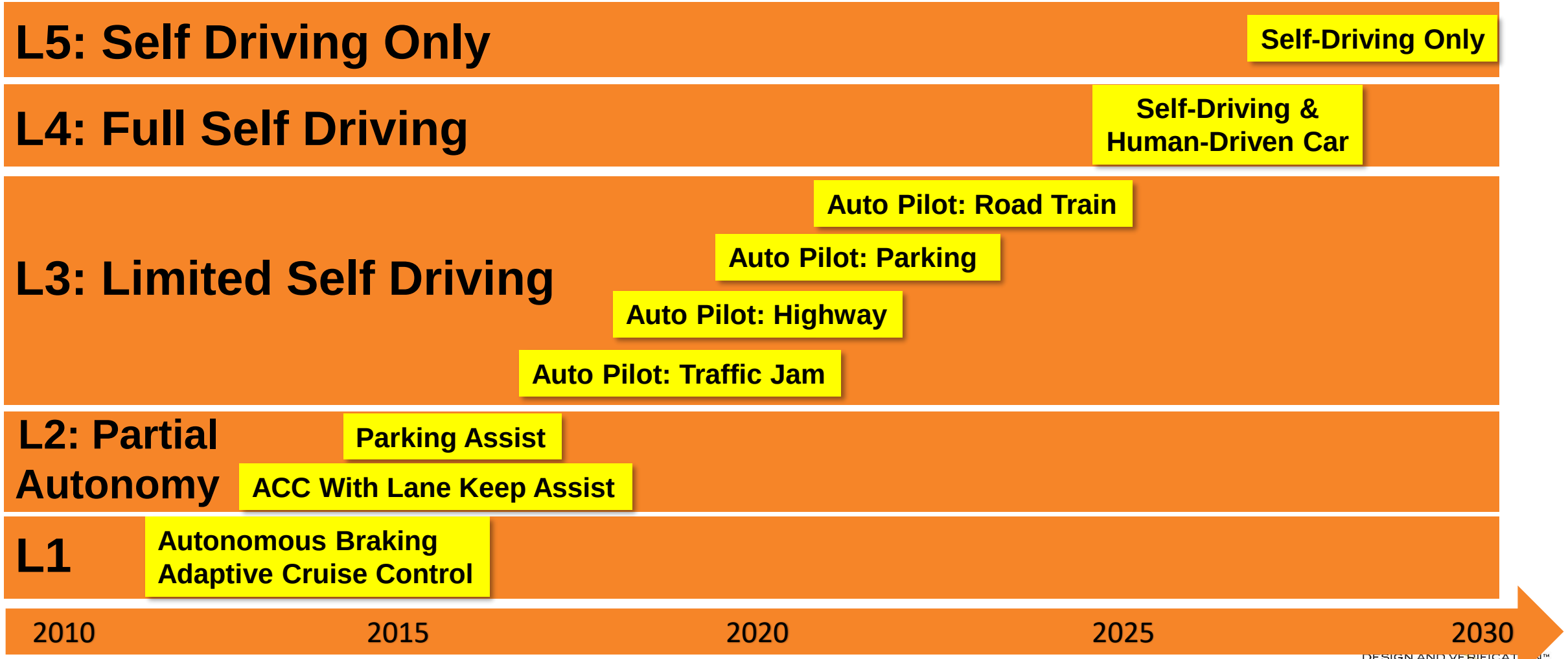
Prime Minister Shinzo Abe presented a plan that includes testing an autonomous car system on public roads sometime this fiscal year with the

The strategy, presented at a meeting chaired by

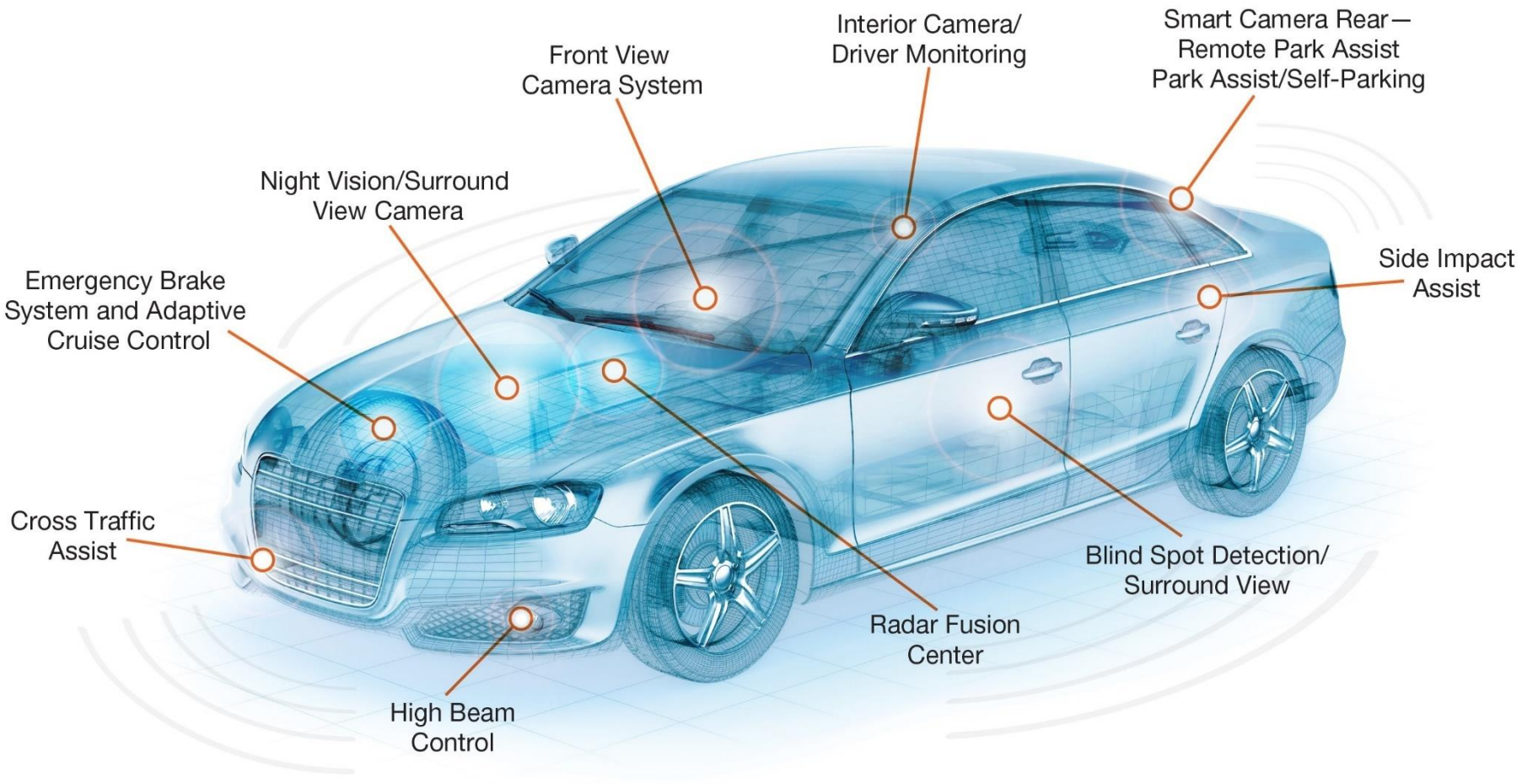
Levels of Automation in Cars



Roadmap of Autonomous Cars



Complex SOCs For ADAS



Enabling Safe, Secure, Smarter Cars

...from Silicon to Software

Software cybersecurity & quality

Verify functional safety (ISO 26262)

Automotive-certified IP

High-reliability IC design

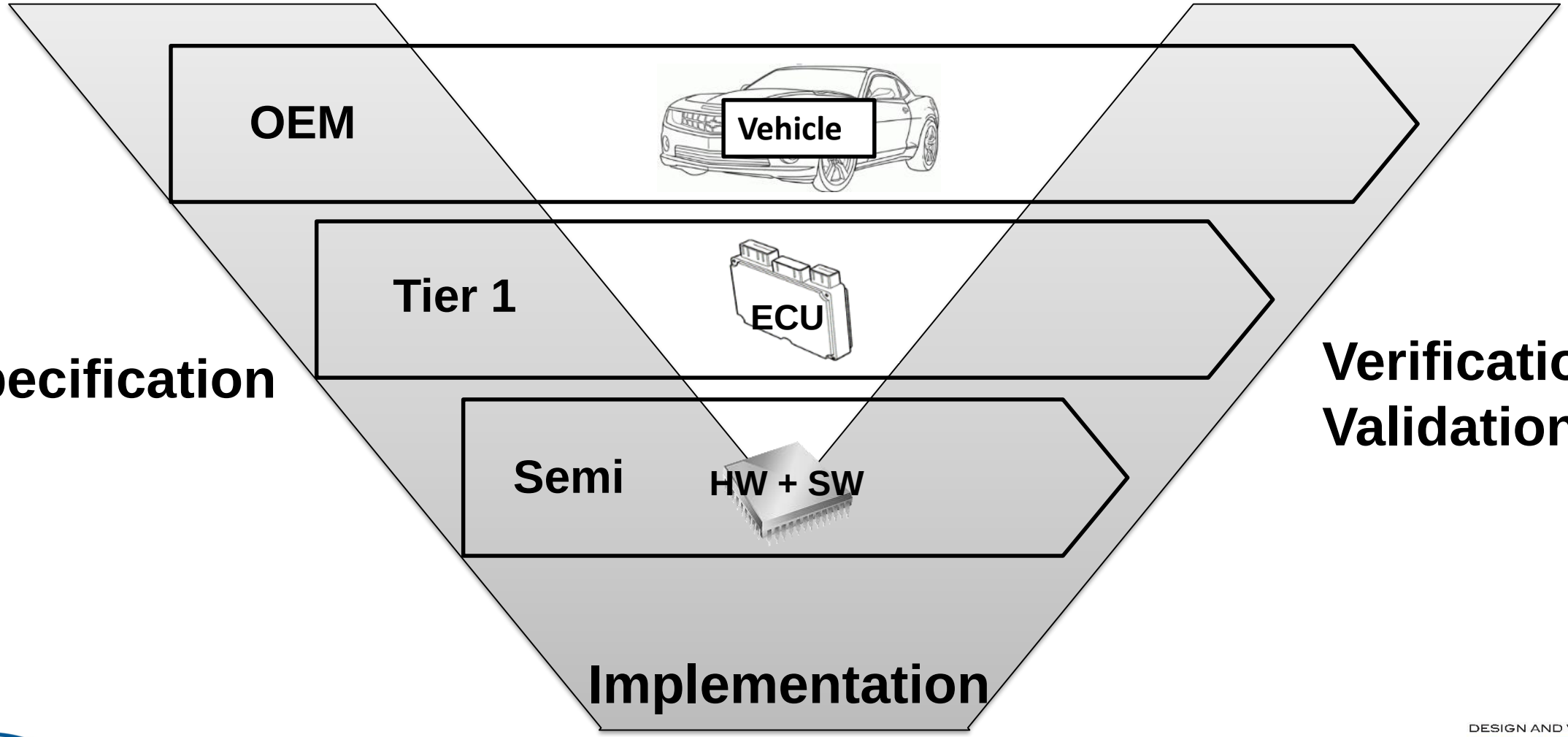
ISO 26262-certified Test

ISO 26262 PRIMER FOR SEMICONDUCTOR

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)

Safety Goals/Requirements

- Done at OEM / Tier 1 level
- 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.

Hazard Analysis and Risk Assessment (HARA)

- Determines the Automotive Safety Integrity Level (ASIL)

Probability of Exposure + Controllability by Driver + Severity of Failure = ASIL (A B C D)

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

A Safety Element out of context (SEooC) is a safety-related element which is not developed for a specific item. This means it is not developed in the context of a particular system or vehicle.

See ISO 26262 Part 10 "Guideline on ISO 26262", Chapter 9 "Safety element out of context"

Chips and IPs are normally 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

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.

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.

- **Safety** is a mind set
- What can go wrong?
  - At any level, notably
    - Conception: hw, sw
    - Verification
    - Manufacturing
    - In operation, in a permanent or transient way
    - …

- And of course
  - Measuring, addressing, minimizing impact, documenting, …
- Faults are either
  - Systematic
  - Random

Found/covered by Functional Verification tools

Assessed by Functional **Safety** Verification tools

ISO 26262 Requirements – Hardware Development

Show that design functionality is correct, works properly in the context of the system, and is safe

Demonstrate and document that design and verification flows are robust

- Implementation tools and flows <u>do not introduce</u> design bugs (systematic faults)
- Functional verification tools and flows <u>do not fail to report</u> design bugs

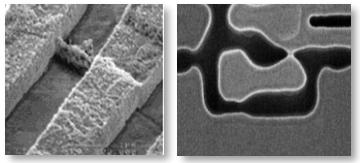
Systematic Faults

Always permanent

Reduced DPPM

- DFT
- Functional patterns

Random Faults

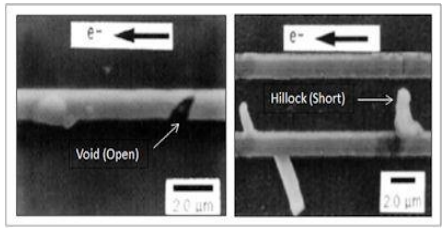


Permanent

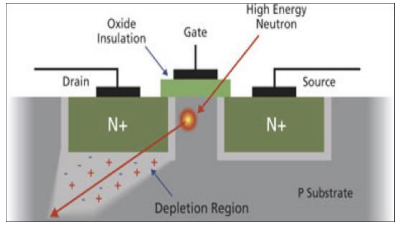
Demonstrate and document that safety mechanisms operate properly

- Safety mechanisms triggered in presence of faulty behavior, and not otherwise
- Safety mechanisms are effective in reaching a safe design state

Random Faults



Permanent



Transient

Development → **Manufacturing** → **In Operation**

Lifecycle of Component / System / Automobile

ISO 26262 Requirements – Hardware Development

Show that design functionality is correct, works properly in the context of the system, and is safe

Demonstrate and document that design and verification flows are robust

- Implementation tools and flows do not introduce design bugs (systematic faults)
- Functional verification tools and flows do not fail to report design bugs

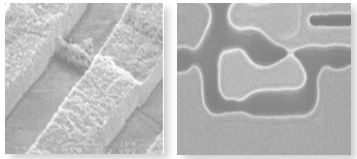
Systematic Faults

Always permanent

Reduced DPPM

- DFT
- Functional patterns

Random Faults

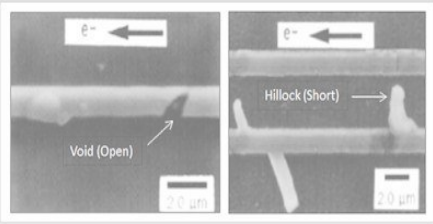


Permanent

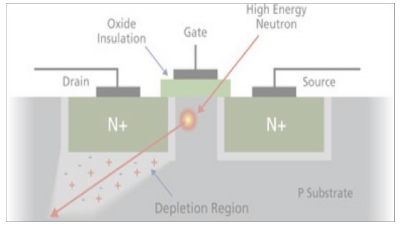
Demonstrate and document that safety mechanisms operate properly

- Safety mechanisms triggered in presence of faulty behavior, and not otherwise
- Safety mechanisms are effective in reaching a safe design state

Random Faults



Permanent



Transient

Development → **Manufacturing** → **In Operation**

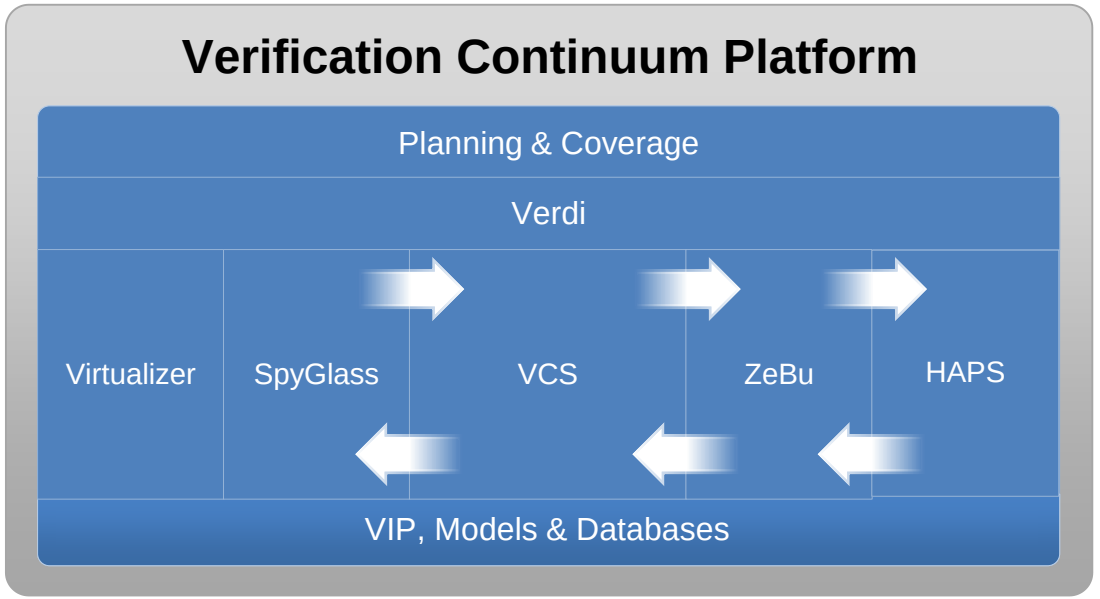
Lifecycle of Component / System / Automobile

Functional Verification is Essential Starting Point

Prevent / Eliminate Bugs

Avoid Systematic Faults – Design Bugs (Permanent Faults)

Verification & Validation:
Use State of the Art Functional Verification methodology

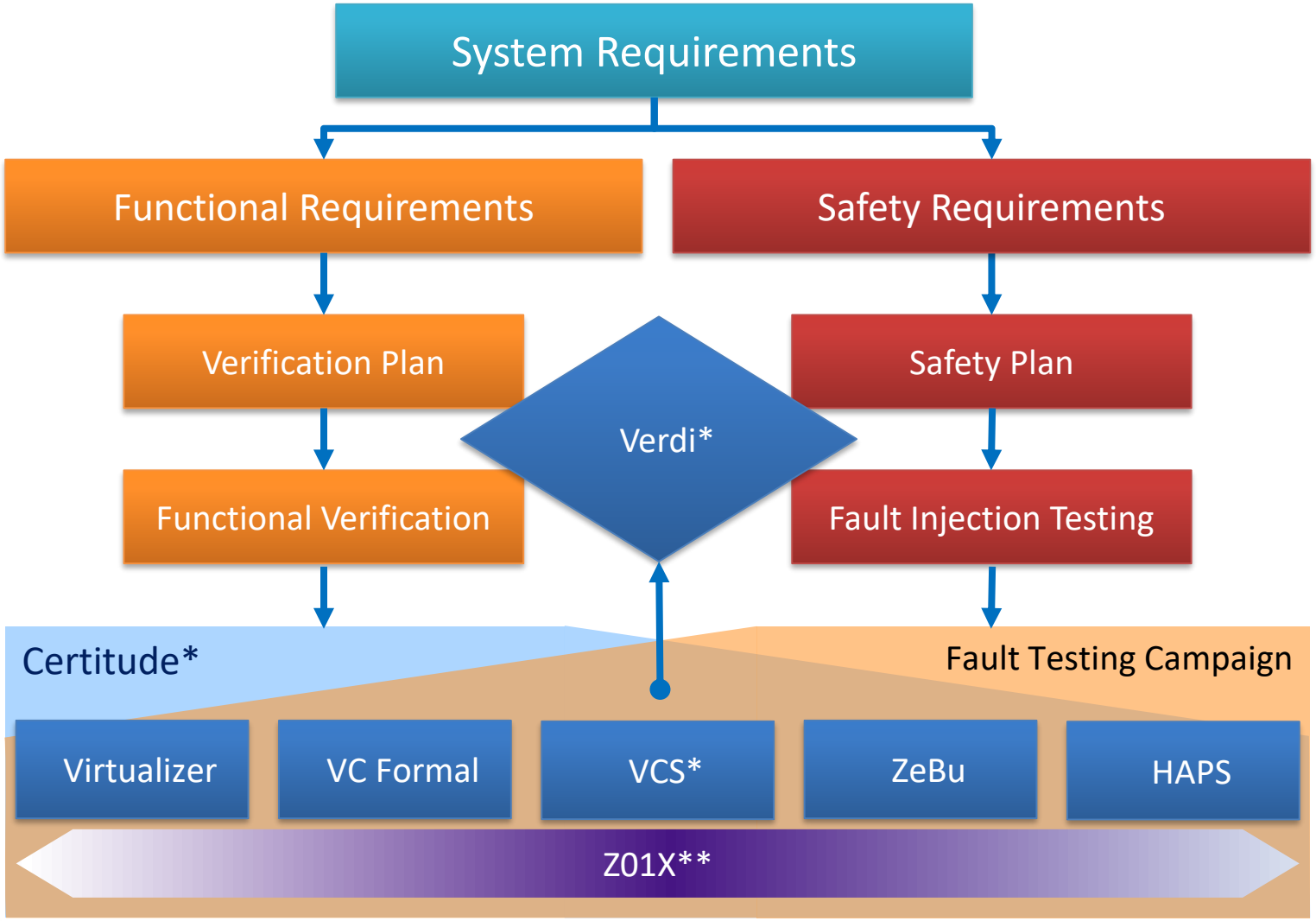


Synopsys Functional Verification Technology Platform

- 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 for IPs and SoCs

Executive Overview



Accelerate fault simulation campaign

- Most comprehensive solution for systematic and random faults testing
- Fastest simulation engines

Integrated with ISO 26262 flows

- Failure mode effects analysis
- Safety plan traceability and results

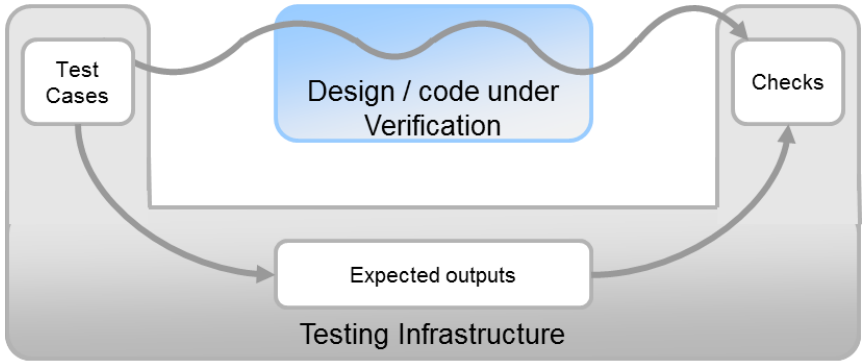
Proven

- TuV Certified Tools
- Deployed for Synopsys Certified IP development
- Adopted by market leaders

Functional Verification Qualification

In ISO 26262 context

- ISO 26262 part 8 Clause 11.2: *"Risk of **systematic faults** [...] is **minimized**"*
- Problem: test Infrastructures deliver pass/fail status
  - Do not directly address whether designs have bugs or not



		Design actually has bug(s)	
		Yes	**No**
Verification reports bugs	**No** (all tests Passed)	**False Negative**	Ok (done)
	Yes (some Failures)	Ok (debug design)	False Positive

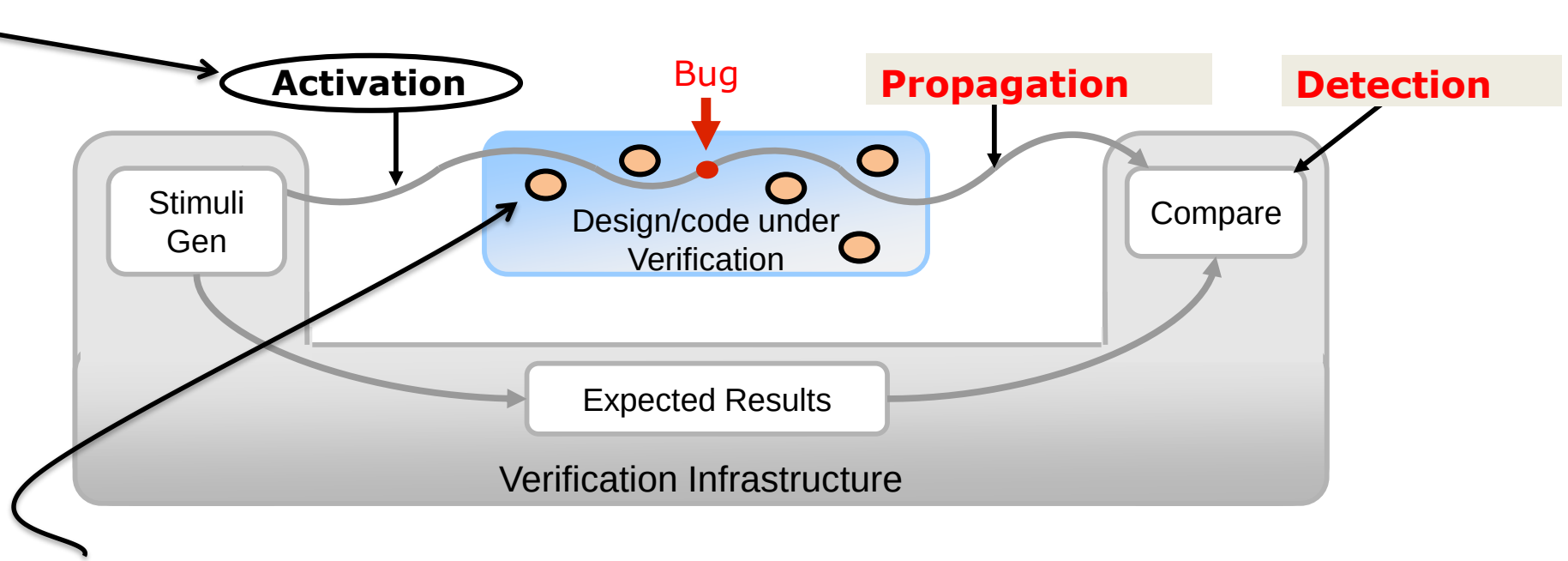
- Reported failures are debugged (good) => there is always something to fix
- **BUT** False Negative are silent
  - Are there any? Where are they?
  - Traditional methods can't help here

Is your verification tool failing to report functional bugs?

Assessing Verification Effectiveness

Traditional methods

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

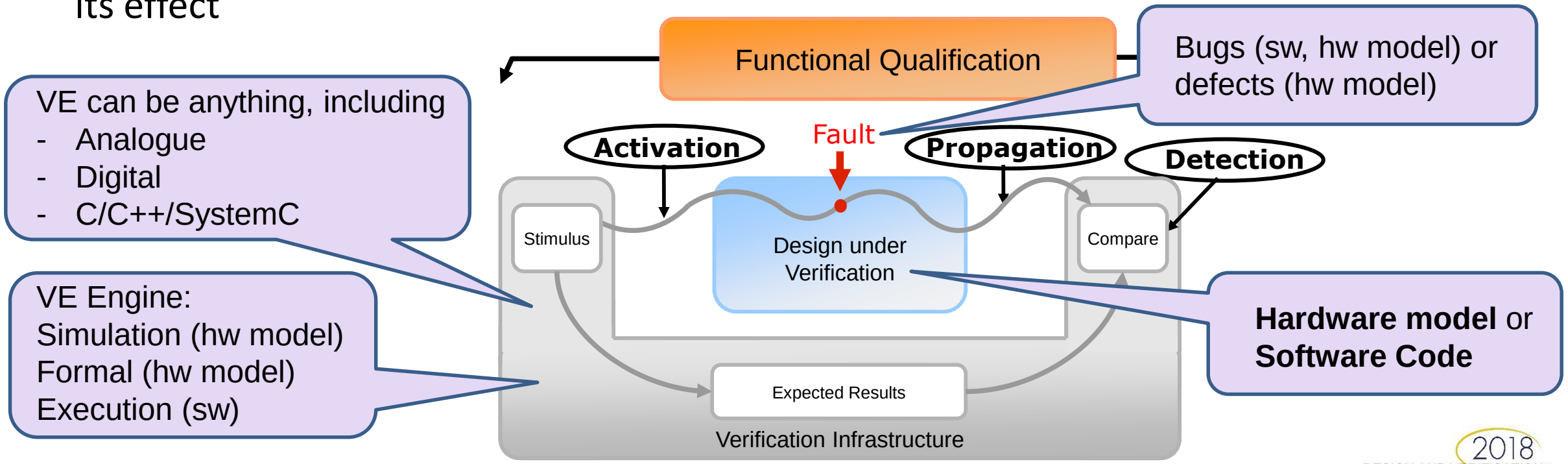


Functional coverage *checks "important" functional points, however comprehensiveness of functional points is unknown*

- Code/functional coverage are used to assess verification effectiveness
  - BUT they deliver a very partial picture

Assessing Verification Effectiveness

- Mutation testing applies universally in verification
- **Automatically inserts** “artificial bugs” into the design
- Runs verification process on “broken” design
- Measures the ability of the environment to exercise the fault, propagate and detect its effect



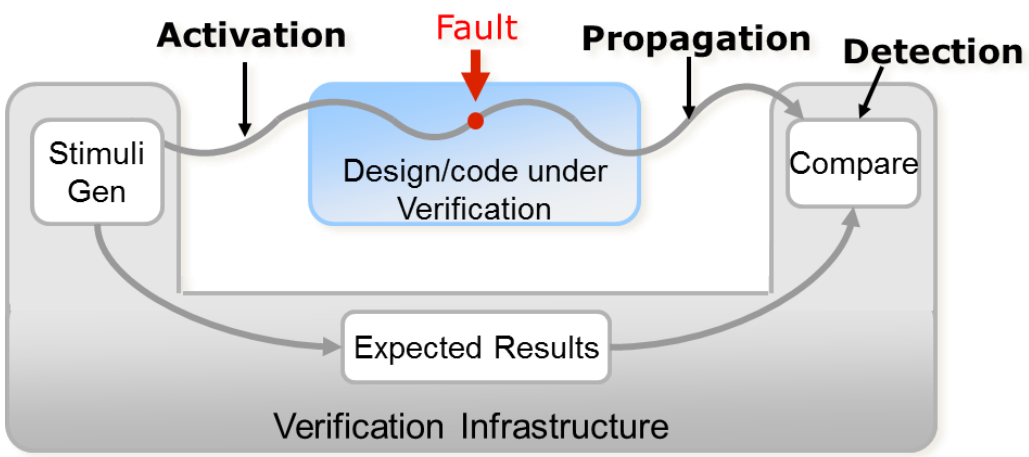
How Certitude Fault Injection Works?

- **Modifies design code to insert faults/defects**

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

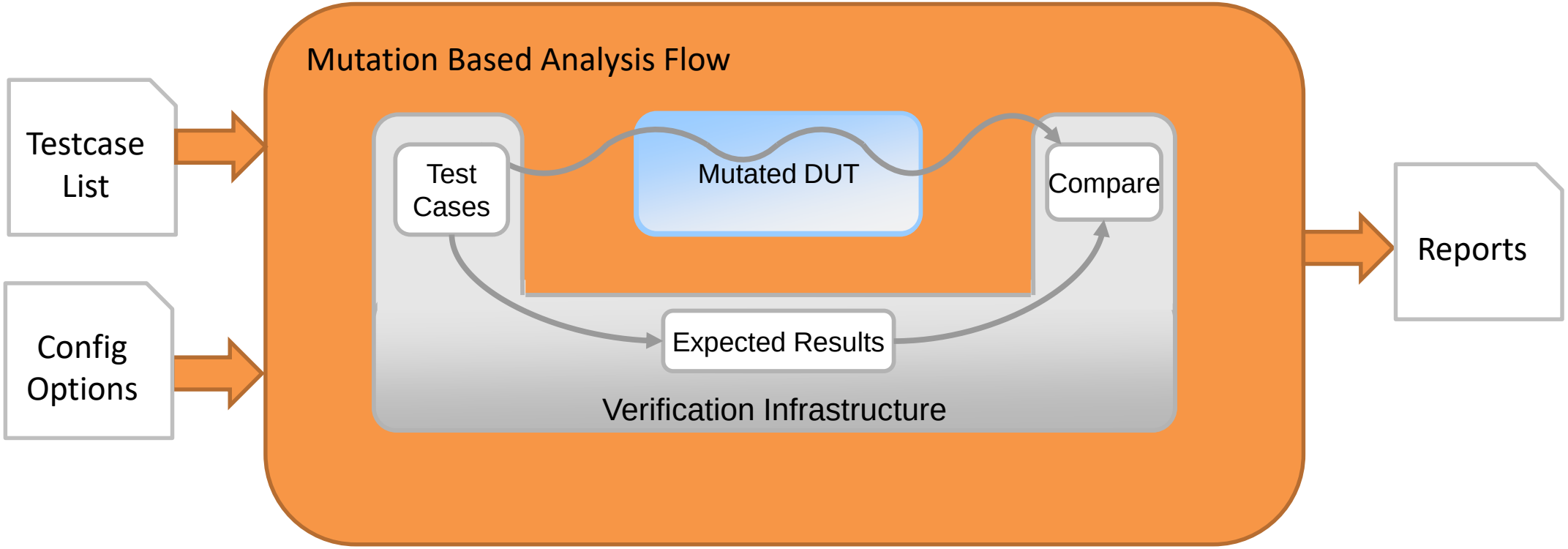
```
if (a)        →  if (TRUE)  // force execution
   f1();          f1();      // of "if" branch
else              else
   f2();          f2();
```

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

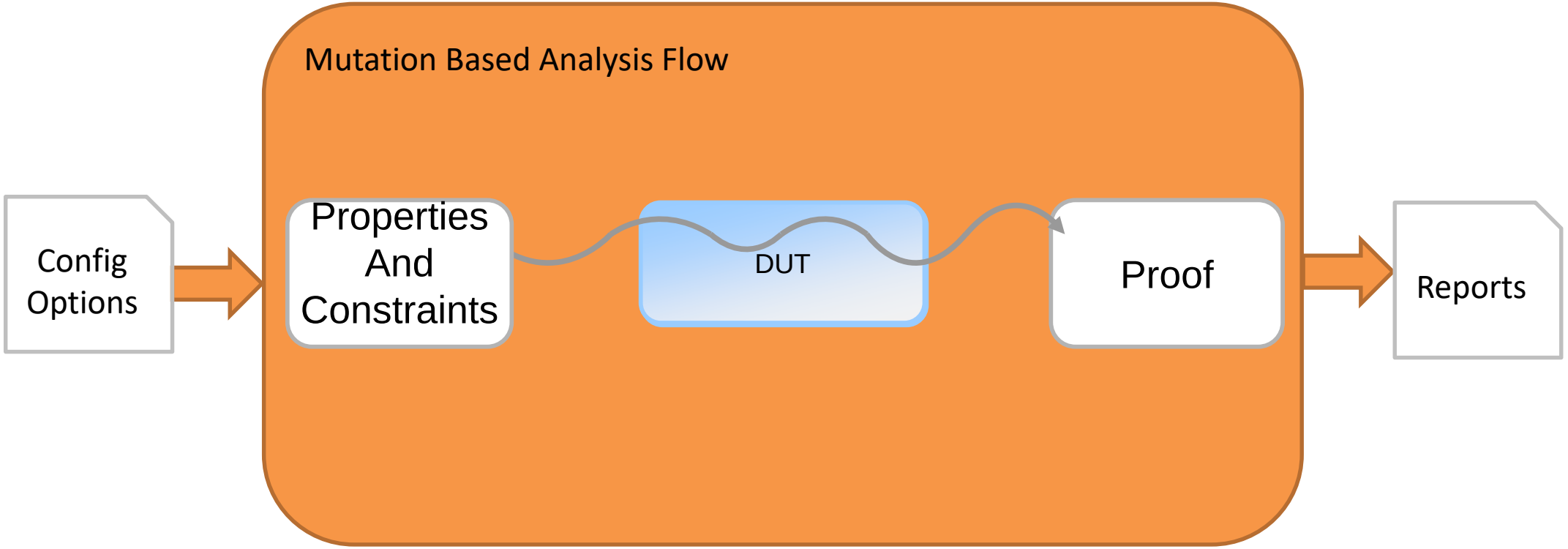


- **Pass the broken design to the verification**
  - Does at least one test fail? ***Great!***
    - Environment/Safety Mechanisms robust enough to detect the design is broken
  - Do all tests pass? ***Help!*** ⇒ **False Negative result => <u>VE is hiding bugs</u>**
    - Original and broken design looks ok for the verification

Easy Integration Within Existing Environments

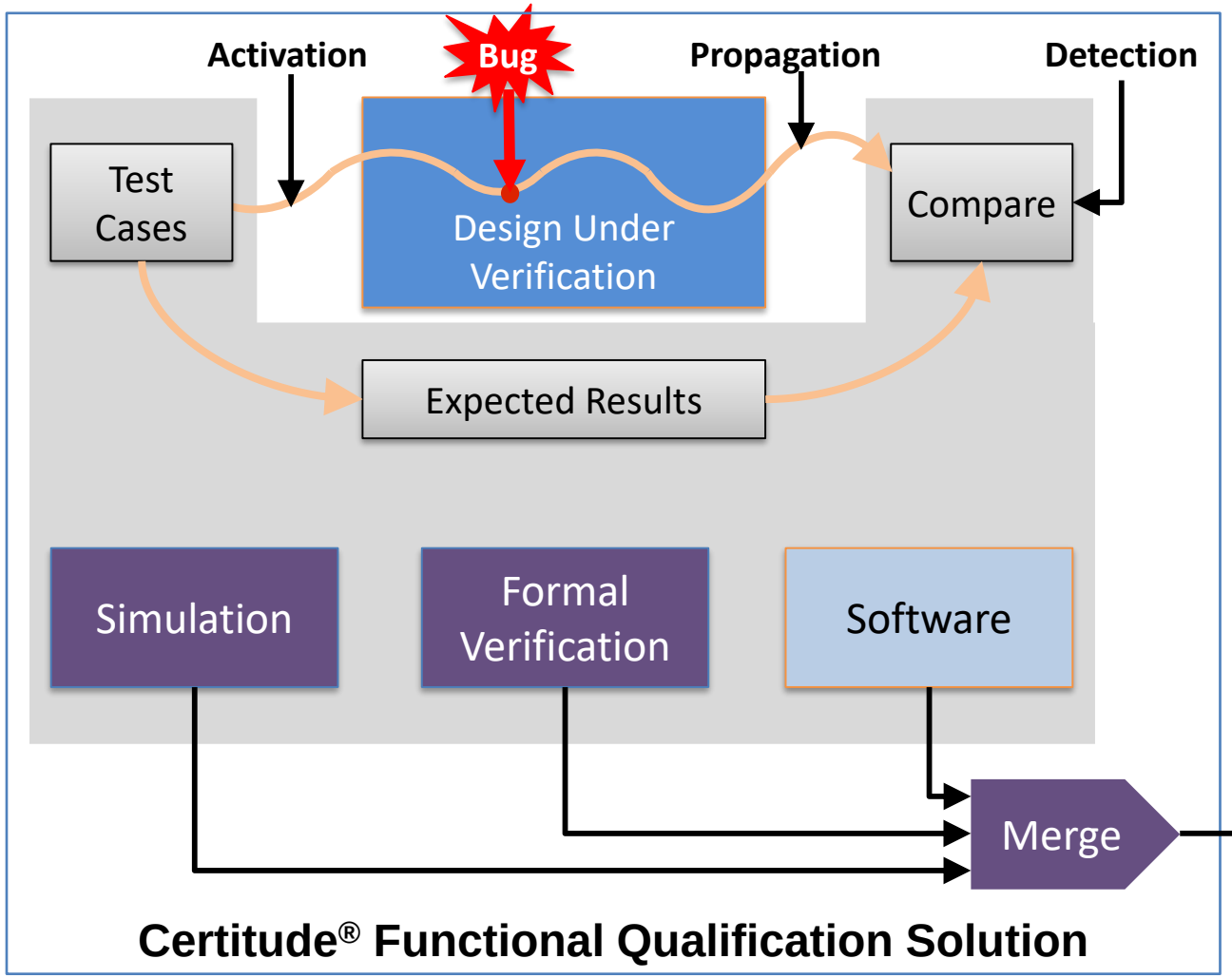


Formal Verification



Demonstrate Verification Flows are Robust

Evidence-based verification quality analysis for ISO 26262 Part 8-9 assessments



Inject and qualify systematic faults at architecture, system, and RT level

Measure verification completeness and functional correctness of design

Natively integrated with VCS and VC Formal, and works with C/C++/SystemC flows

Unified functional verification environment quality metrics

ISO 26262 Requirements – Hardware Development

Show that design functionality is correct, works properly in the context of the system, and is safe

Demonstrate and document that design and verification flows are robust

- Implementation tools and flows do not introduce design bugs (systematic faults)
- Functional verification tools and flows do not fail to report design bugs

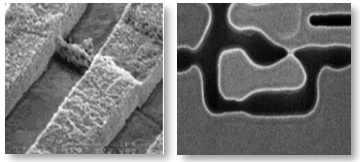
Systematic Faults

Always permanent

Reduced DPPM

- DFT
- Functional patterns

Random Faults

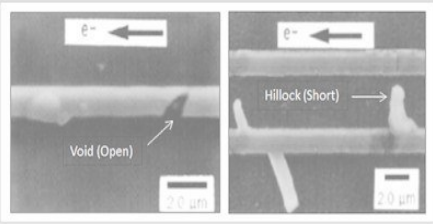


Permanent

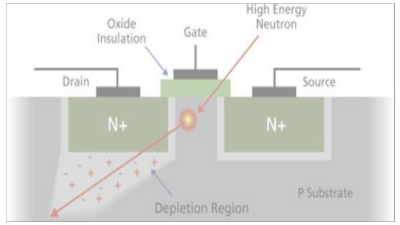
Demonstrate and document that safety mechanisms operate properly

- Safety mechanisms triggered in presence of faulty behavior, and not otherwise
- Safety mechanisms are effective in reaching a safe design state

Random Faults



Permanent



Transient

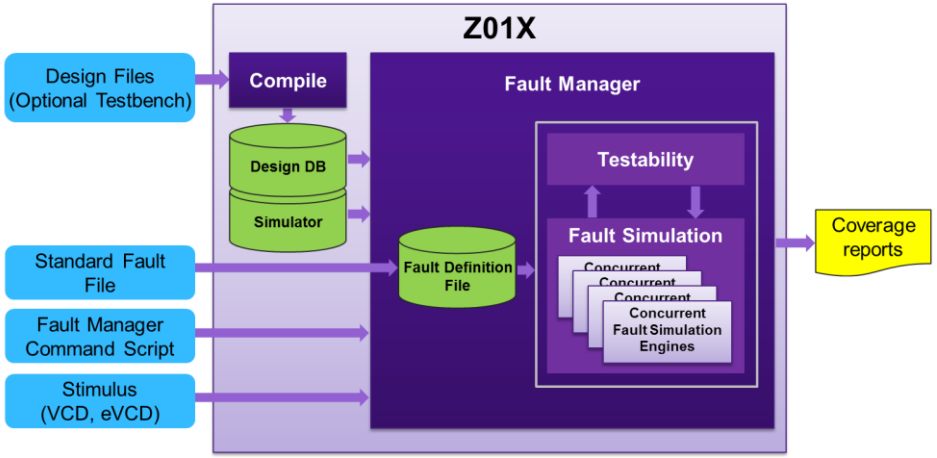
Development → **Manufacturing** → **In Operation**

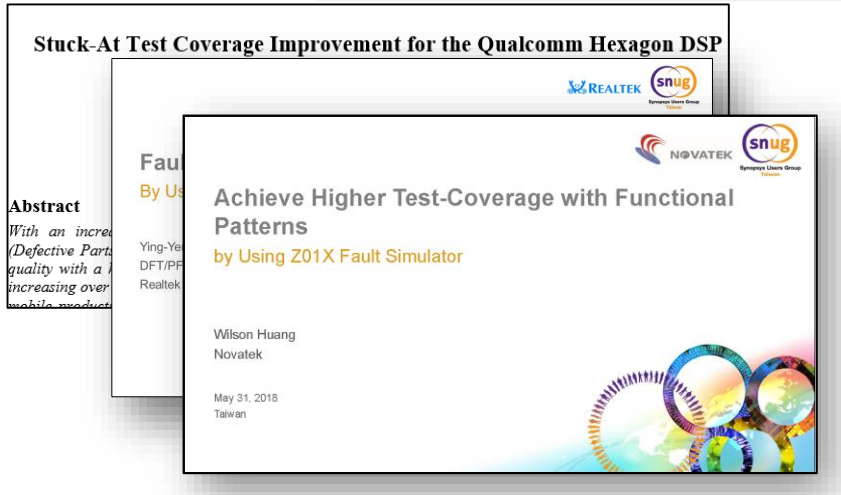
Lifecycle of Component / System / Automobile

Fault Injection Testing – Z01X Manufacturing

Highest performance fault simulation solution

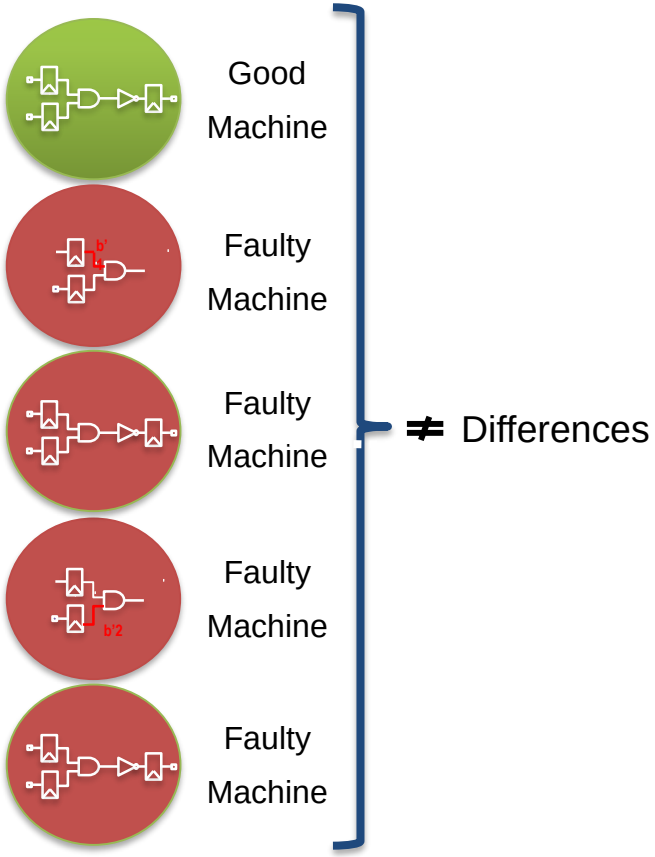
- Challenges
  - Stringent fault simulation is needed for highest fault coverage
  - Comprehensive fault model support is required
  - Performance and capacity demands are extreme
- Objective
  - Generate additional coverage and usefully grade patterns with acceptable TAT
- Results
  - RealTek described their results in SNUG Taiwan 2017
  - NovaTek described their results in SNUG Taiwan 2018
  - Qualcomm scheduled for SNUG Austin 2018



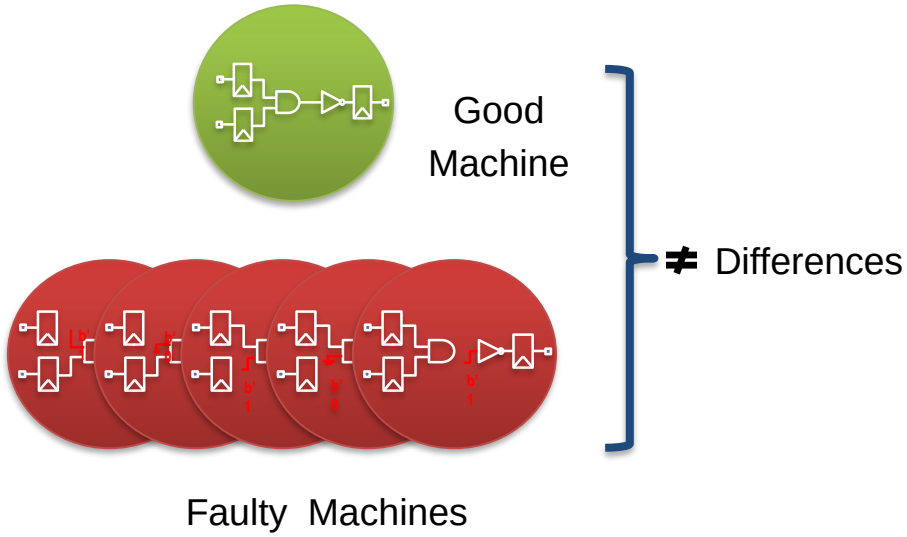


Z01X Concurrent Fault Simulation

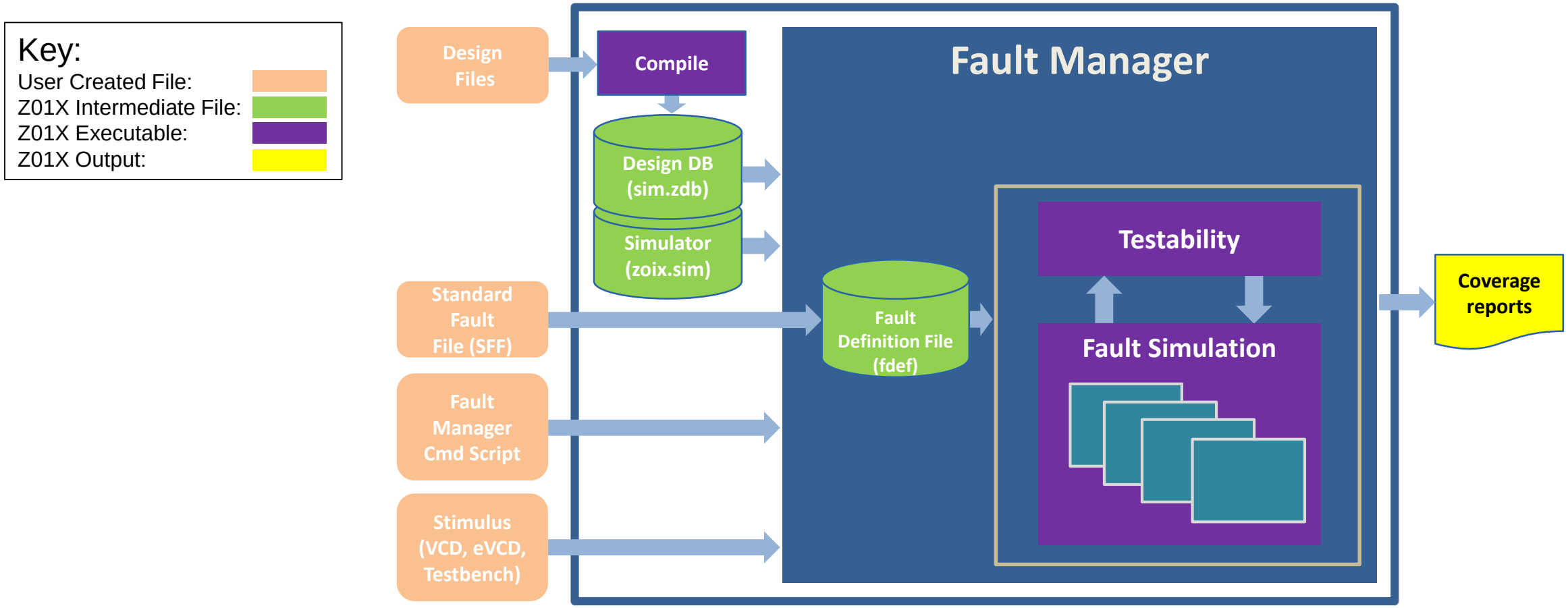
Parallel Simulation Technology
One fault per simulation



Z01X Concurrent Simulation Technology
Thousands of faults in a single simulation
Orders of magnitude faster than parallel



Z01X Flow



If testability shows % coverage below required level. Do not fault simulate!

ISO 26262 Requirements – Hardware Development

Show that design functionality is correct, works properly in the context of the system, and is safe

Demonstrate and document that design and verification flows are robust

- Implementation tools and flows do not introduce design bugs (systematic faults)
- Functional verification tools and flows do not fail to report design bugs

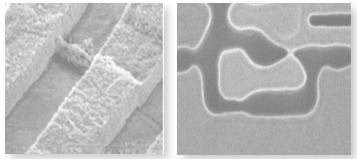
Systematic Faults

Always permanent

Reduced DPPM

- DFT
- Functional patterns

Random Faults

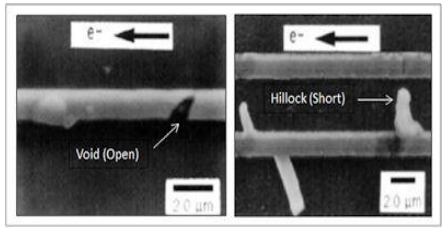


Permanent

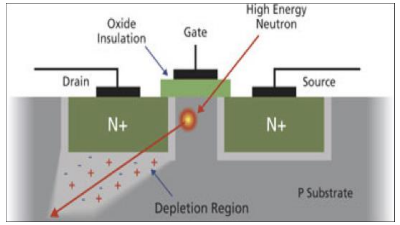
Demonstrate and document that safety mechanisms operate properly

- Safety mechanisms triggered in presence of faulty behavior, and not otherwise
- Safety mechanisms are effective in reaching a safe design state

Random Faults



Permanent



Transient

Development → **Manufacturing** → **In Operation**

Lifecycle of Component / System / Automobile

Verification Goal Comparison

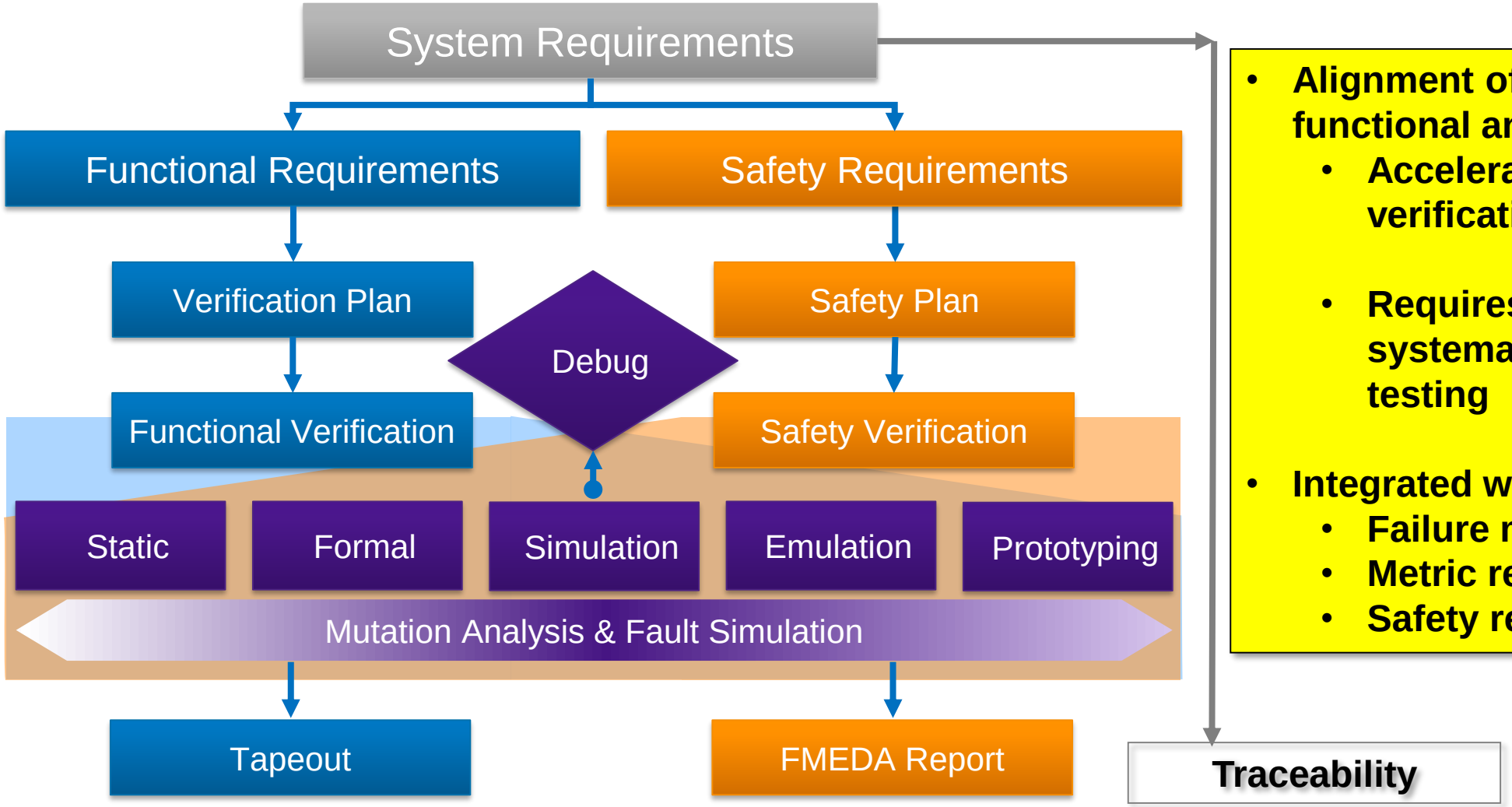
Functional Verification
Prevent / Eliminate Bugs

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

Functional Safety Verification
Control Failures

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

Verification Flow Alignment

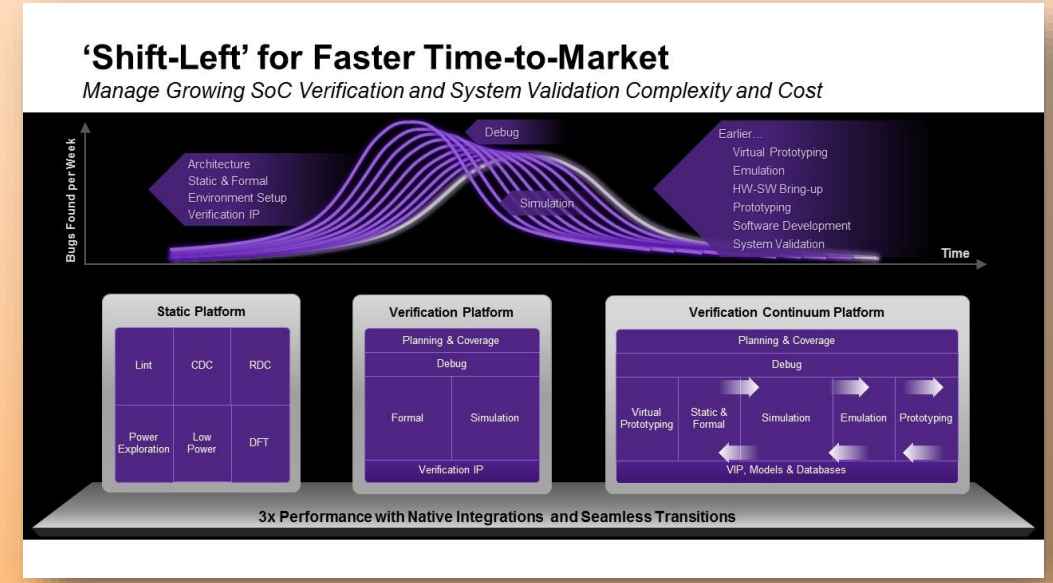


- **Alignment of requirements for functional and safety verification**
  - **Accelerate complete verification process**
  - **Requires solution for systematic and random fault testing**
- **Integrated with ISO 26262 Flows**
  - **Failure mode effects analysis**
  - **Metric reports**
  - **Safety requirements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

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

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 faults

Metric	ASIL B	ASIL C	ASIL D
Single Point Fault Metric	$\geq 90\%$	$\geq \mathbf{97}\%$	$\geq \mathbf{99}\%$

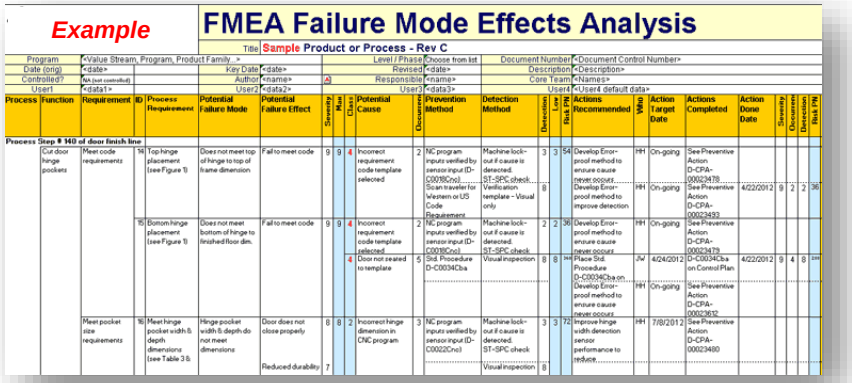
- Maximize detection of multi-point latent faults

Metric	ASIL B	ASIL C	ASIL D
Latent Fault Metric	$\geq 60\%$	$\geq \mathbf{80}\%$	$\geq \mathbf{90}\%$

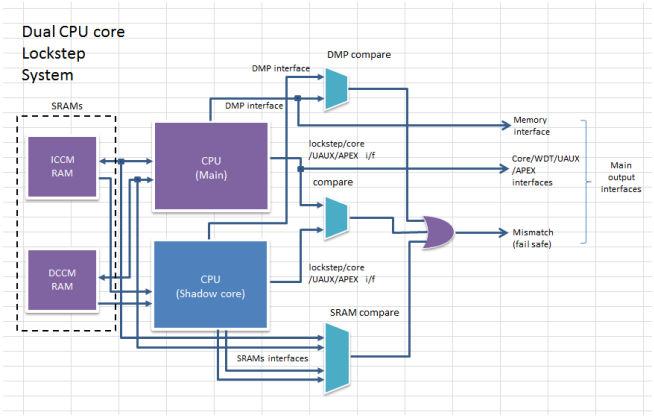
Functional Safety Process

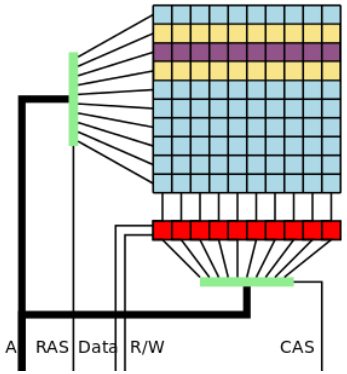
Implement and Confirm Quality of Safety Mechanisms (SM)

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



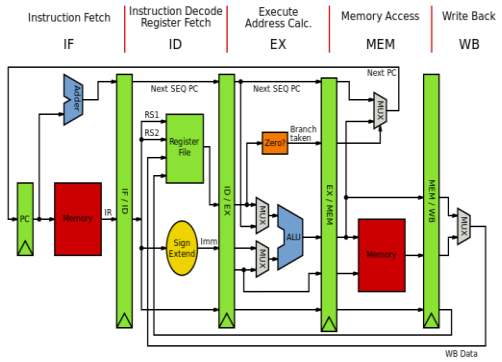
Dual-Core Lockstep





ECC Memory Protection

Software Test Libraries

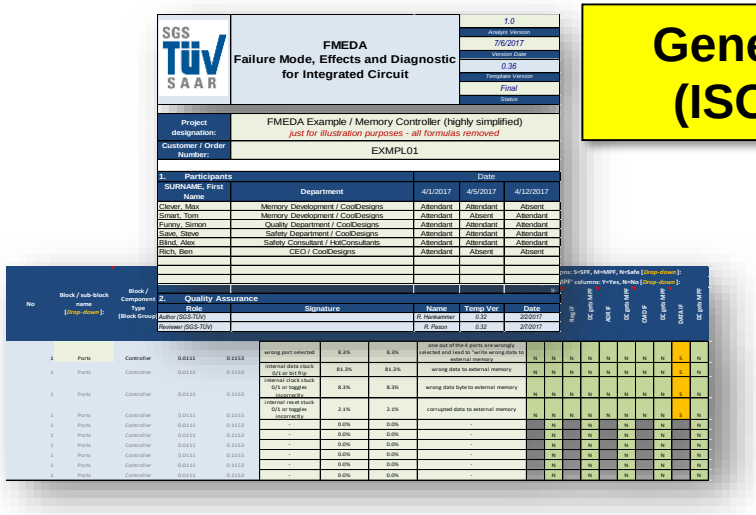


Custom Safety Mechanisms

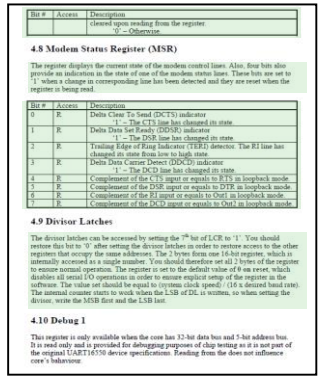
Unique Functional Safety Needs - Summary

Generate FMEDA reports (ISO-26262 deliverable)

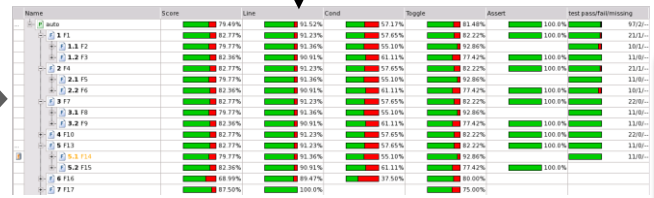
Certified development flows & Safety documentation



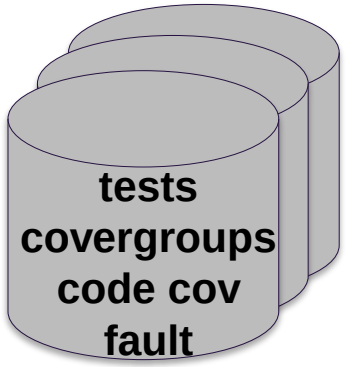
Safety Requirements Traceability



Safety Requirements



Safety Verification Plan



Safety Coverage

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

FMEA/FMEDA process flow

FMEA

Divide design in functional blocks
Component->Parts

Divide further each part to areas with specific functionality, or Sub Parts (SPs)

Identify Failure Modes (FMs) and Safety Mechanisms (SMs) for each SP

Create an FMEA for each FM/SM for a Sub-Part
[One SP may have several FM/SM]

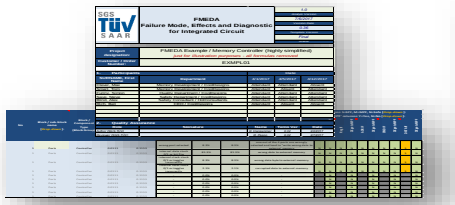
FMEDA

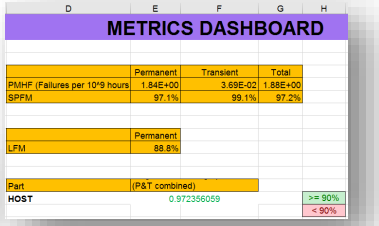
Map each FMEA to the relevant design Sub-Part
[Use a Scaling Factor in case of multiple FMs in the same SP]

Calculate **estimated ISO2626 Metric**
Specify design data, design FIT rate, faults safeness and SMs faults coverage estimates

Run Fault campaign for **measured ISO 26262 Metric**
For each FMEDA define observation and detection points, test suite and faults list

Results should match!





FMEA Inputs example

- Design block level list and diagram.

Reset Logic

Flag Logic

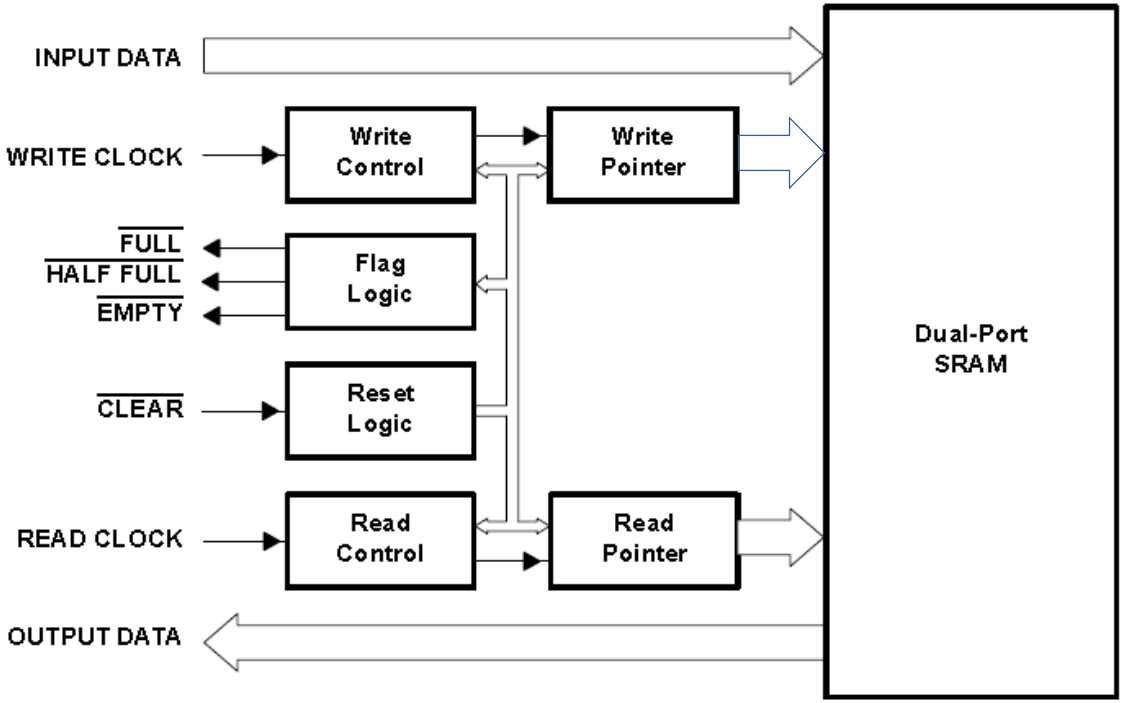
Read Control

Read Pointer

Write Control

Write Pointer

SRAM

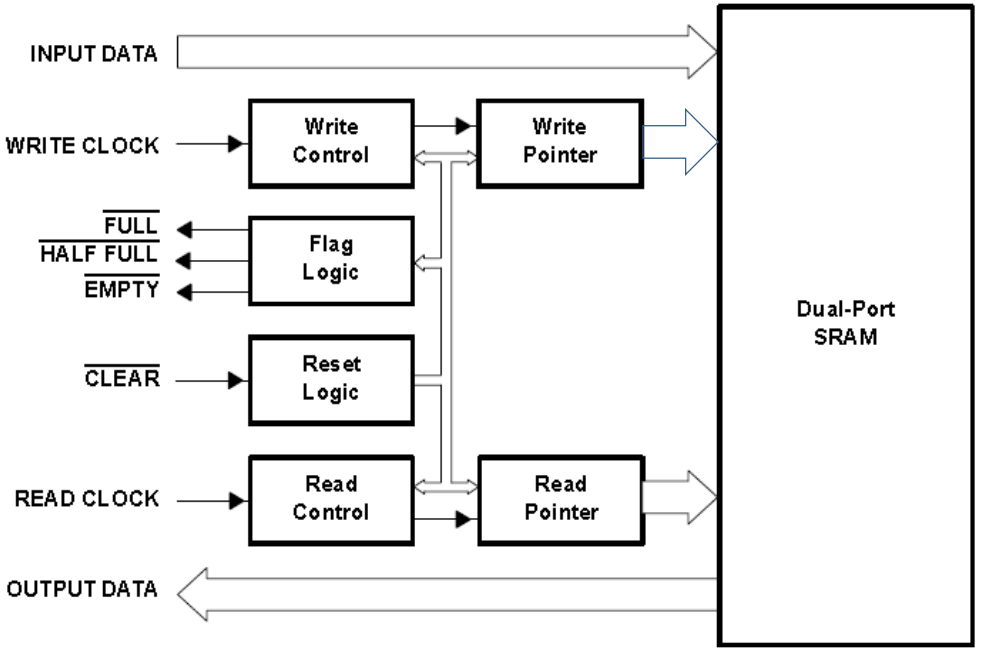


Block Diagram of FIFO with Static Memory

Failure Mode Effect Analysis example

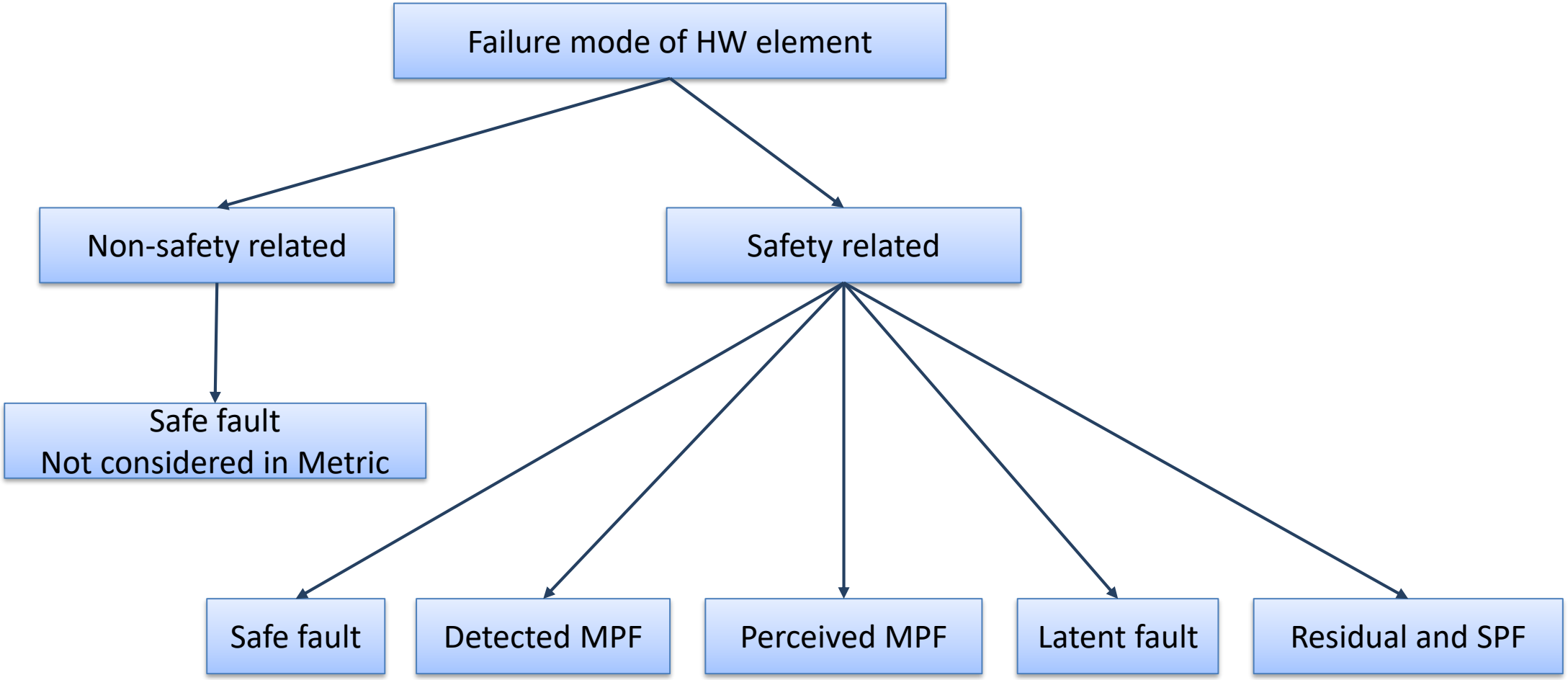
- Failure Mode 1:
  - Failure: Full signal is not raised when FIFO is full
  - Effect: Data will be overwritten or lost
  - Safety Mechanism: Redundant Control Logic

- Failure Mode 2:
  - Failure: Data in SRAM is corrupted
  - Effect: Invalid data
  - Safety Mechanism: ECC



Block Diagram of FIFO with Static Memory

Fault Classification Simplified



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 w	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 w	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
  - 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

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

FMEDA Creation Flow

SP level Analysis – one FMEDA per one FMEA line

Administrator – per Project

Provide Design Data
[Hierarchical data ion all SPs – digital area, RAM bits, FF, Latches etc]

Provide Technology Failure Rate
[FIT per area unit, FIT per RAM bit etc.]

User/IP owner – per Sub Part

Specify SM type
[will provide initial Diagnostic coverage estimate]

Update Fsafe percentage
[Fsafe is the portion of faults which go not violate the safety goal]

Associate a design element
[If multiple FMEDAs on the same design element – use Scale Factor]

→ **Estimated ISO 26262 Metric**

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

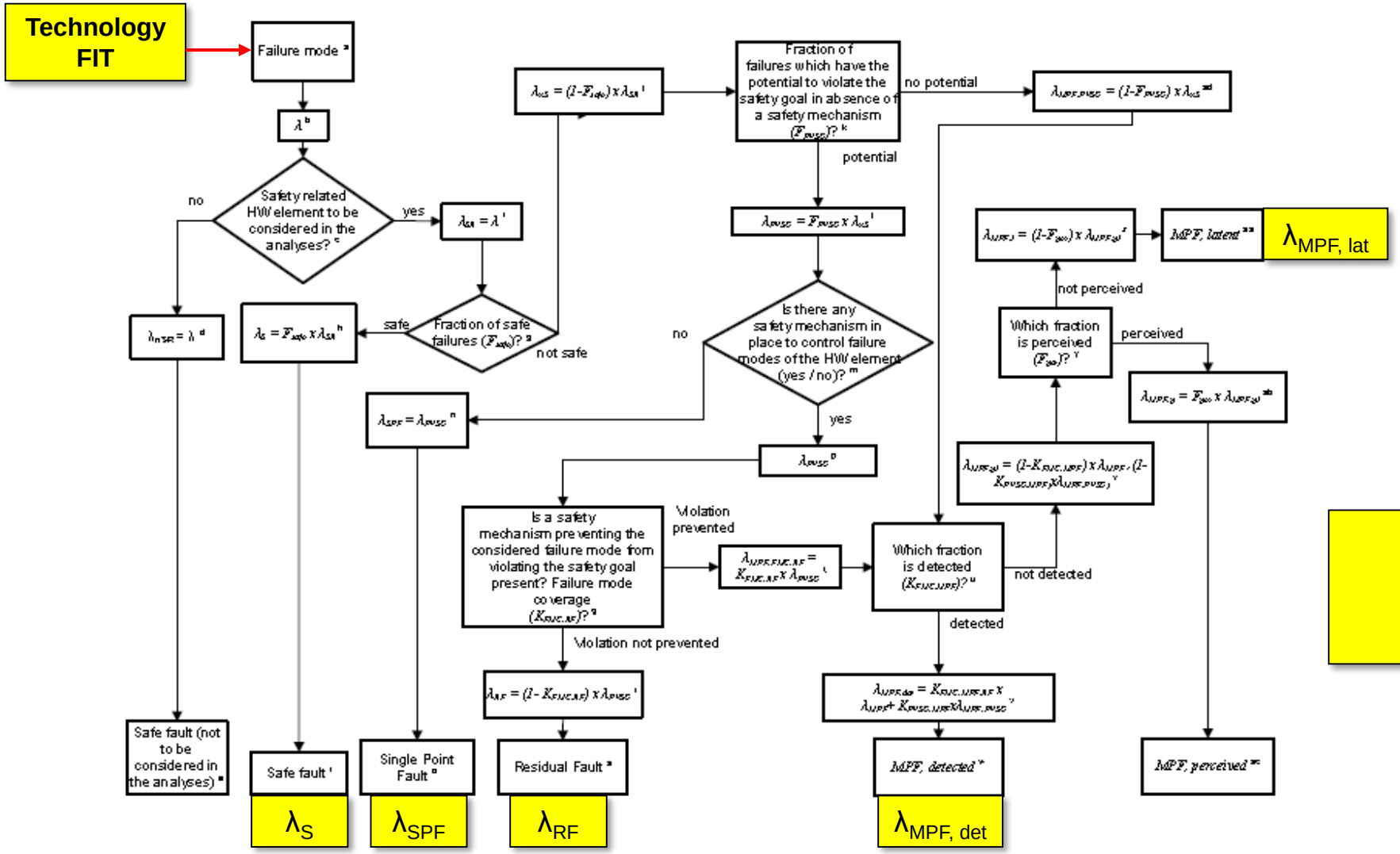
FMEDA Diagnostic Coverage Components

- **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



Source: ISO 26262-5 Annex B

FMEDA Creation Flow

SP level Analysis – one FMEDA per one FMEA line

Administrator – per Project

Provide Design Data
[Hierarchical data ion all SPs – digital area, RAM bits, FF, Latches etc]

Provide Technology Failure Rate
[FIT per area unit, FIT per RAM bit etc.]

User/IP owner – per Sub Part

Specify SM type
[will provide initial Diagnostic coverage estimate]

Update Fsafe percentage
[Fsafe is the portion of faults which go not violate the safety goal]

Associate a design element
[If multiple FMEDAs on the same design element – use Scale Factor]

→ **Estimated ISO 26262 Metric**

Read back results

Fault Simulation Campaign

Prepare a Fault Measurement campaign per FMEDA

→ **Measured ISO 26262 Metric**

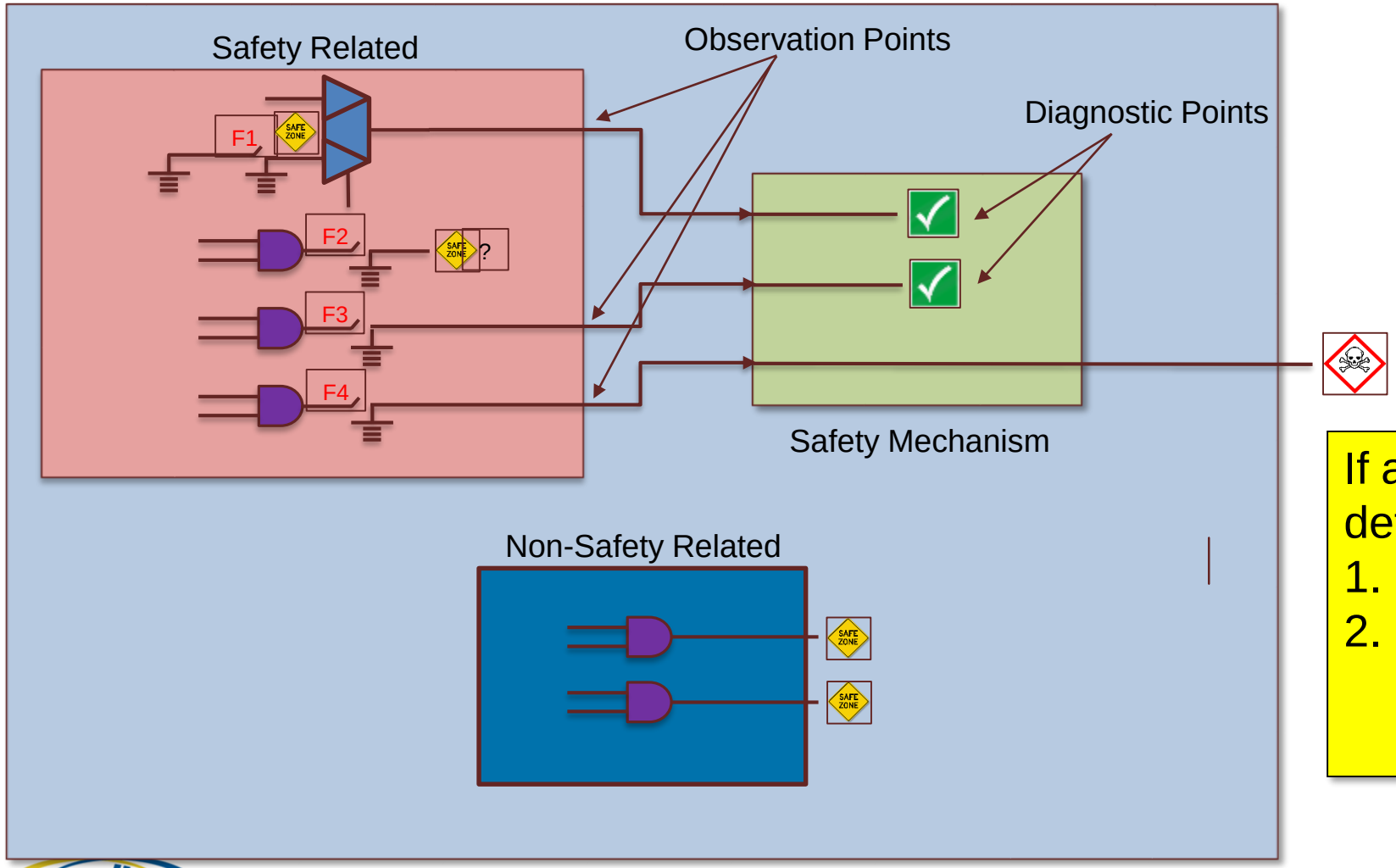
Fault Injection Campaign

- Determine Diagnostic Coverage of the SM
  - inject faults in the design
  - checking if they are detected by the SM

- 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 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

Preparation for Fault Simulation

Set up FMEDA observation points by analyzing failure mode effects

A fault which does not propagate to any observation point is either safe, or ‘dangerous undetected’

Set up FMEDA test suite (get it from module/system verification suite)

Workload should toggle the logic around the fault as much as possible

Set up FMEDA detection points according to safety mechanism

A detection point is the physical net which toggles as a results of the safety mechanism detecting the fault

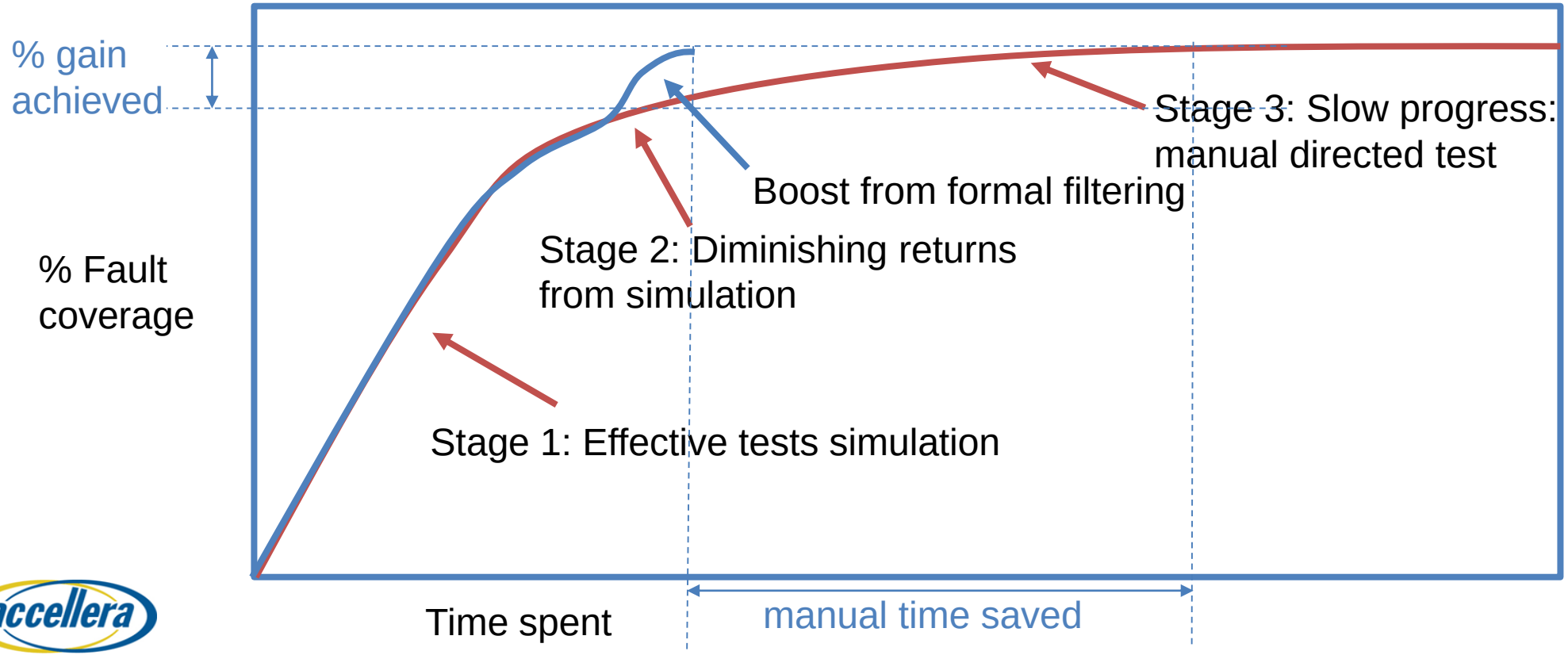
Unfavorable Simulation Results Analysis

- A fault which does not propagate to any observation point is either safe, or ‘dangerous undetected’
- Use Formal tool to further classify faults
- Provide DUT and Fault list to VC-Formal
  - Fault proven to be not-controllable or not observable
    - Fault is Safe
  - Fault proven to be controllable and observable
    - Analyze scenario provided by VC-Formal and improve provided use case(s)

    ⇒ Productivity and safety increased
  - Inconclusive
    - Human analysis required

Benefits of Formal Fault Filtering

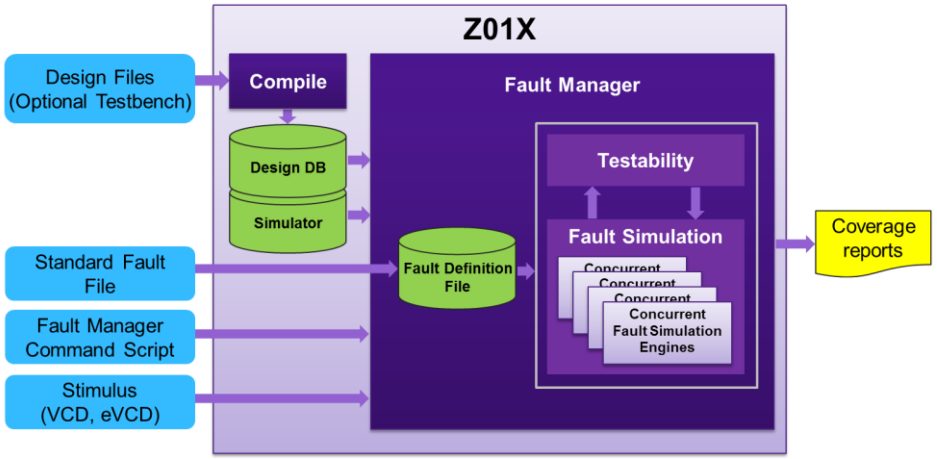
- Simulation and constrained random tests help achieve high % of fault coverage quickly
- Eventually the benefits of simulation and manual directed tests diminish: progress plateau
- Formal filtering of faults can provide a boost to fault coverage %



Fault Injection Testing – Z01X Functional Safety

Highest performance fault simulation solution for ISO 26262 compliance requirements

- Challenges
  - Stringent fault simulation is needed for ISO 26262 compliance
  - Both permanent and transient fault model support is required
  - Performance demands are extreme
- Objective
  - Generate fault coverage metrics with acceptable TAT
- Results
  - MobilEye adopted Z01X for their EyeQ4 design when existing (competitive) solution was too slow
  - Z01X adoption WW is growing rapidly in automotive semiconductor and systems companies



Mobileye Adopts Key Synopsys Automotive Functional Safety Verification Solution to Enable ISO 26262 Compliance of its Next-Generation ADAS SoCs

Mobileye Adopts Z01X Functional Safety for EyeQ4

Nov 21, 2016

Synopsys Accelerates Development of Safety-Critical Products with Design Solutins for ARM Cortex-R52

High speed Z01X and Certitude fault simulation help assure functional safety for automotive safety standards

Sep 19, 2016

FMEA/FMEDA Process Overview (ISO 26262)

SoC -> IP analysis

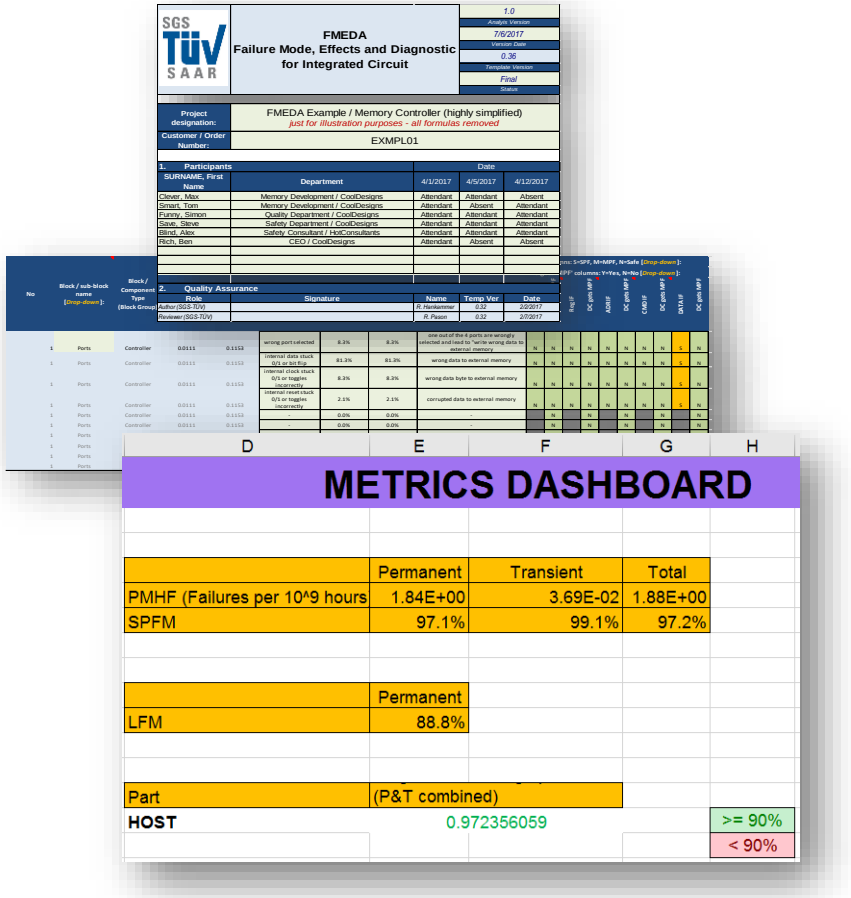
Design Data, Technology Data

Create FMEA

Create FMEDA

Fault Coverage Measurement:

- Formal Fault Reduction
- Fault Simulation



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%

Synopsys' Unique Position for Automotive Verification

- Deep R&D collaboration with leading automotive semiconductor suppliers
- Automotive supply chain relationships with Tier1 and OEMs

- Fastest verification engines: Static, Formal, Simulation, Emulation, FPGA prototyping
- Early SW development platform with hybrid emulation and Verdi HW/SW debug

- Unique technologies: Certitude, Z01X, FMEDA automation, virtual prototyping and models
- ISO26262 technical expertise and experience

Questions?

Contact: jmforey@synopsys.com

Thank You