

USF-based FMEDA-driven Functional Safety Verification

Francesco Lertora, Software Engineering Group Director, SVG

Frederico Ferlini, Sr. Principal Product Engineer, SVG

15 September 2024

Outline

- Session 1
 - Introduction
 - Functional Safety Analysis Overview
 - Deep Dive
 - Architectural FMEDA
 - Detailed FMEDA
 - SoC Safety Analysis
 - Safety Metrics Verification
- Session 2
 - Fault Campaign Management
- Summary

EDA as an Ecosystem of International and Industry Standards

1800 - IEEE Standard for SystemVerilog Unified Hardware Design, Specification, and Verification Language

1364 - IEEE Standard for Verilog Hardware Description Language

1076 - IEEE Standard for VHDL Language Reference Manual

Library Exchange Format (LEF)/Design Exchange Format (DEF)

1801 - IEEE Standard for Design and Verification of Low-Power, Energy-Aware Electronic Systems

Timing Constraints – SDC

1497 IEEE Standard for Standard Delay Format (SDF) for the Electronic Design Process

Liberty™ library format

GDSII - Graphic Design System

OASIS® – Open Artwork System Interchange Standard

1685 - IEEE Standard for IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows

Why not for safety?

- Describe safety features, targets (intent) and exchange safety-related information

Motivations & Mission

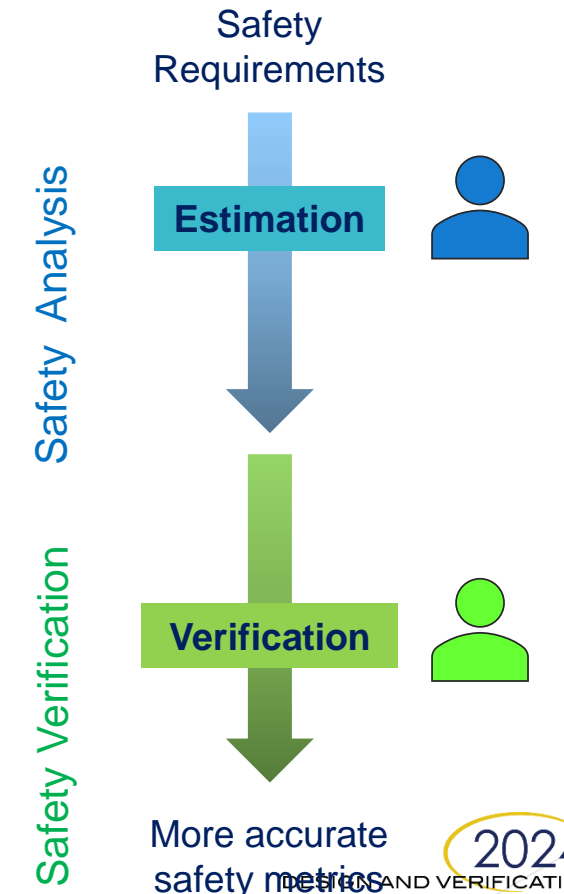
- Lack of formalism, standards ambiguity, differentiated assessors approach, lead to customer-specific methodologies + widespread usage of Spreadsheets
 - «consulting-driven» market side-effects:
 - ‘keep it obscure’
 - ‘this is *my* (certified) methodology’
 - ‘(only) We will tell you what you have to do’...

To develop a modular safety analysis platform to exchange safety-related information and to enable Design For Safety with Cadence® Tools

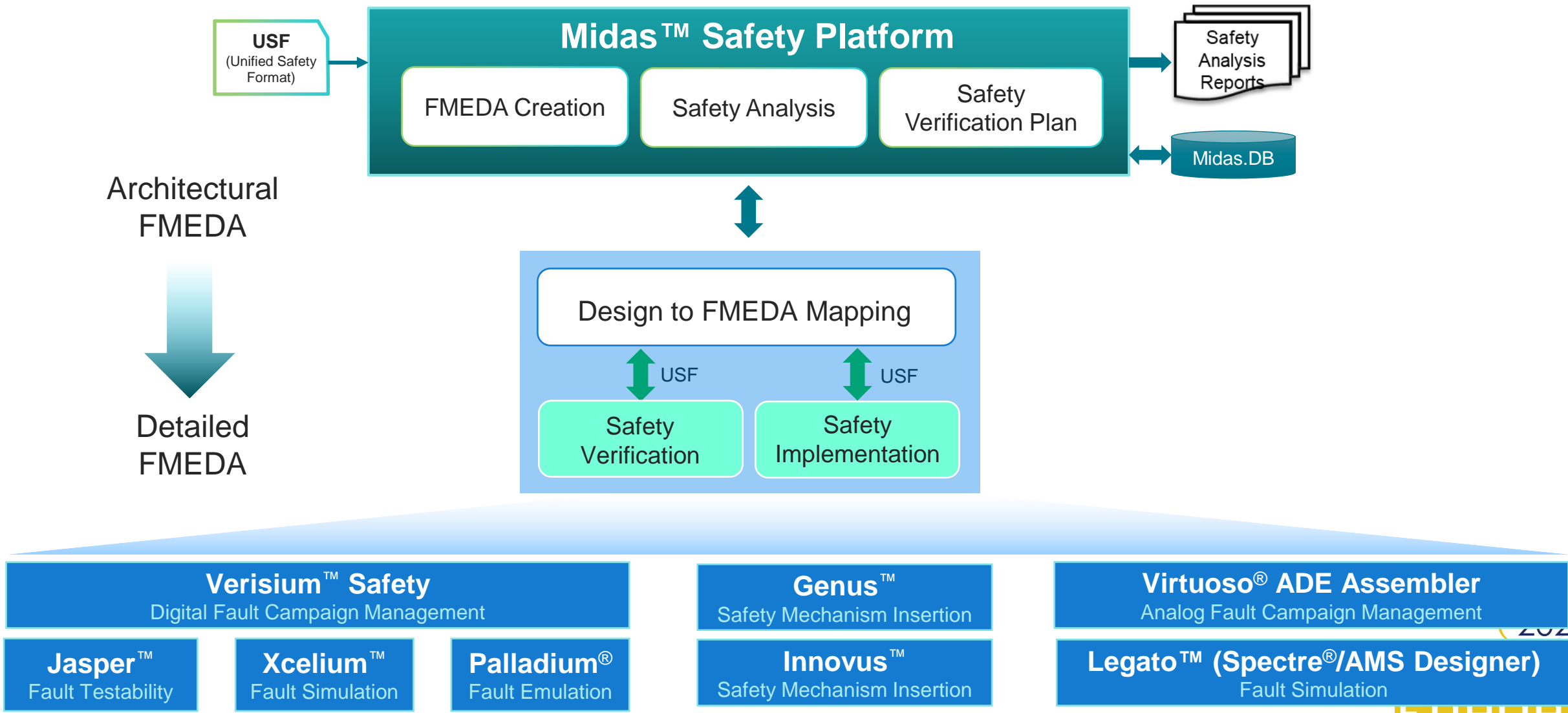
- Cadence is committed to adopt and support the IEEE 2851 family of standards

Closing the Gap between FMEDA and Safety Verification

	Abstraction	Safety Step	User
Functional Safety Concept	Functional	FMEA	Safety Architect (System level)
Technical Safety Concept	Block Diagram	FMEDA (architectural)	Safety Architect (SoC level)
Design	RTL/Netlist	FMEDA (detailed)	Safety Engineer (RTL/gate level)
Safety Verification	Netlist	Safety Verification (Formal/Fault Injection)	Safety Verification Engineer
Safety Metrics	Verification Result	FMEDA backannotation	Safety Verification Engineer



Cadence Functional Safety Full Flow



Digital Safety Verification

Fault campaign management, analysis, simulation and emulation

Fault Campaign Management – Verisium Safety

Unified campaign management across all engines

Backannotation of DC results into Midas FMEDA

Provides requirements traceability and reporting

Fault Analysis – Jasper FSV App

Structural analysis to reduce the fault list

Formal analysis for accurate fault classification

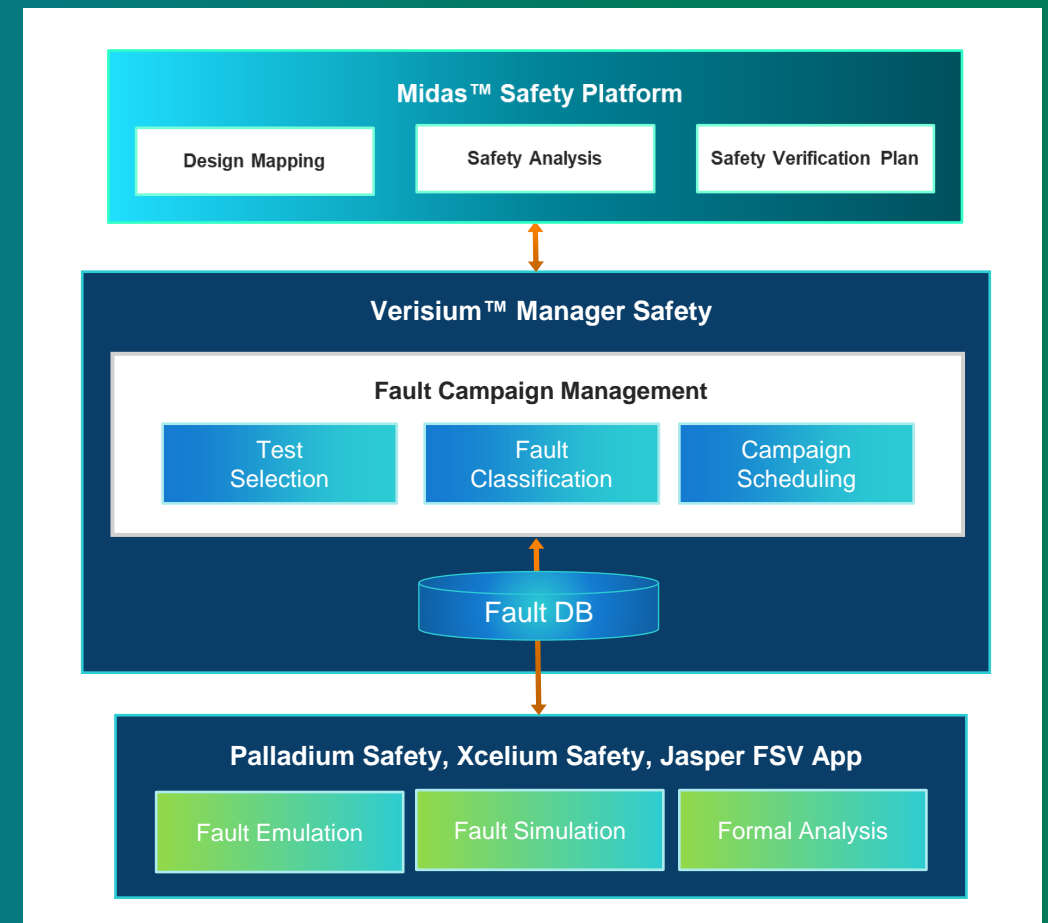
Fault Simulation – Xcelium Safety

Native serial and concurrent fault verification

Same simulator for functional verification (GOOD machine) and fault simulation (BAD machine)

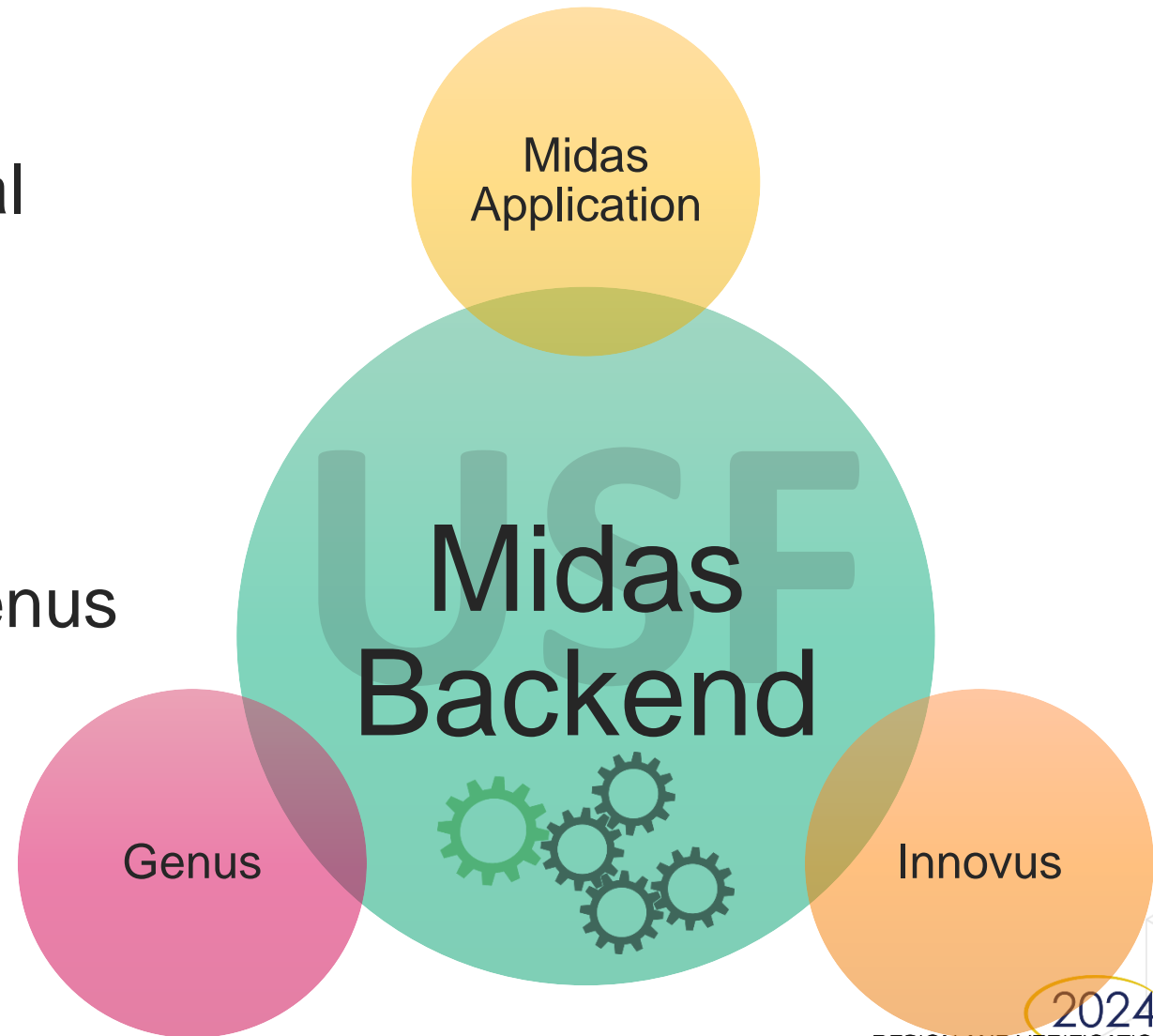
Fault Emulation – Palladium Safety

Run full SoC with SW or STLs



Midas Safety Platform Modularity

- The Midas backend is the ‘functional safety engine’
 - Support for Midas command line interface
 - ISO26262; IEC61508
 - BFR
- Same backend is integrated into Genus and Innovus
- Core features can be made easily available in different contexts





Functional Safety Analysis Overview



Functional Safety Analysis

Architectural FMEDA

Detailed FMEDA

- Device Safety (IP/SoC) architectures
- No direct access to design information

- During or after design implementation
- Using real design information

FMEDA Project (IP and SoC)

IP FMEDA, FMEDAs grouping and SGs definition

BFR calculation engine (IEC TR 62380)

Technologies (Digital, Analog, ...)

Safety Hierarchy (Parts/Subparts)

Failure Modes

Safety Mechanisms

DC on FM-SM, different DC heuristics for combining from multiple SM

Mapping Safety Hierarchy to Design Hierarchy

Only for a detailed FMEDA: direct (with – exclude support) or extraction-based (COI)

Metrics & Reports

Queries

Rules check

Custom attributes, What-if analysis, flexible-customizable template

Architectural FMEDA

USF

FMEDA Project (IP and SoC)

```
set_fmEDA myFMEDA -ASIL B -t -p -arch
```

BFR calculation engine (IEC TR 62380)

Technologies (Digital, Analog, ...)

```
create_technology DigLib -type Digital -fitperm 1.07e-6 -  
fittrans_gate 1.64e-6 -fitbit 1.64e-6 -refarea 1.026
```

Safety Hierarchy (Parts/Subparts)

```
create_part "OpenRISC Core" -fmEDA myFMEDA  
create_subpart FETCH -desc "Instruction Fetch Unit" -part  
"OpenRISC Core" -fmEDA myFMEDA
```

Failure Modes

```
create_failure_mode FM_ARCH_1 -desc "Any failures of FETCH sub-  
block" -type Mission -technology DigLib -subpart FETCH -gates  
2500 -flops 100 -safe_perm 1 -safe_trans 0 -fmEDA myFMEDA
```

Safety Mechanisms

```
create_safety_mechanism SM-IF -desc "Instruction Fetch  
redundancy" -type Custom -class HW  
apply_safety_mechanism SM-IF -to FM_ARCH_1 -fmEDA myFMEDA -  
dcperm 95 -dctrans 0 -dclat 100
```

Mapping Safety Hierarchy to Design Hierarchy

Metrics & Reports

Queries

```
report_safety -fmEDA myFMEDA permanent html Permanent.html  
report_safety -fmEDA myFMEDA transient csv Transient.csv  
query_usf myFMEDA -obj_type failure_mode -obj_id FM_ARCH_1
```

Detailed FMEDA

USF

FMEDA Project (IP and SoC)

```
set_fmEDA myFMEDA -ASIL B -t -p -detailed
```

BFR calculation engine (IEC TR 62380)

```
create_technology DigLib -type Digital -fitperm 1.07e-6 -  
fittrans_gate 1.64e-6 -fitbit 1.64e-6 -refarea 1.026
```

Technologies (Digital, Analog, ...)

```
create_part "OpenRISC Core" -fmEDA myFMEDA -instances  
{hinst:or1200_cpu/or1200_if hinst:or1200_cpu/or1200_genpc}  
create_subpart FETCH -desc "Instruction Fetch Unit" -part  
"OpenRISC Core" -fmEDA myFMEDA -instances  
{hinst:or1200_cpu/or1200_if}
```

Safety Hierarchy (Parts/Subparts)

Failure Modes

```
create_failure_mode FM_ARCH_1 -desc "Any failures of FETCH sub-  
block" -type Mission -technology DigLib -subpart FETCH -  
safe_perm 1 -safe_trans 0 -fmEDA myFMEDA -instances  
{hinst:or1200_cpu/or1200_if}
```

Safety Mechanisms

```
create_safety_mechanism SM-IF -desc "Instruction Fetch  
redundancy" -type Custom -class HW  
apply_safety_mechanism SM-IF -to FM_ARCH_1 -fmEDA myFMEDA -  
dcperm 95 -dctrans 0 -dclat 100
```

Mapping Safety Hierarchy to Design Hierarchy

Metrics & Reports

Queries

```
report_safety -fmEDA myFMEDA permanent html Permanent.html  
report_safety -fmEDA myFMEDA transient csv Transient.csv  
query_usf myFMEDA -obj_type failure_mode -obj_id FM_ARCH_1
```

Refine FMEDA Data for Optimized Safety Design

Architectural FMEDA

Counts	9 Parts / 16 Sub-Parts / 35 Failure Modes
Total FIT (Raw FIT) Permanent - A	1.030e-01
Total FIT (Raw FIT) Transient - A	1.651e-01
Safety related FIT Permanent - Asr	1.005e-01
Safety related FIT Transient - Asr	1.611e-01
Probabilistic Metric for random Hardware Failures PPHF Permanent - in FIT	4.269e-02
Probabilistic Metric for random Hardware Failures PPHF Transient - in FIT	6.753e-02
Probabilistic Metric for random Hardware Failures PPHF Latent - in FIT	0.000e+00
Single Point Fault Metric - SPFH Permanent	57.52%
Single Point Fault Metric - SPFH Transient	58.09%
Latent Fault Metric - LFM	100.00%
Total Not Safety Related faults Permanent - AnSR	2.500e-03
Total Not Safety Related faults Transient - AnSR	4.000e-03

Detailed FMEDA

Counts	9 Parts / 16 Sub-Parts / 35 Failure Modes
Total residual faults Permanent - Arf	1.215e-02
Total residual faults Transient - Arf	1.948e-02
Total Multi Point Primary - Ampf	0.000e+00
Total Multi Point Secondary Permanent - Ampf	5.781e-02
Total Multi Point Secondary Transient - Ampf	9.360e-02
Total Multi Point Detected - Ampf_det	5.781e-02
Total Multi Point Latent faults - Ampf_l	0.000e+00
Technologies	DigiLib
Design Information	Total Area: 134678.6 #Eq. Gates: 131265.7 #Flips: 6563.0
Design Information - Mapped Failure Modes	Total Area: 98728.7 #Eq. Gates: 96219.0 #Flips: 4431.0
Design Information for Mapped Safety Relevant Failure Modes	Total Area: 96364.7 #Eq. Gates: 93822.7 #Flips: 4328.0
SPFHp (Digital)	57.52%

Optimized Safety Design

Counts	9 Parts / 16 Sub-Parts / 35 Failure Modes
Total FIT (Raw FIT) Permanent - A	1.030e-01
Total FIT (Raw FIT) Transient - A	1.729e-01
Safety related FIT Permanent - Asr	1.005e-01
Safety related FIT Transient - Asr	1.690e-01
Probabilistic Metric for random Hardware Failures PPHF Permanent - in FIT	4.269e-02
Probabilistic Metric for random Hardware Failures PPHF Transient - in FIT	6.761e-02
Probabilistic Metric for random Hardware Failures PPHF Latent - in FIT	0.000e+00
Single Point Fault Metric - SPFH Permanent	57.52%
Single Point Fault Metric - SPFH Transient	59.99%
Latent Fault Metric - LFM	100.00%
Total Not Safety Related faults Permanent - AnSR	2.500e-03
Total Not Safety Related faults Transient - AnSR	3.900e-03

2%
higher
SPFM

- No design data available
- FMEDA hierarchy only
- Failure rates and distribution solely based on early estimations

- With design data
- Design to FMEDA hierarchy mapping
- HW safety metric based on design data

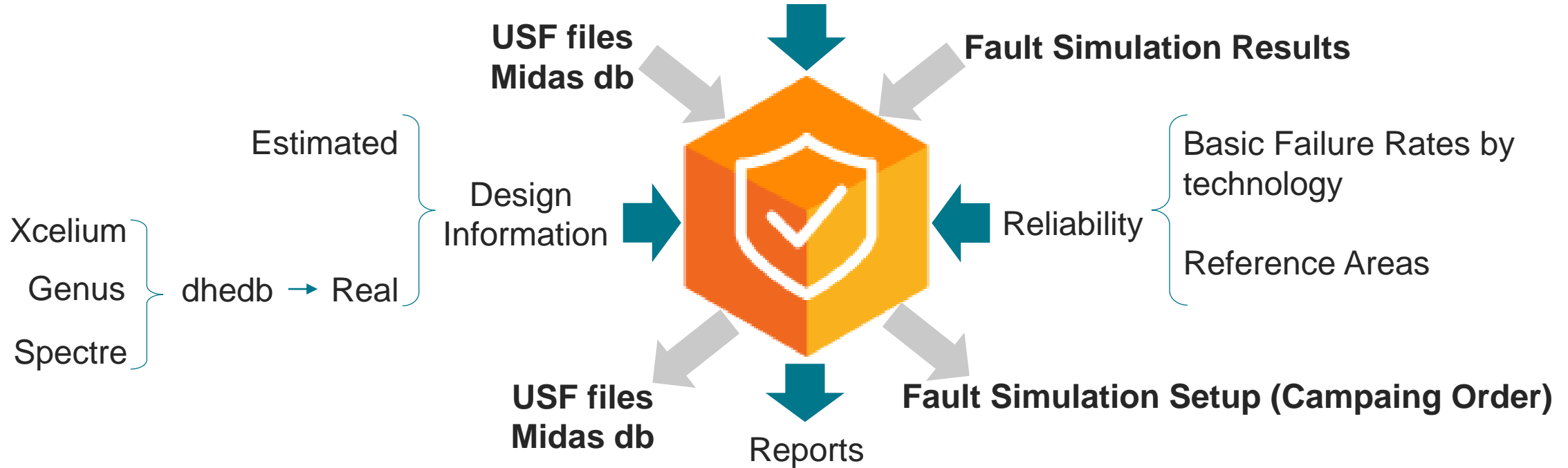
- With design & simulation data
- Design to FMEDA hierarchy mapping
- HW safety metric based on design & simulation data

Optimized FMEDA metric by using design & simulation-based data

Inputs / Outputs

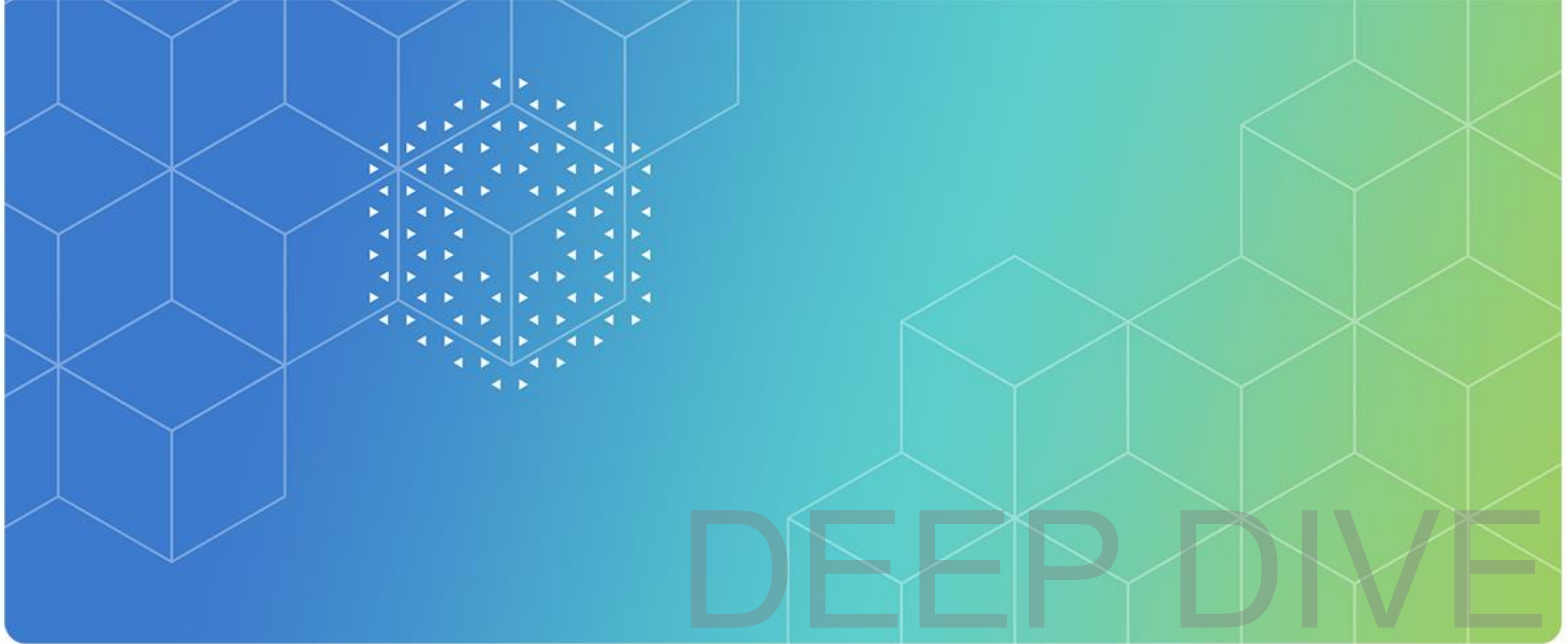
- Definition of the FMEDA Project
- Parts, Subparts, Failure Modes, Safety Mechanism
- Design Mapping (for a Detailed FMEDA)
- Import Spreadsheets

FMEDA Authoring



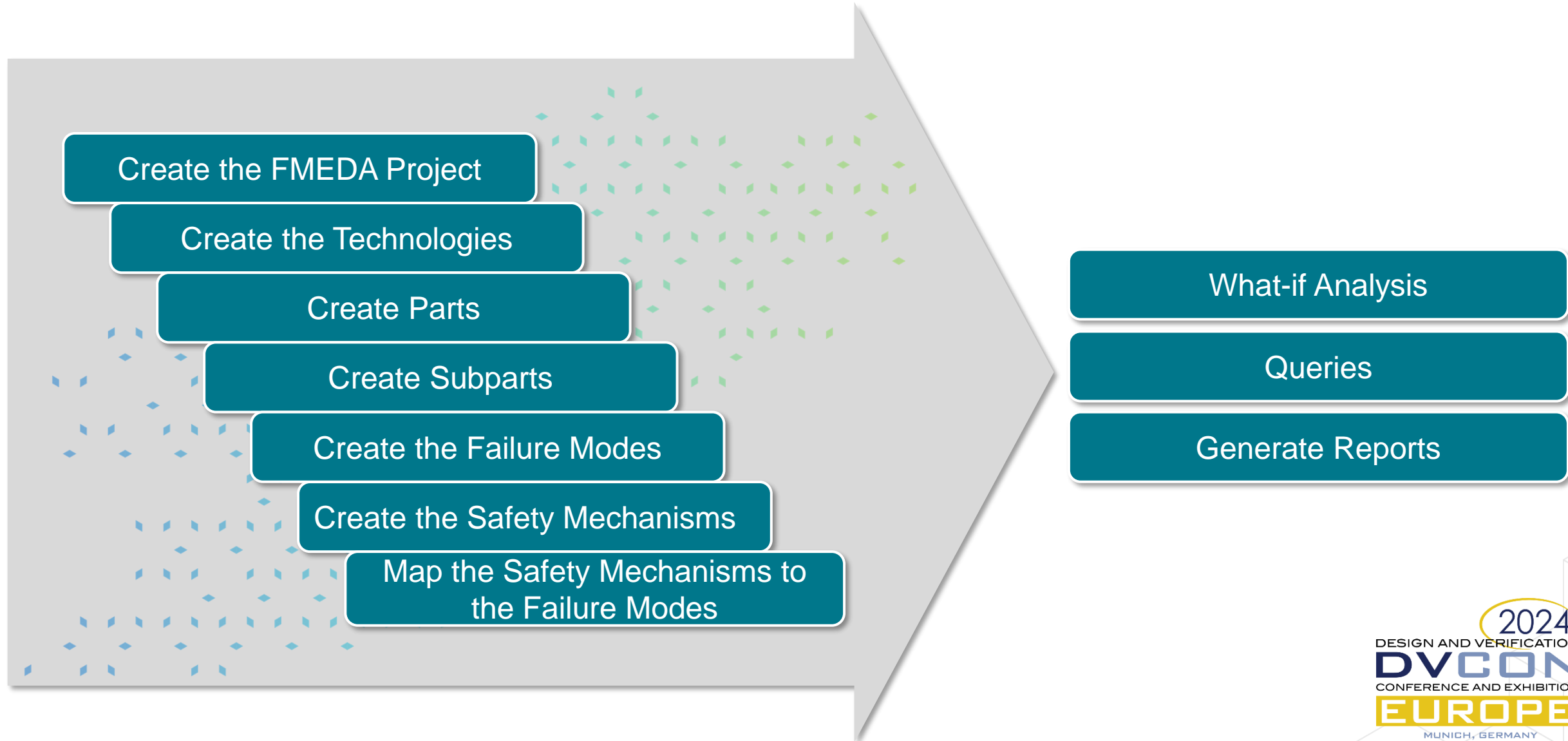
- FMEDA (Permanent+Transient)
- Summary
- SoC Report





Architectural FMEDA

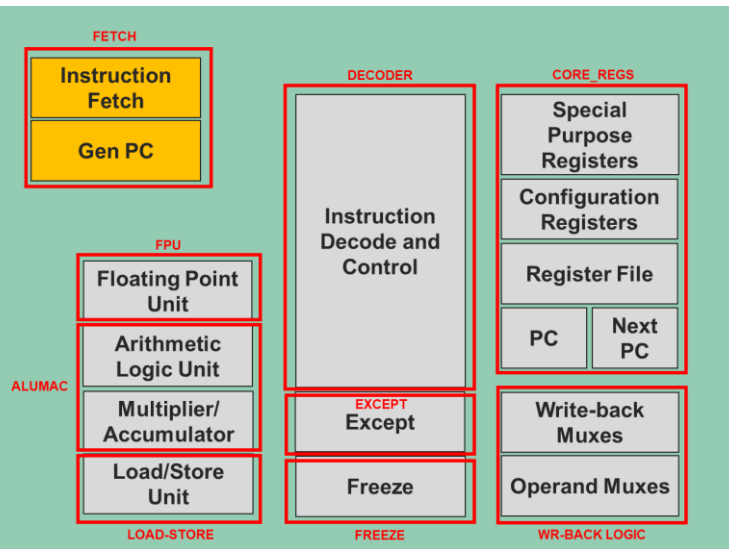
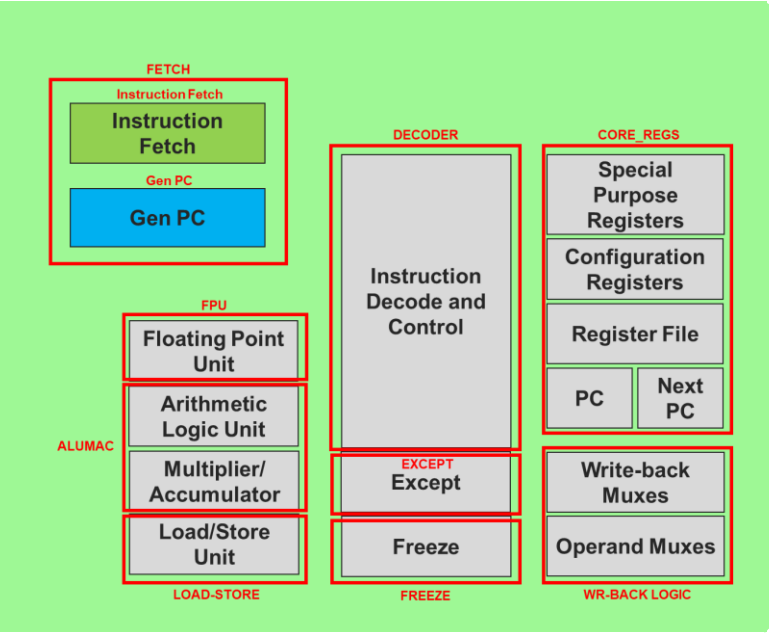
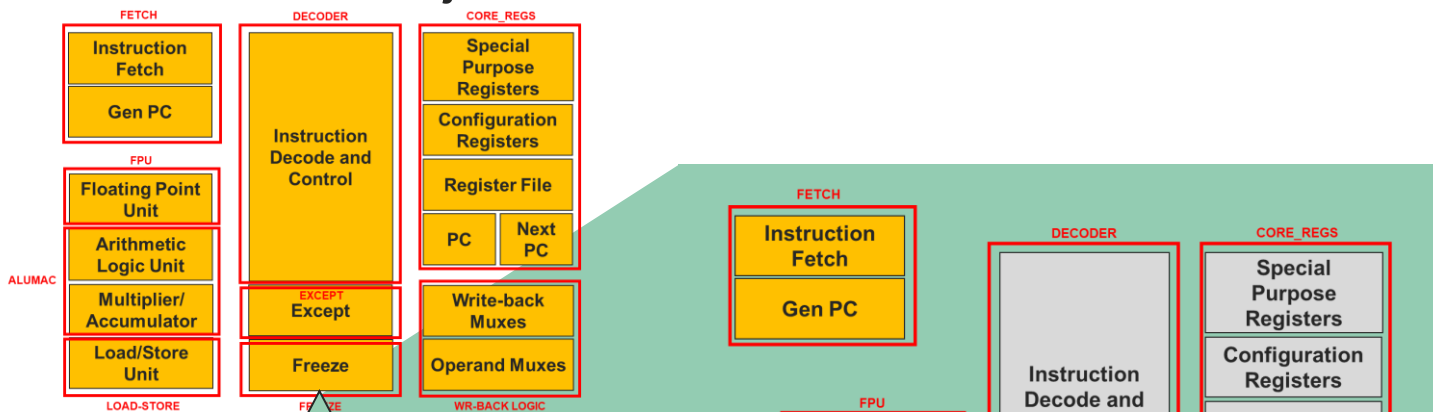
Architectural FMECA Authoring Steps



Design Decomposition

- Safety analysis are typically performed with a reduced number of hierarchical levels compared with the design hierarchy

FMEDA Project



Parts

First level of hierarchical decomposition

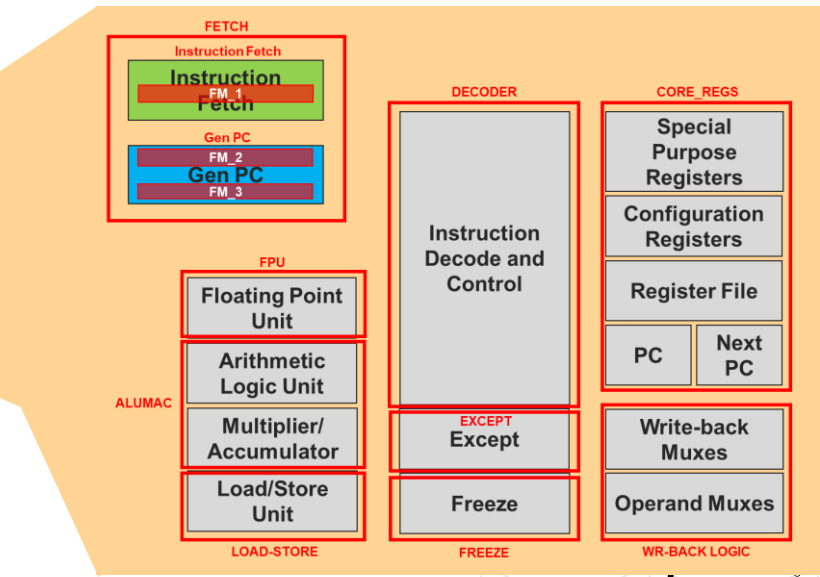
Subparts

Second or greater level of hierarchical decomposition

Elementary Subparts

Failure Modes

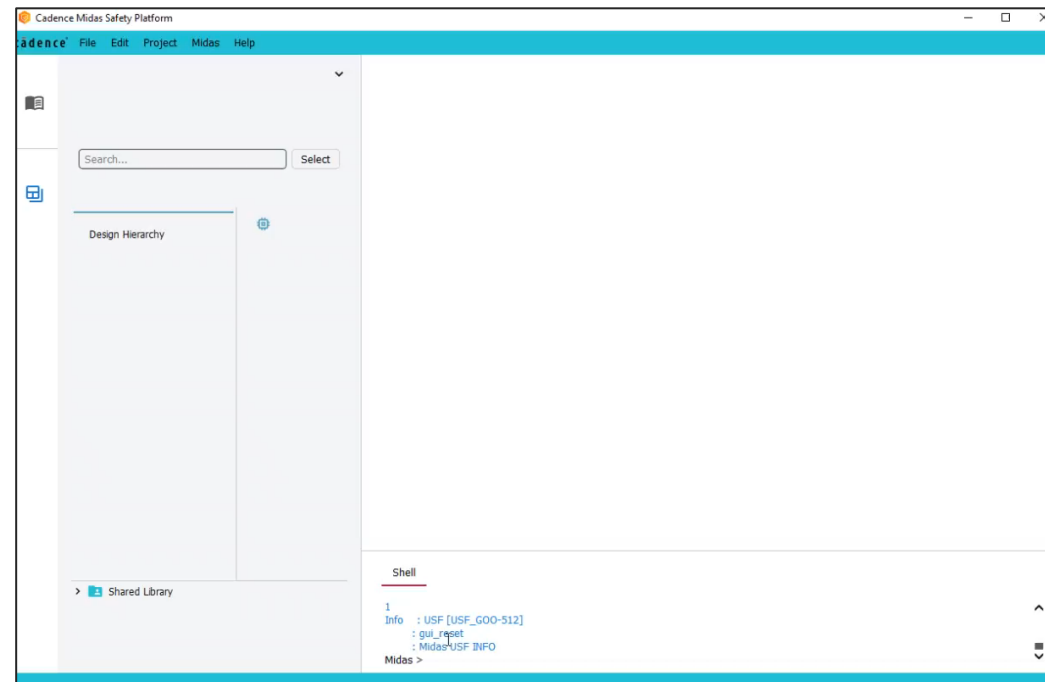
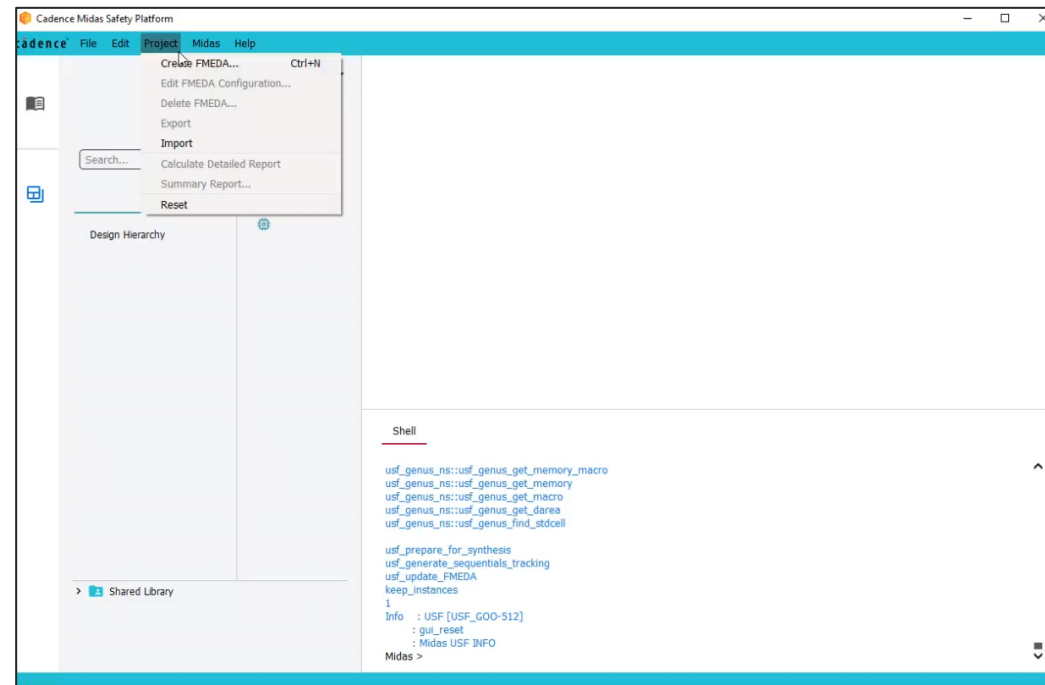
Manner in which an element or an item fails to provide the intended behavior



- The target is to define the failure modes, not to describe the circuit functionalities

Functional Safety Authoring

- The GUI provides an user-friendly FMEDA authoring environment
- Safety objects can also be created with USF commands
- The solution is fully scriptable
- Mixing GUI and scripted-automations is further possible



What-if Analysis: FMEDA Static Configurations

- Create configurations changing values in the FMEDA (e.g., design info., SM DCs)
- Each configuration generates safety metrics to be compared
- The configurations can be saved and restored

The screenshot displays the Cadence Midas Safety Platform interface for the project 'OR1200_architectural'. The main window shows a table of FMEDA configurations with columns for Name, Count, SPFMp, SPFMT, LFM, and PMHFp. The 'SM-IF3 off' configuration is highlighted in green. A 'Configuration Parameters - SM-IF3 off' dialog box is open, showing a table with columns for Name, Parameter, and Value. The dialog shows 'SM-IF3/FM_ARCH_1' with a parameter of 'Status' and a value of 'Off'. The background shows a design hierarchy on the left and a shell window with various warnings and messages.

Name	Count	SPFMp	SPFMT	LFM	PMHFp
Default	1 Parts / 10 Sub-Parts / 8 Failure Modes	87.85%	85.18%	99.58%	1.042e-02
Add SM-RF_2 to FM_ARCH_5	1 Parts / 10 Sub-Parts / 8 Failure Modes	87.85%	85.18%	99.58%	1.042e-02
Large FM_ARCH_2	1 Parts / 10 Sub-Parts / 8 Failure Modes	85.86%	85.14%	99.64%	1.441e-02
SM-IF3 off	1 Parts / 10 Sub-Parts / 8 Failure Modes	84.89%	82.78%	99.56%	1.281e-02

Name	Parameter	Value
SM-IF3/FM_ARCH_1	Status	Off

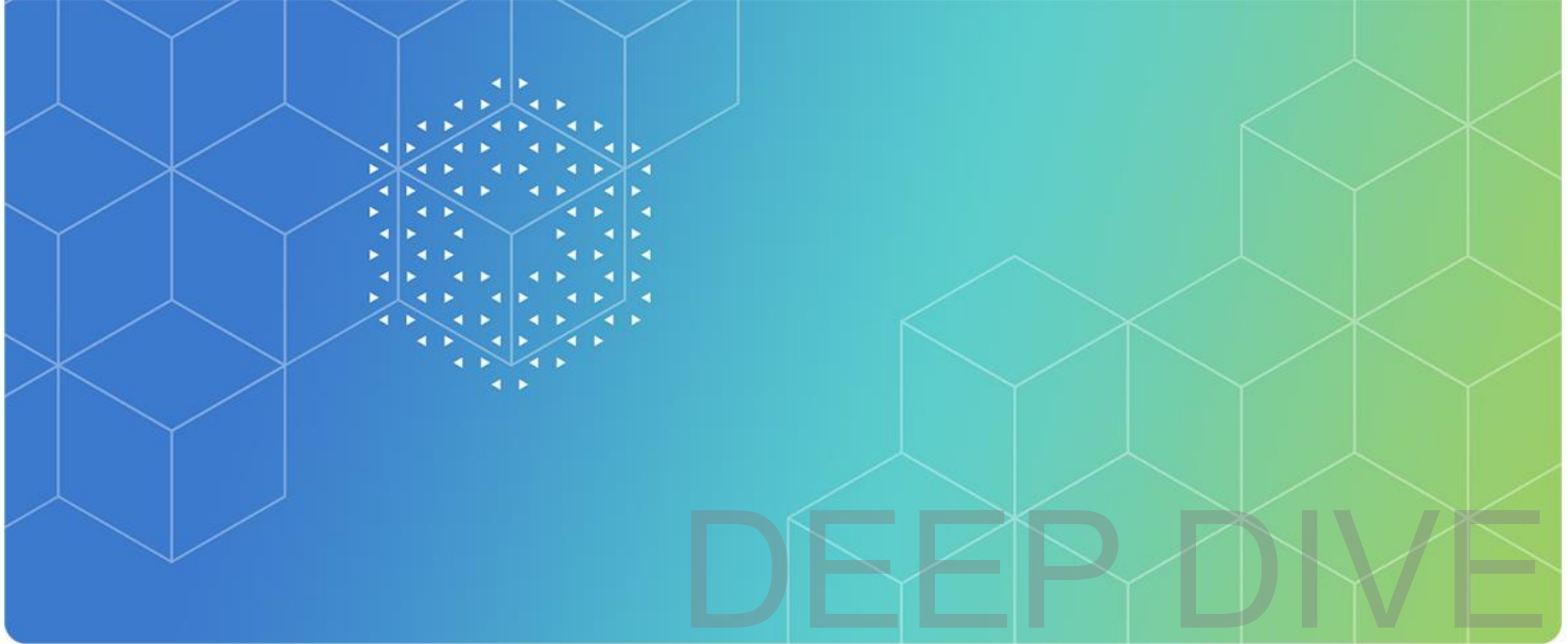
What-if Analysis: FMEDA Dynamic Configurations

- It is possible to select one parameter (e.g, DC), define the interval and an output metric to be reported
- By leveraging the USF backend Midas provides the result of the simulation
- Graphs, and values, can be saved and restored

The screenshot displays the Cadence Midas Safety Platform interface for a project named 'RISC_CORE_ARCH'. The 'SM Mappings' tab is active, showing a table of safety mechanisms and their failure modes. The table has columns for Safety Mechanism, Failure Mode, Dcp, Dct, Dcl, and Status. Two rows are visible under the SM-RF mechanism: FM_7 and FM_9, both with a Dcp value of 95 and a Status of 'On'. Below the table, the 'Shell' output shows several system messages related to USF subpart configurations.

Safety Mechanism	Failure Mode	Dcp	Dct	Dcl	Status
SM-RF	FM_7	95	95	100	On
SM-RF	FM_9	95	95	100	On
SM-IF					
SM-SR					
SM-Fetch					
SM-FPU					
SM_IF_SEC_SM					
SM-LS					
SM-DECODER					
SM-ALU					

```
Shell
: update_usf_subpart - subpart_safe to on
: Midas USF INFO
Info : USF [USF_INF-6913]
: update_usf_subpart - subpart_noeffect to off
: Midas USF INFO
Info : USF [USF_INF-6913]
: update_usf_subpart - subpart_safe to on
: Midas USF INFO
Info : USF [USF_INF-6913]
: update_usf_subpart - subpart_noeffect to off
: Midas USF INFO
Midas >
```



Detailed FMEDA



Detailed FMEDA Authoring Steps

Get Design Information (DHE)

Import Design Information

Create the FMEDA Project

Create the Technologies (Base Failure Rates)

Create Parts

Define Parts Mapping to Design

Create Subparts

Define Subparts Mapping to Design

Create the Failure Modes

Define Failure Mode Mapping to Design

Create the Safety Mechanisms

Map the Safety Mechanisms to the Failure Modes

Detailed-FMEDA Specific Steps

What-if Analysis

Queries

Generate Reports

Safety Metric Verification

Design Hierarchy Extraction

Genus

```
usf_genus_ns::usf_genus_dhe
```

```
[designInstance]  
{dheFileName}  
{ffFileName}  
[-bbox bboxFileName]  
[-seq_leaf {instances_name_list}  
-comb_leaf {instances_name_list} [-stopathier]]
```

Xcelium

```
xrun -elaborate  
-fault_mdb_gen  
[-fault_top <top_instance | top_module>]  
[-fault_mdb_file <dheDB_filename>]  
[-fault_mdb_ff]  
[-fault_lib_mfile <lib_list_file>]  
[-fault_mdb_overwrite]  
[other_options]  
<source_files>
```

Spectre Circuit Information (info)

- New keyword: what=dhe

Parameter	Description
dheminarea	Lower bound of area value for device to be considered during design hierarchy extraction
dhesubckt	Design hierarchy is generated for all instances of the specified sub-circuits
dheinst	Design hierarchy is generated for the specified sub-circuit instances
dhexsubckt	All instances of the specified sub-circuits are excluded from the design hierarchy
dhexinst	The specified sub-circuit instances are excluded from the design hierarchy
dheparams	Name of the file that provides the rules to calculate area for subcircuits when what=dhe. Area are calculated on instance parameters

```
### subckt name : area expr  
cfmcm : lr*w*nr*multi+lr*s*(nr-1)*multi  
mimcap* : lt*wt*mf  
mimcap_lp5_sin : lt*wt*mf  
nmoscap : wr*lr*multi  
nmoscap_33 : wr*lr*multi  
nmoscap_tgo5 : wr*lr*multi
```



Basic Failure Rate (BFR) Support



ISO 26262-11 / IEC TR 62380: USF Commands

MATHEMATICAL MODEL :

$$\lambda = \left[\underbrace{\left\{ \lambda_1 \times N \times e^{-0.35 \times a} + \lambda_2 \right\}}_{\lambda_{die}} \times \underbrace{\left\{ \frac{\sum_{i=1}^y (\pi_t)_i \times \tau_i}{\tau_{on} + \tau_{off}} \right\}}_{\text{die}} + \underbrace{\left\{ 2.75 \times 10^{-3} \times \pi_\alpha \times \left(\sum_{i=1}^z (\pi_n)_i \times (\Delta T_i)^{0.68} \right) \times \lambda_3 \right\}}_{\lambda_{package}} + \underbrace{\left\{ \frac{\pi_I \times \lambda_{EOS}}{\lambda_{overstress}} \right\}}_{\text{overstress}} \right] \times 10^{-9} / h$$

die
set_IEC62380_1DIE

package
set_IEC62380_1Package

overstress
set_IEC62380_loverstress

• Customizations:

- Mission Profile: set_Mission_Profile; get_Mission_Profile
- Safe/Dangerous Ratio: set_safeness; get_safeness
- Confidence Level: set_Confidence; get_confidence
- Conservative (ISO26262-11) temperature derating
- Package customizations: set_IEC62380_cppackage; get_IEC62380_cppackage

SN29500: USF Command

```
set_SN29500_lambda {lambda ref table} {technology} {size} {factors list} {power consumption}
                  {number of pins} {cooling method} {package} {mission profile}
                  {table 9 category} {voltage type} {operating voltage} {rated voltage}
                  {drift flag}
```

$$\lambda = \lambda_{ref} \times \pi_U \times \pi_T \times \pi_D$$

Analog integrated circuits with extended range of operating voltages

$$\lambda = \lambda_{ref} \times \pi_T \times \pi_D$$

Analog integrated circuits with fixed operating voltages

$$\lambda = \lambda_{ref} \times \pi_U \times \pi_T$$

Digital CMOS-B

$$\lambda = \lambda_{ref} \times \pi_T$$

For all other integrated circuits

Drift sensitivity factor
 Temperature sensitivity factor
 Voltage Dependence factor

Table 1 Failure rates for memories

Memory Type	Operating Voltage (V)										Failure Rate (FIT)
	1.5	1.8	2.5	3.0	3.3	3.6	5.0	5.5	6.0	6.5	
Bipolar RAM	50	50	-	-	-	-	-	-	-	-	75
CMOS RAM	50	50	10	10	10	10	10	10	10	10	50
DRAM	15	10	10	10	10	10	10	10	10	10	50
SRAM	30	25	15	15	15	15	15	15	15	15	50
EPROM	30	30	30	30	30	30	30	30	30	30	50
FLASH	-	30	30	30	30	30	30	30	30	30	50
EEPROM	30	30	30	30	30	30	30	30	30	30	50

Table 2 Failure rates for microprocessors and peripheral, microcontrollers and signal processors

Technology	Integration / Degree of integration						Failure Rate (FIT)
	1-5k	5k-15k	15k-50k	50k-100k	100k-500k	500k-1000k	
Bipolar	50	-	-	-	-	-	70
NMOS	50	60	-	-	-	-	90
CMOS	25	-	-	-	-	-	50
	-	30	-	-	-	-	60
	-	-	50	-	-	-	80
BICMOS	-	-	-	80	100	(150)	90
	-	-	-	50	-	-	75

Table 3 Failure rates for ASICs, ASICs and low threshold, bus and memory circuits

Technology	Integration / Degree of integration		Failure Rate (FIT)
	1-100k	100k-1000k	
Bipolar	50	100	70
CMOS	25	50	50
BICMOS	50	100	90

Table 4 Failure rates for analog integrated circuits

Technology	Integration / Degree of integration			Failure Rate (FIT)
	1-100k	100k-1000k	1000k-10000k	
Operational amplifier	3	8	12	55
Comparator	3	8	12	55
Reference elements	3	8	12	55
Switched capacitor	3	8	12	55
Logic gates	3	8	12	55
Power amplifier	3	8	12	55

Table 5 Failure rates for application-specific integrated circuits (ASIC)

Technology	Integration / Degree of integration						Failure Rate (FIT)
	1-10k	10k-100k	100k-1000k	1000k-10000k	10000k-100000k	100000k-1000000k	
ASIC	50	50	50	50	50	50	50
ASIC	50	50	50	50	50	50	50
ASIC	50	50	50	50	50	50	50
ASIC	50	50	50	50	50	50	50
ASIC	50	50	50	50	50	50	50
ASIC	50	50	50	50	50	50	50



Midas GUI BFR Tools

ISO 26262-11 / IEC TR 62380

SN29500

Design Information

Technology: **Si MOS: Linear circuits**

Number of Transistors:

Manufacturing Year:

Mission Profile: **Motor Control**

Custom Mission Profile

Temperatures:

	Tac (°C)	τ (%)
Temp. 1	45	56
Temp. 2		
Temp. 3		

Ratios on/off

T_{on}

T_{off}

2 night starts

n_1

ΔT_1

4 day light starts

n_2

ΔT_2

Non used Vehicle

n_3

ΔT_3

Technology Structure: **MOS; BiCMOS (low voltage)**

Percentage of the chip:

Package Information

Thermal Expansion α_s : **PTFE Glass (polytetrafluoroethylene)**

Thermal Expansion α_c : **Epoxy (Plastic package)**

Package Type for λ_s : **TQFP, 10x10 mm2, 40-60 pin**

Custom Package

Width: Length: Pitch:

Package Type for Thermal Resistance: **QFP plastic package**

Number of the pins of the package:

Cooling method: **Natural convection**

Power Consumption:

Interface circuits: **Non Interfaces - All electrical environment**

Calculations

λ_{die}

$\lambda_{transistor}$

$\lambda_{package}$

$\lambda_{overstress}$

Base Failure Rate Tool

Aref

Device Type:

Technology:

Size:

λ_{ref} value:

Factors

Temperature Voltage Drift Stress

Package Information

Package Type for Thermal Resistance:

Number of the pins of the package:

Cooling method:

Power Consumption:

Mission Profile:

Voltage

U:

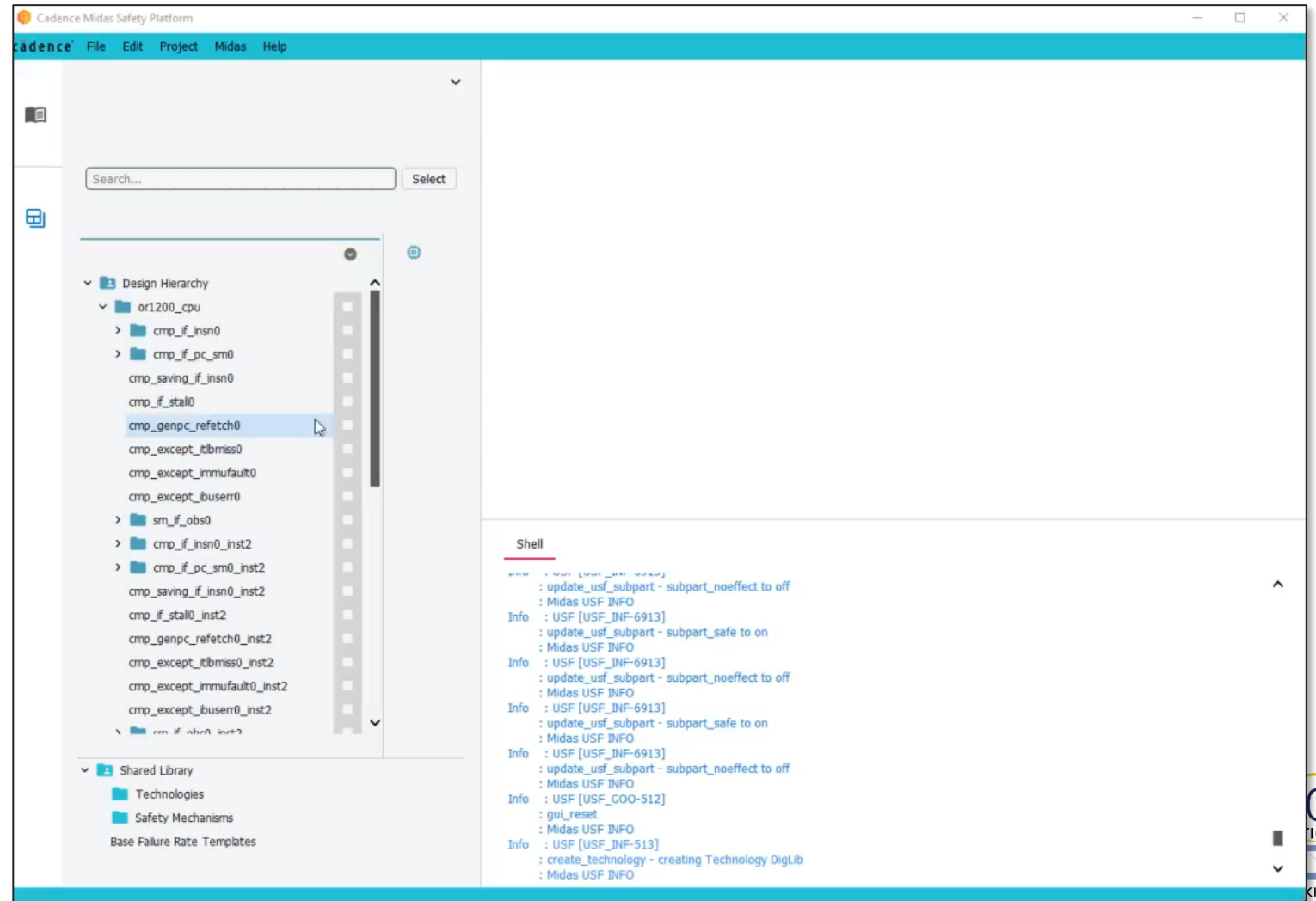
Umax:

Computation

λ

Leverage Design Information in the BFR Computation

- Create a Technology by using the IEC 62380 BFR tool with automatic computation of the number of transistors
- The technology is saved in the shared library, available for all FMEDA projects



Design Information Mapping

- Drag & drop Design information to Parts, Subparts and Failure Modes
- Area, equivalent number of gates and number of sequential elements are automatically computed

The screenshot displays the Cadence Midas Safety Platform interface for a project named 'RISC_CORE_DETAILED'. The left sidebar shows a hierarchical design tree with folders for 'FETQ', 'DECODER', 'EXCEPT', 'CORE REGS', and 'WR-BACK LOGIC'. Below this is a search bar and a 'Select' button. The main workspace is divided into two panes. The top pane shows a table of design instances with columns for 'FM ID', 'Technology', 'Design Instances', 'Exclude Instances', 'Computed Flapping', 'Area', '#Gates', and '#Flop Bits'. The bottom pane shows a 'Shell' window with system messages and warnings.

FM ID	Technology	Design Instances	Exclude Instances	Computed Flapping	Area	#Gates	#Flop Bits
FM_1	DigLib	hinst:or1200_cpu/or1200_wbmux		hinst:or1...	0.0	0.00	0
FM_10	DigLib	hinst:or1200_cpu/or1200_operandmuxes		hinst:or1...	935.4	935.37	33
FM_11	DigLib	hinst:or1200_cpu/or1200_operandmuxes		hinst:or1...	1316.4	1316.36	65
FM_12	DigLib	hinst:or1200_cpu/or1200_fpu/Sparse Logic		hinst:or1...	817.4	817.38	37
FM_13	DigLib	hinst:or1200_cpu/or1200_fpu/fpu_arith/Sparse Logic		hinst:or1...	2099.2	2099.20	150
FM_14	DigLib	hinst:or1200_cpu/or1200_fpu/fpu_arith/fpu_addsub		hinst:or1...	1242.1	1242.14	29
FM_15	DigLib	hinst:or1200_cpu/or1200_fpu/fpu_arith/fpu_div		hinst:or1...	2524.6	2524.64	135
FM_16	DigLib	hinst:or1200_cpu/or1200_fpu/fpu_arith/fpu_mul		hinst:or1...	3083.8	3083.81	152
FM_17	DigLib	hinst:or1200_cpu/or1200_fpu/fpu_arith/fpu_post_norm_div		hinst:or1...	4058.2	4058.17	237
FM_18	DigLib	hinst:or1200_cpu/or1200_fpu/fpu_arith/fpu_post_norm_mul		hinst:or1...	5721.7	5721.66	261
FM_19	DigLib	hinst:or1200_cpu/or1200_fpu/fpu_arith/fpu_postnorm_addsub		hinst:or1...	2088.6	2088.59	75
FM_2	DigLib	hinst:or1200_cpu/or1200_genpc/Sparse Logic		hinst:or1...	1079.3	1079.35	32
FM_20	DigLib	hinst:or1200_cpu/or1200_fpu/fpu_arith/fpu_pre_norm_div		hinst:or1...	1411.8	1411.78	36
FM_21	DigLib	hinst:or1200_cpu/or1200_fpu/fpu_arith/fpu_pre_norm_mul		hinst:or1...	241.8	241.79	10
FM_22	DigLib	hinst:or1200_cpu/or1200_fpu/fpu_arith/fpu_prenorm_addsub		hinst:or1...	1813.3	1813.28	78



Post-processing, Query & Reporting



Failure Mode Distribution (FMD) Post-processing

- Post-process the failure mode distribution

```
usf_set_fmd {-fmeda fmendaprj}
            [-part part_name]
            [-subpart subpart_name]
            [-permanent]
            [-transient]
            [-strategy {area_uniform | fit_constant | custom} |
             -fm fm_name [-value {0-100}]]
            [-distribution {distributions}]
            [-rounding_cost {default | cascade | sum_of_dist_diffs}]
```

- Example: custom redistribution

```
create_subpart clk_rst -part dbg -fmeda core ... -gates 21.6 -flops 3
```

```
create_failure_mode FM_1 -part dbg -subpart clk_rst -fmeda core ...
```

```
create_failure_mode FM_2 -part dbg -subpart clk_rst -fmeda core ...
```

```
usf_set_fmd -fmeda -part dbg -subpart etm_clk_rst \
            -strategy custom -distribution {FM_1 {50.0% 50.0%} FM_2 {50.0% 50.0%}}
```

Permanent

Transient

FM ID	* Part	* SubPart	Area	#Gates	#Flop Bits	#bits
FM_1	dbg	clk_rst	10.4	10.80	2	0
FM_2	dbg	clk_rst	10.4	10.80	1	0

query_usf USF Relational Queries

The **query_usf** command reports in a 'TCL friendly' format the information to create safety automations

LEVEL 0	<code>query_usf *</code>	Listing available information
LEVEL 1	<code>query_usf {fmeda} {-obj_id id} {-obj_type type}</code>	Direct query
LEVEL 2	<code>query_usf {fmeda} {-obj_id id} {-obj_type type} [-ref_type RefType] [-ref_id refid]</code>	By referencing another object

- How many FMEDA projects do we have?

```
- query_usf *  
  - FMEDAPRJ FMEDA_OpenRisc
```

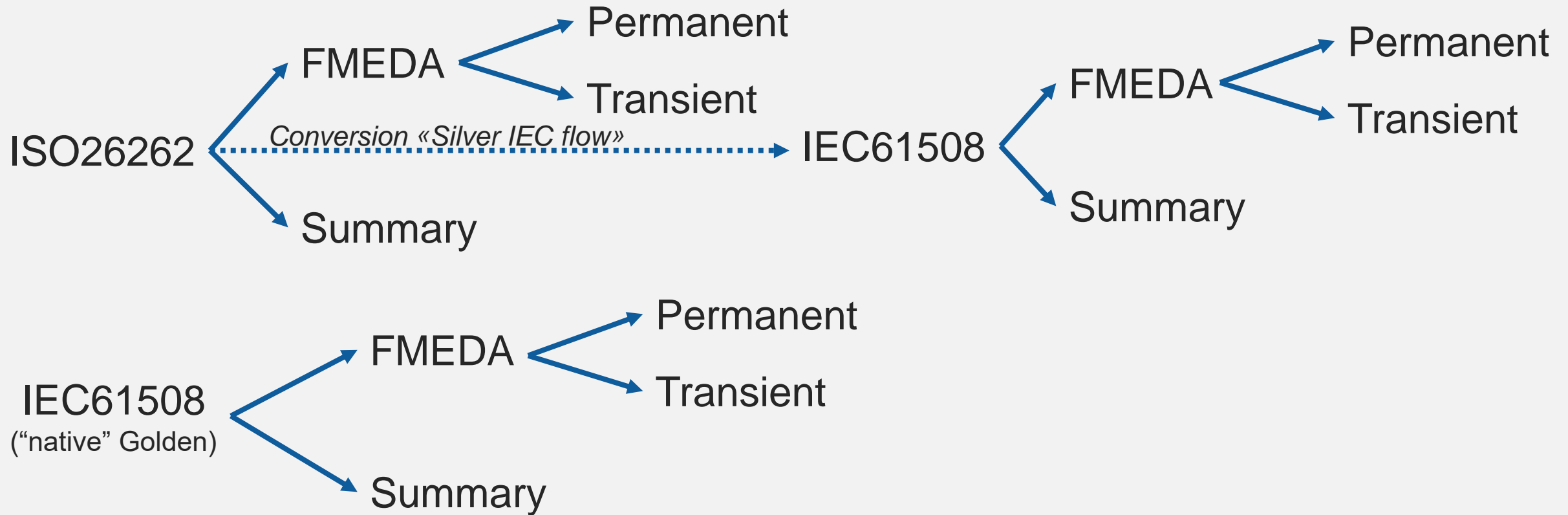
- How many Failure Modes have been defined for this project?

```
- query_usf FMEDA_OpenRisc -obj_type failure_mode -obj_id *  
  - FAILUREMODES FM_1 FM_2 FM_3 FM_4 FM_5 FM_6 FM_7 FM_8 FM_9 FM_10 FM_11 FM_12 FM_13 FM_14  
    FM_15 FM_16 FM_17 FM_18 FM_19 FM_20 FM_21 FM_22 FM_23 FM_24 FM_25 FM_26 FM_27 FM_28  
    FM_29 FM_30 FM_31 FM_32 FM_33 FM_34 FM_35
```

- Report the metrics for a specific FMEDA project

```
- query_usf FMEDA_OpenRisc -obj_type fmeda -obj_id metrics  
  - FMEDAPRJ FMEDA_OpenRisc off on on B off on {9 16 35} {57.5% 58.1% 100.0%} {4.269e-02  
    6.753e-02 0.000e+00 1.005e-01 1.611e-01} DigLib {{134678.6 131265.7 6563.0} {98720.7  
    96219.0 4431.0} {96364.7 93922.7 4328.0}} {57.52% -- -- -- --} {100.00% -- -- -- --}  
    {58.09% -- -- --}
```


USF Reports: ISO26262 and IEC61508

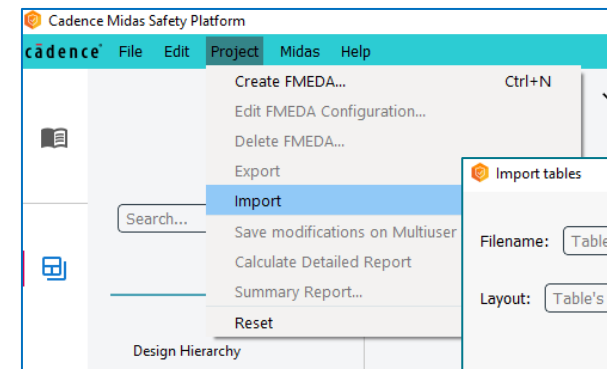
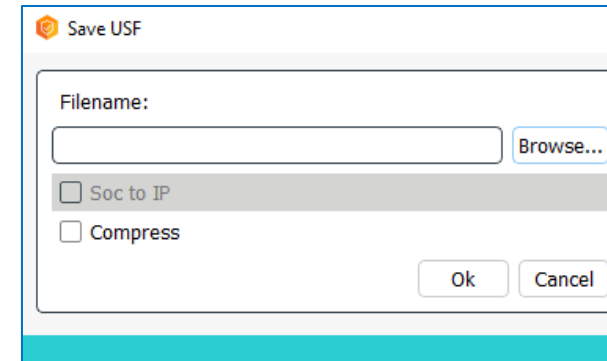
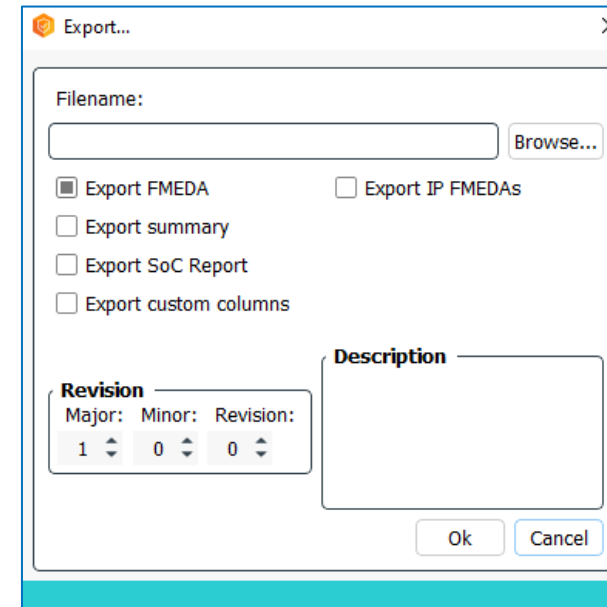


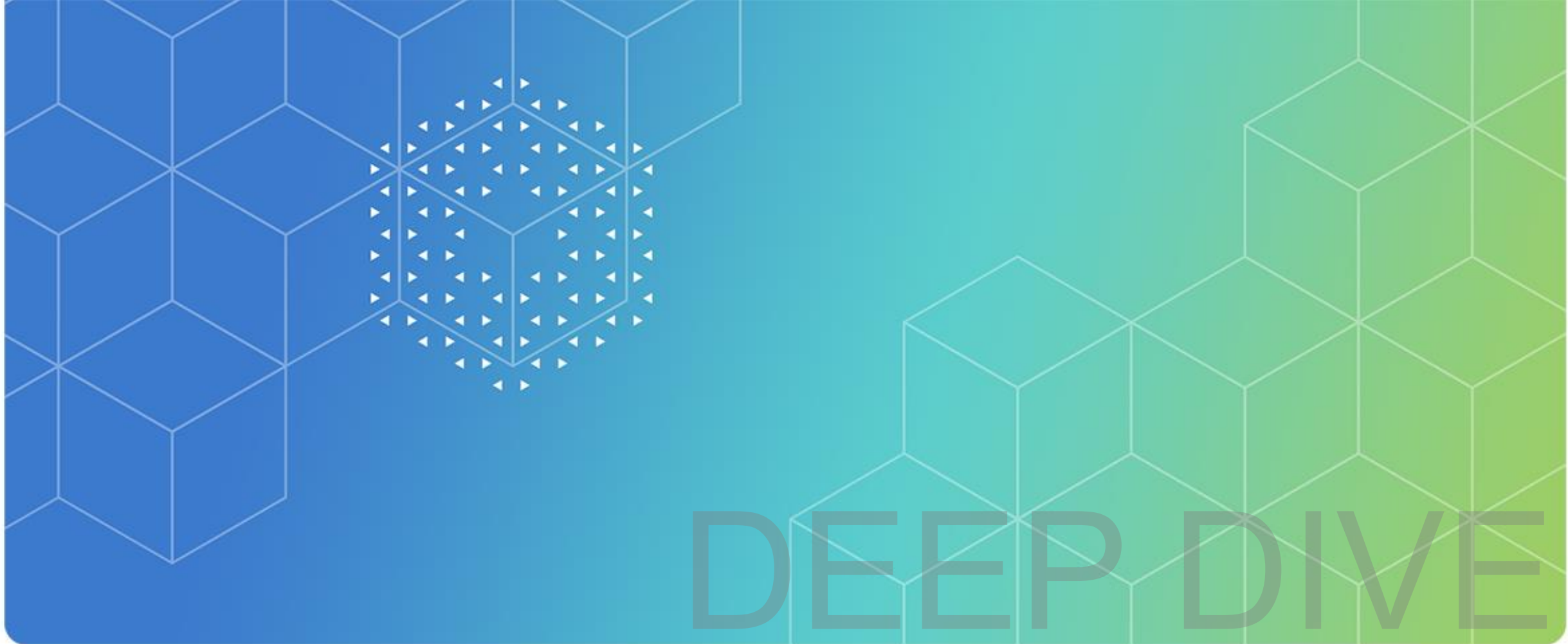
```
report_safety -fmeda myFMEDA permanent html "reports/ISO_PERMANENT.html"  
report_safety -fmeda myFMEDA transient html "reports/ISO_TRANSIENT.html"  
report_safety -fmeda myFMEDA report html "reports/ISO_SUMMARY.html"
```

```
report_safety -standard iec61508 -fmeda myFMEDA permanent html "reports/IEC_PERMANENT.html"  
report_safety -standard iec61508 -fmeda myFMEDA transient html "reports/IEC_TRANSIENT.html"  
report_safety -standard iec61508 -fmeda myFMEDA report html "reports/IEC_SUMMARY.html"
```

Midas Application Import-Export

- Rationales
 - Use USF (text file) for exchange/integration
 - Use Spreadsheets for final reporting and auditing
- Microsoft Excel import/export is supported



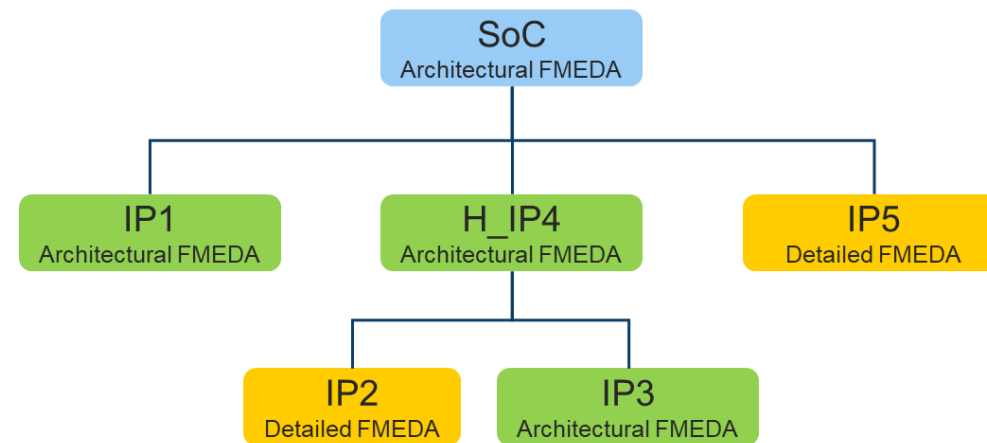


SoC Safety Analysis



SoC Safety Analysis Integration

- SoC metrics are calculated combining (grouping) IP FMEDAs
- IP FMEDA work is partitioned, the owner of the overall safety analysis is grouping the IP FMEDAs into a SoC FMEDA
- Multiple levels of hierarchy are supported
- Combination of detailed and architectural FMEDA is possible
- Keep the details in the IP FMEDAs but keep SoC FMEDA as simple as possible
- Propagation and combination of Safety Goals (aka Failure Mode Effect)
- Ability to support weights of Failure Modes to different Safety Goals



```
# FMEDA 1
usf_reset
set troot1 {...}

load_usf [file join $troot1 "arm_cortex_m7_fmEDA.usf"]
save_usf [file join $troot1 IP_USF "fmEDA_1.usf"] -compress
set fmEDA1 [lreplace [query_usf *] 0 0]

# FMEDA 2
usf_reset
set troot2 {...}
load_usf [file join $troot2 "dtmf.usf"]
save_usf [file join $troot2 IP_USF "fmEDA_2.usf"] -compress
set fmEDA2 [lreplace [query_usf *] 0 0]

# FMEDA ...

# Create SoC and group IP FMEDA
usf_reset
set_fmEDA SOC -soc -ASIL B -permanent -transient -architectural
group_fmEDA -fmEDA_list [list $fmEDA1 $fmEDA2] \
             -fmEDA_file [list [file join $troot1 IP_USF "fmEDA_1.usf"] \
                               [file join $troot2 IP_USF "fmEDA_2.usf"]] -to SOC
```

Grouping IP FMEDAs into a SoC FMEDA: USF Command

<code>group_fmEDA</code> {-fmEDA_list} [-fmEDA_file] {-to fmEDA_soc} [-linkonly]	
<code>-fmEDA_list</code> FMEDA_tags_list	Specify a list of FMEDA to link to a SoC FMEDA. With the format <code>FMEDAIP(num_replica)</code> , automatically creates replicas of the same FMEDA
<code>-fmEDA_file</code> FMEDA_files_list	Optional. Specify a list of FMEDA project files to link to an SoC FMEDA. The files are assumed to be generated using <code>save_usf</code> commands.
<code>-to</code> fmEDA_soc	Specify that the SoC FMEDA is used as a reference for the FMEDA project. The SoC FMEDA must be previously created with the <code>set_fmEDA</code> command using the <code>-soc</code> option.
<code>-linkonly</code>	Optional. Link an IP FMEDA to the SoC FMEDA without copying parts, subparts, and failure modes

Examples

- `group_fmEDA -fmEDA_list {myFMEDA1 myFMEDA2} -to mySOCFMEDA`
- `group_fmEDA -fmEDA_list {myFMEDA1 myFMEDA2} -fmEDA_file {myFMEDA1.usf myFMEDA2.usf} -to mySOCFMEDA`

SoC FMEDA Project: Midas Application

The screenshot shows the Cadence Midas Safety Platform interface for a SoC project. On the left, a file tree is visible, listing various FMEDA items under the 'SOC' folder. On the right, a summary table is displayed, showing the following data:

ID	Part count	SubPart count	Failure Mode count	SPFMp	SPFMT	LFM
SOC	10	31	53	96.66%	97.73%	94.41%
FMEDA_OpenRisc	9	16	35	57.52%	58.09%	100.00%
USF_RAK	1	15	18	98.53%	98.19%	94.25%

Grouping IP FMEDAs into a SoC FMEDA

- Safety Hierarchy
- SoC Summary

`report_safety -fmeda SoC soc.html SoC_soc.html`

Safety Goals (aka Failure Mode Effects, High Level Failure Modes)

- Can be used to track the metrics of a list of failure modes of a given IP FMEDA

```
create_safety_goal SG_1 -description "My safety goal 1" -fmeda "FMEDA_DTFM" \
    -fm_list {FM_TDSP}
create_safety_goal SG_2 -description "My safety goal 2" -fmeda "FMEDA_DTFM" \
    -fm_list {FM_GROUPED FM_CONV_INST}
```

ID	Part	SubPart	Failure Mode	Safety Releva	FM Type	Techno logy	Area	#Gates	#Flop Bits	#bit	Raw Permanent	Total Safety	F _{SAFE} (p) %	Fail rate Safe Fault	Fail rate non-Safe	λ(p) %	K _{RF} (p) %	Single Point	SG_1	SG_2
FM_ROM	TOP	MYRO	ROMFM	Yes	Mission	ROMLi	0	0	0	0	0.00E+00	0.00E+00	0.00%	0.00E+00	0.00E+00	0.00%	0.00%	0.00E+00		
FM_RAM	TOP	MYRAM	RAMFM	Yes	Mission	RAMLi	210487	0	0	8192	6.55E-02	6.55E-02	0.00%	0.00E+00	6.55E-02	98.83%	0.00%	6.55E-02		
FM_TDSP	TOP	TDSP	TDSP_CORE_INST FM	Yes	Mission	DigLib	6488.5	6488.53	256	0	4.54E-04	4.54E-04	0.00%	0.00E+00	4.54E-04	0.68%	0.00%	4.54E-04	X	
FM_CONV_INST	TOP	CONV_	RESULTS_CONV_INST	Yes	Mission	DigLib	3716.2	3716.17	199	0	2.60E-04	2.60E-04	0.00%	0.00E+00	2.60E-04	0.39%	0.00%	2.60E-04		X
FM_GROUPED	TOP	GROUP	BASKET FM	Yes	Mission	DigLib	924.4	924.43	62	0	6.47E-05	6.47E-05	0.00%	0.00E+00	6.47E-05	0.10%	0.00%	6.47E-05		X

- It is possible to create SoC Safety Goals linked to IPs Safety Goals

```
create_safety_goal SGTOP -description "My new safety goal" -fmeda FMEDA_SOC \
    -sg_list SG_1
```

- It is possible to export the Safety Goals metrics into a report

```
report_safety -fmeda FMEDA_DTFM safety_goal.html "fmeda_sg.html"
```

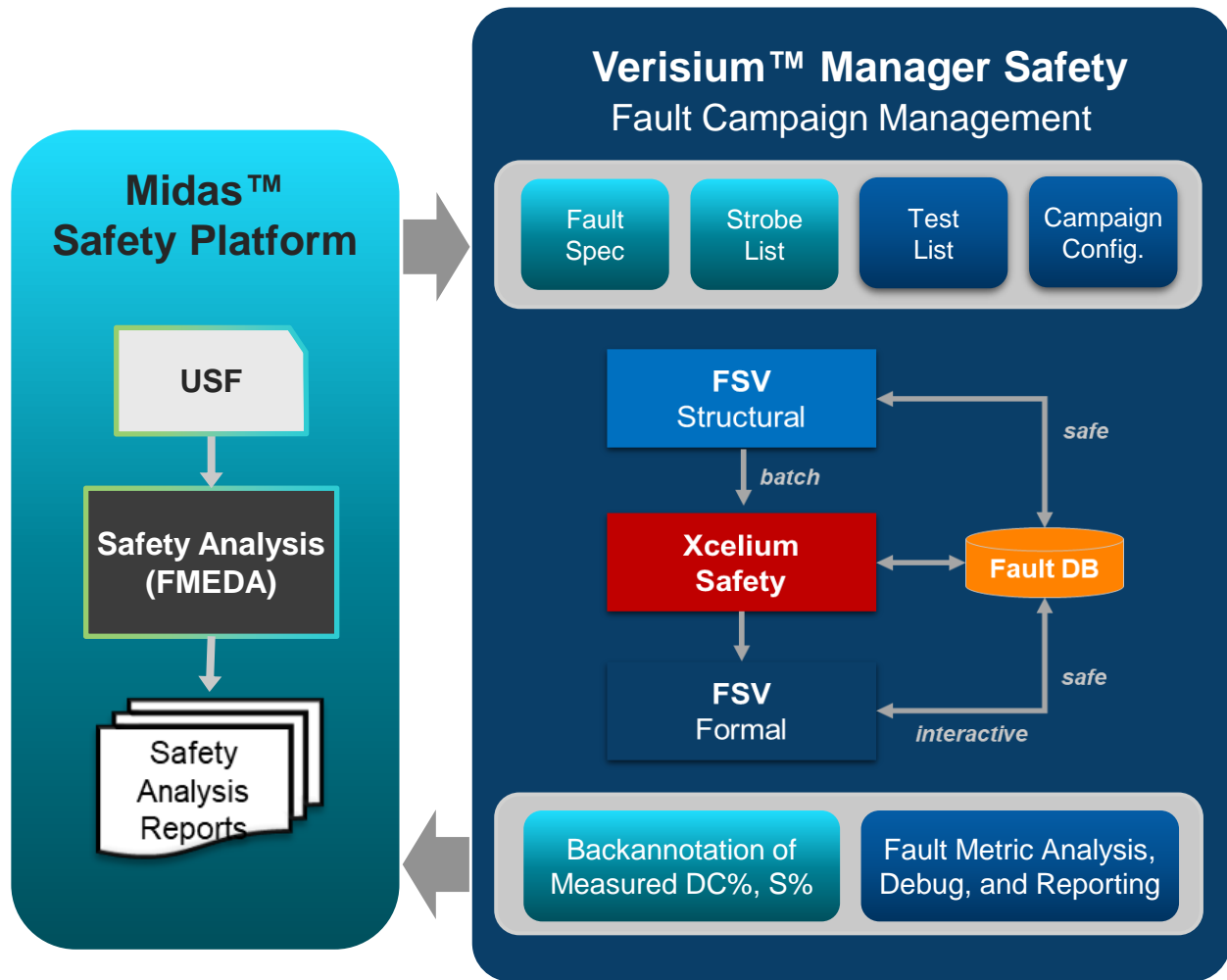
SG ID	FMEDA	Safety Goal Violations	SPFMp	SPFMt	LFM	PMHFp	PMHFp%	PMHFt	PMHFt%	Design Failure Rate Permanent (FIT)	Design FITp%	Design Failure Rate Transient (FIT)	Design FITt%
SG_1	FMEDA_DTFM	My safety goal 1	0.00%	0.00%	--	4.542e-04	58.3%	1.106e-02	57.9%	4.542e-04	58.3%	1.106e-02	57.9%
SG_2	FMEDA_DTFM	My safety goal 2	0.00%	0.00%	--	3.248e-04	41.7%	8.039e-03	42.1%	3.248e-04	41.7%	8.039e-03	42.1%



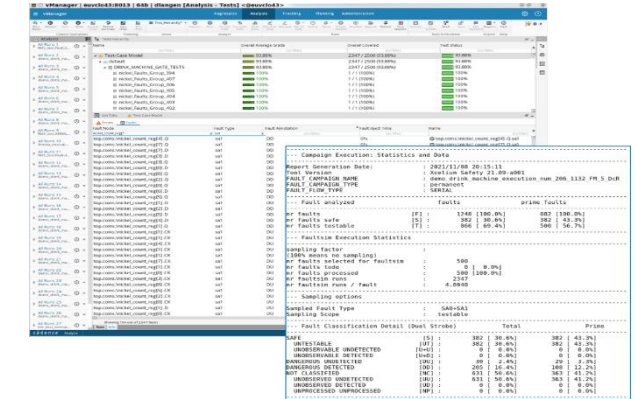


Safety Metrics Verification

Fault Campaign Management – Automation & Optimization

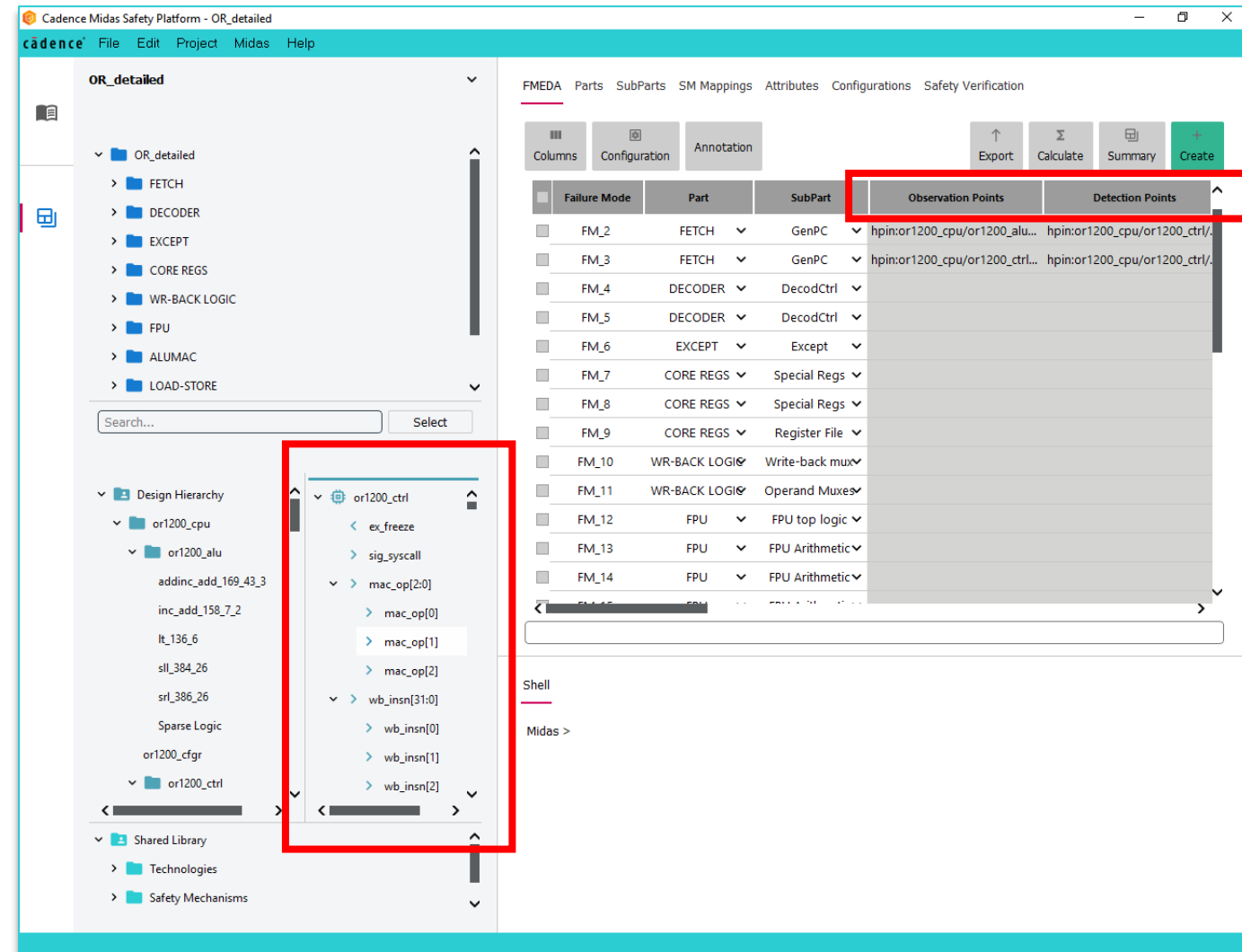


- Test selection and ranking
 - Coverage-based test selection
 - Customizable ranking criteria
- Fault list reduction
 - Fault sampling
 - Fault collapsing
 - Testability analysis
 - Test Dropping
- Fault campaigns execution
 - Measured Diagnostic Coverage and Safeness
 - Backannotation of results to FMEDA
 - Generate reports and analyze fault metric
 - FMEDA, fault classification, campaign summary,...



Strobing Points Definition

- Strobing points can be dragged & dropped from the design hierarchy into the related fields of the FMEDA
- The operation can be scripted



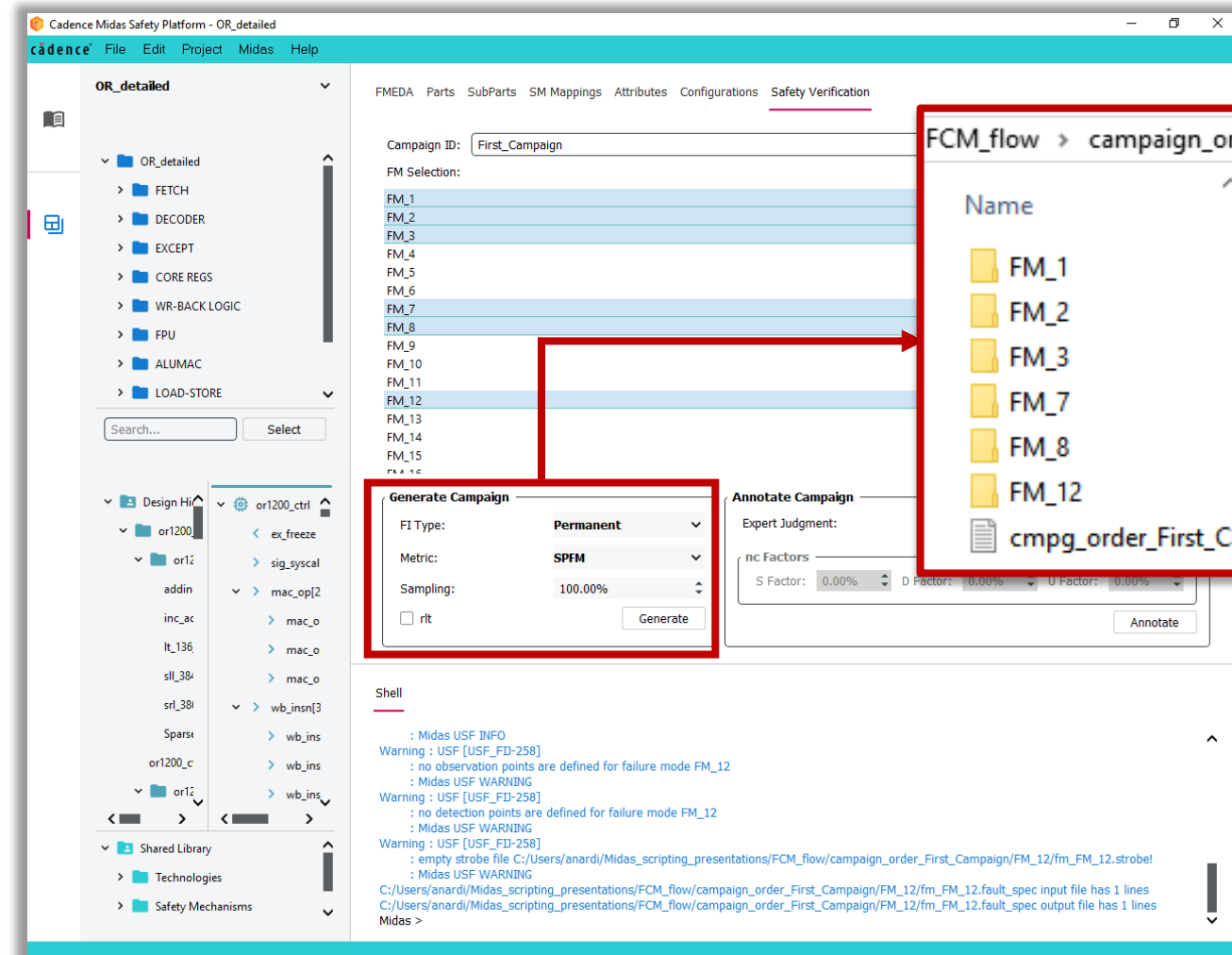
```
create_failure_mode ... [-observation_points obspoints] [-diagnostic_points detpoints]
```

```
apply_safety_mechanism <SM_TAG> {-to FM_TAG} [-dcperm DCp_VAL] [-dctrans DCt_VAL] [-dclat DCI_VAL] \  
{-fmeda FMEDA_TAG} [-diagnostic_points detpoints]
```

Driving Fault Simulation Campaign for DC Validation

Fault Injection Campaign Order Generation

- Generation of the campaign order
 - Summary of the Fault Injection Campaign
 - Fault specification file
 - Strobe specification
 - Verisium Manager configuration



```
usf_campaign_order CAMPAIGN_FI1 -fmeda MYFMEDA -fm {I2CO_FM_1 I2CO_FM_2} \  
-p -generate -sampling 100 -spfm
```

Back-annotation of the Fault Injection Campaign Results

FMEDA Parts SubParts SM Mappings Attributes Configurations **Safety Verification**

Campaign ID:

FM Selection:

- FM_1
- FM_2
- FM_3
- FM_4
- FM_5
- FM_6
- FM_7
- FM_8
- FM_9
- FM_10
- FM_11
- FM_12
- FM_13
- FM_14
- FM_15

Generate Campaign

FI Type: **Permanent**

Metric: **SPFM**

Sampling: **100.00%**

rit

Annotate Campaign

Expert Judgment: **standard**

nc Factors

S Factor: **0.00%** D Factor: **0.00%** U Factor: **0.00%**

FMEDA Parts SubParts SM Mappings Attributes Configurations Safety Verification

Columns Configuration **Annotation** Export Calculate Summary Create

Show estimated values
Show annotated values

Failure Mode	Observed	$\lambda(p)$	$\lambda(p)\%$	Krf(p)%
FM_1		8.107e-04	1.45%	63.00%
FM_10		5.956e-04	--	--
FM_11		8.200e-04	--	--
FM_12		5.100e-04	0.88%	70.00%
FM_13		1.148e-03	1.99%	70.00%
FM_14		8.068e-04	1.40%	70.00%
FM_15		1.486e-03	2.57%	70.00%
FM_16		1.760e-03	3.04%	70.00%
FM_17		2.349e-03	4.06%	70.00%

Shell

Info : USF [USF_FD-514]
: FM_1: Fault simulation results (S DU DD NC UO NPNS TOT UNUN UNDE): 5 37 63 0 0 0 35732 0 0 2
: Midas USF INFO
Info : USF [USF_FD-514]
: Expert judgment (standard): DC(63.00) FSafe(4.76)
: Midas USF INFO

```
usf_campaign_order CAMPAIGN_FI1 -fmeda MYFMEDA -fm {I2C0_FM_1 I2C0_FM_2 I2C0_FM_3 I2C0_FM_4} -p -annotate \
-expert standard
```

Generate Final Reports

- Once annotated, both estimated and measured values are available
- Switch between the two modes and generate reports
- Save and restore

The screenshot displays the Cadence Midas Safety Platform interface for a project named 'RISC_CORE_DETAILED'. The main window shows a table of FMEDA data with columns for FM ID, Krff(p), Single Point Fault λ spf(p), Residual Fault failure rate λ rf(p), and DC Algo. The 'Annotation' button in the top right is highlighted with a red box. The 'Krff(p)' column contains values such as 100.00%, 90.00%, 98.00%, 80.00%, 80.00%, --, 95.00%, 95.00%, 99.99%, --, 90.00%, 90.00%, 90.00%, and 90.00%. The '99.99%' value is also highlighted with a red box.

A 'Summary Report' window is overlaid on the main window, titled 'Summary of the RISC_CORE_DETAILED FMEDA'. It includes the text 'Using annotated values' and 'Generated on: Sun Jan 14 22:58:07 2024'. The report contains the following data:

Metric	Value
Safety related FIT Transient - Asr	6.714e+00
Probabilistic Metric for random Hardware Failures PMHF Permanent - in FIT	6.132e-03
Probabilistic Metric for random Hardware Failures PMHF Transient - in FIT	5.027e-01
Probabilistic Metric for random Hardware Failures PMHF Latent - in FIT	1.623e-05
Single Point Fault Metric - SPFM Permanent	93.79%
Single Point Fault Metric - SPFM Transient	92.51%
Latent Fault Metric - LFM	99.98%
Total Not Safety Related faults Permanent - AnSR	7.040e-03

```
query_usf RISC_CORE_DETAILED -obj_type fmeda -obj_id metrics -annotate
```

```
report_safety -fmeda RISC_CORE_DETAILED report html summary.html -annotate
```



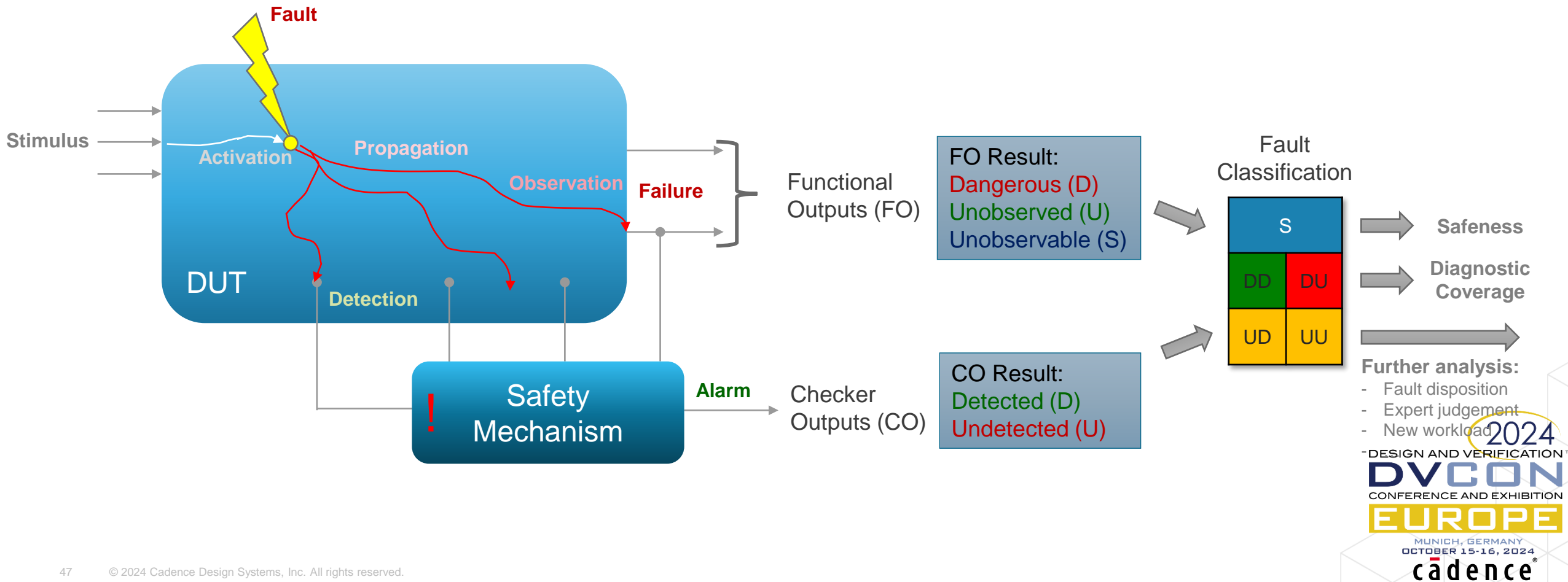
USF-based FMEDA-driven Functional Safety Verification

Fault Campaign Management (Verisium Manager Safety + Xcelium Safety + Jasper Safety)

Frederico Ferlini, Sr Principal Product Engineer, SVG

Automotive / Functional Safety / Random Faults / ...

- Goal: prevent or mitigate the effect of a hazardous event due to (operational) **random faults**
- Requirement: deliver **diagnostic coverage** according to ASIL (Automotive Safety Integrity Level)
- Method: integrate **safety mechanisms** across the system architecture
- Validation: show evidence and assess robustness via **fault injection**



Digital Safety Verification

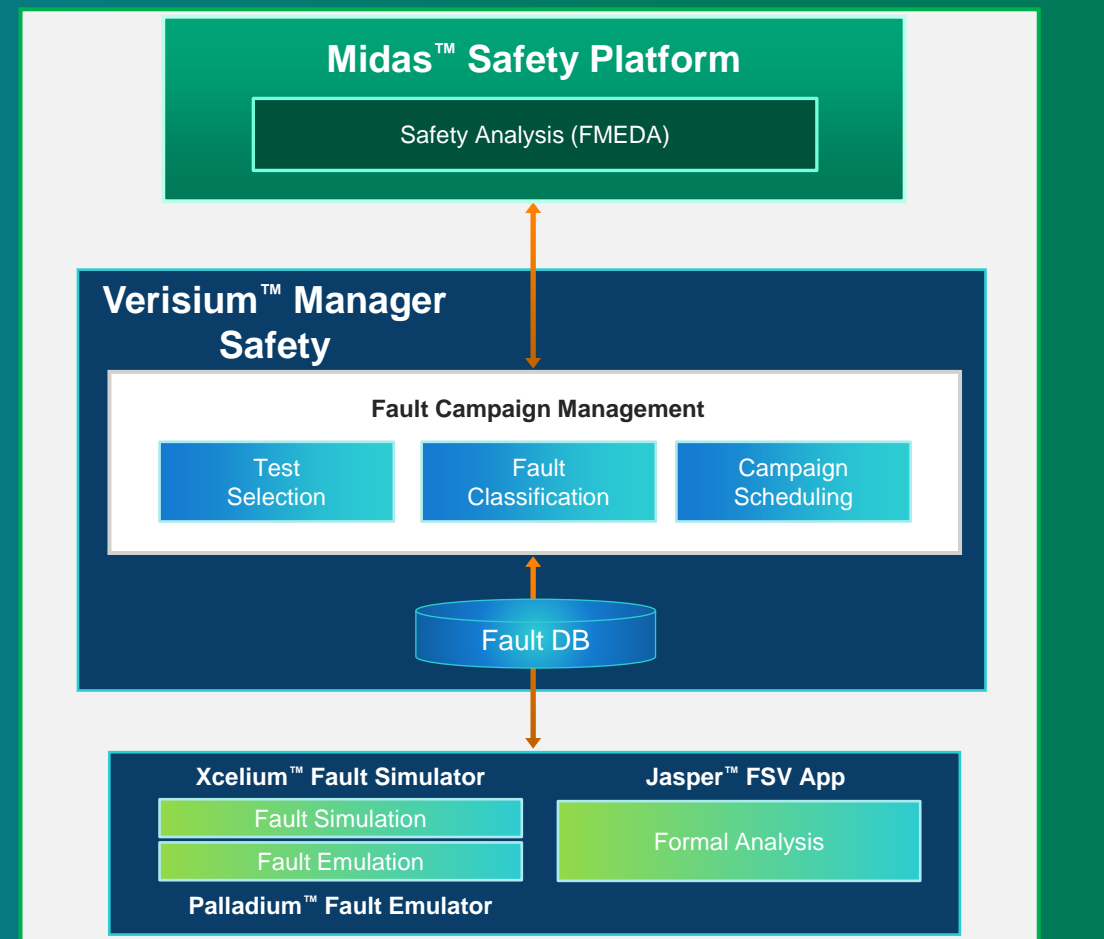
FMEDA-driven safety verification

Campaign Automation – Verisium™ Manager Safety
Unified front-end to manage all engines and analyze results
Validation and FMEDA back-annotation of diagnostic coverage

Complexity Reduction – Jasper™ FSV App
Applies industry-leading formal techniques to fault analysis
Increases safety verification performance

Injection Engine – Xcelium™ Fault Simulator
Native serial and concurrent fault simulation engine

Acceleration – Palladium™ Fault Emulator
Seamless define faults and strobe as for simulation





Verisium™ Manager Safety

Fault Campaign Manager – FCM

Fault Campaign Automation and Analysis

Verisium Manager Safety Fault Campaign Manager - FCM

1. Prepare Data

- Single front-end campaign configuration
- Expand fault targets & instrument design
- Translate strobe definition

2. Minimize Fault Set

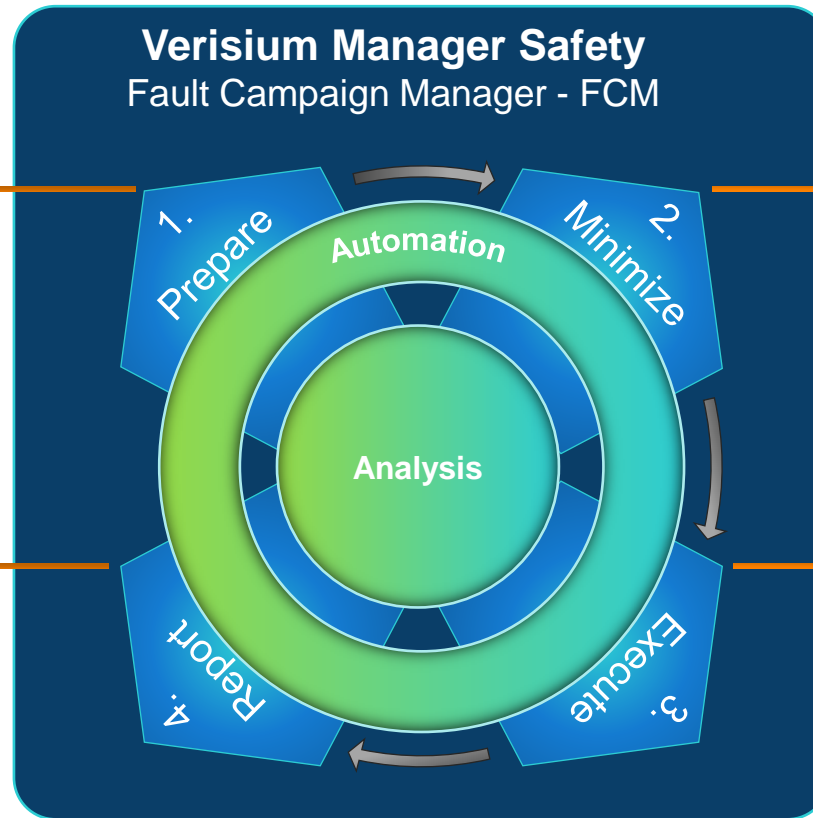
- Collapse redundant faults
- Identify unobservable/safe faults
- Test-based fault pruning

4. Generate Reports

- Campaign summary report
- Diagnostic Coverage / Safeness
- FMEDA validated results

3. Execute Campaign

- Create runs per fault groups
- Verisium Manager state-of-the-art DRM
- Drop exhausted faults/tests



Fault Metric Analysis

- Merge fault results across different campaigns
- Disposition of not-classified faults
- Offer insights towards analysis closure

Prepare

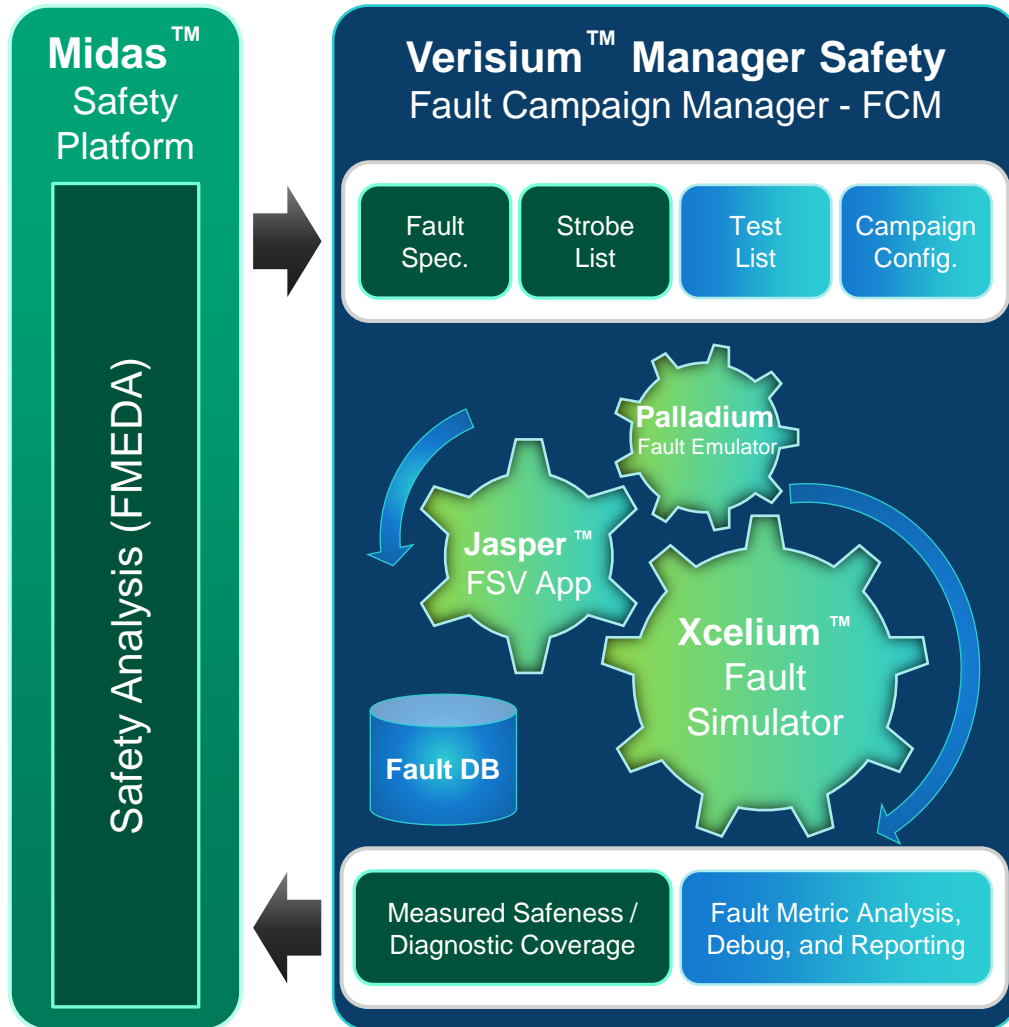
Minimize

Execute

Report

Analyze

FMEDA-driven Fault Campaign



• Inputs

○ Safety Engineer

- Fault Targets (derived from FMEDA ⇔ design mapping)
- Strobe List (observation and detection points)

○ Verification Engineer

- Test List (selected for fault analysis)
- Campaign Configuration
 - Optimizations, runs distribution, customization, etc.

• Outputs

○ Summary Report

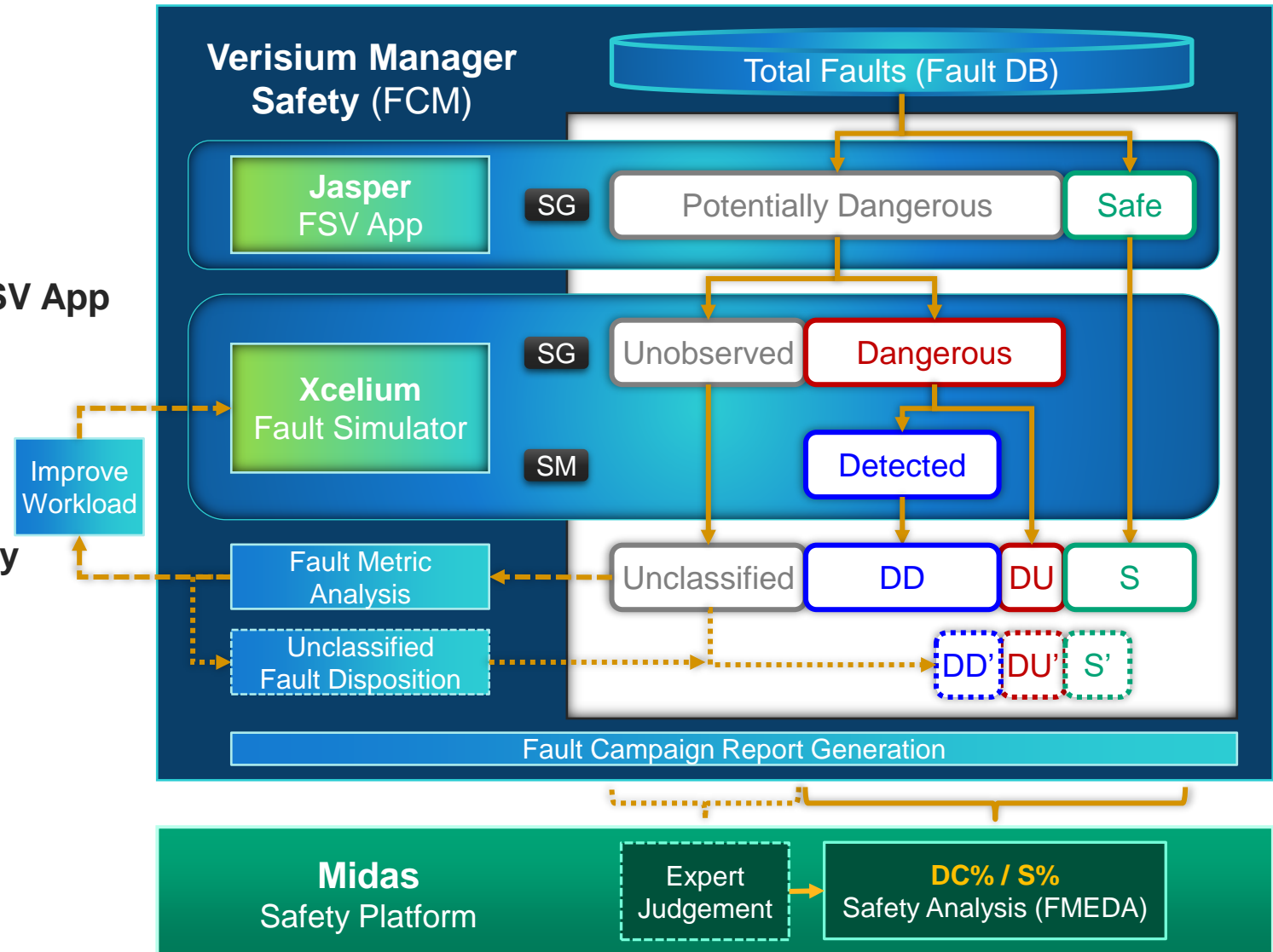
- Measured Fault/Diagnostic Coverage, Safeness

○ Fault Annotation

- Fault Metric Analysis, annotated fault list, ...

Fault Classification

- **Safeness (S%)**
 - Unable to violate Safety Goal (SG)
 - Exhaustive fault analysis with **Jasper FSV App**
- **Diagnostic Coverage (DC%)**
 - Safety Mechanism (SM) performance
 - Simulation evidence with **Xcelium Safety**
 - Dangerous faults Detection (DD)
- **Closure**
 - Dedicated fault metric analysis
 - Insight for Workload/SM improvements
 - Disposition of the Not Classified faults



Fault Campaign Steps

Automation

Runs

Index	Name	Status	Duration (sec.)
1	/flow_steps/prep	passed	16
2	/flow_steps/o_exec	passed	151
3	/flow_steps/o_rank	passed	40
4	/flow_steps/g_elab	passed	153
5	/flow_steps/fst	passed	157
6	/flow_steps/g_sim	passed	338
7	/flow_steps/fsv_tc	passed	156
8	/flow_steps/f_exec_c	passed	332
9	/flow_steps/f_exec	passed	281
10	/flow_steps/f_rprt	passed	37

Campaign Steps

Verisium Manager | sjcvl-safety:44001 | 64b | ferlini [Regression Center] (on sjfdcl1008)

Regression Analysis Planning Composer (Beta) Tracking

My_Flows* Launch Import Launch Safety Collect Runs Refresh Scripts Manager MyAction Export Export Merge Stop Stop Auto. Suspend Resume Set as completed Delete Relocate Open dir Session Info Metrics

Views Global Operations Scripts Sessions

Flow Sessions

Session Status	Name	Total Runs	#Passed	#Failed	#Running	#Waiting	#Other	Start Time	Owner	Is Sflow
completed	OR_GLS_HYB.HYBRID.ferlini.2023_09_0...	10	10	0	0	0	0	9/7/23 2:22 AM	ferlini	TRUE

Showing 1 out of 205 items

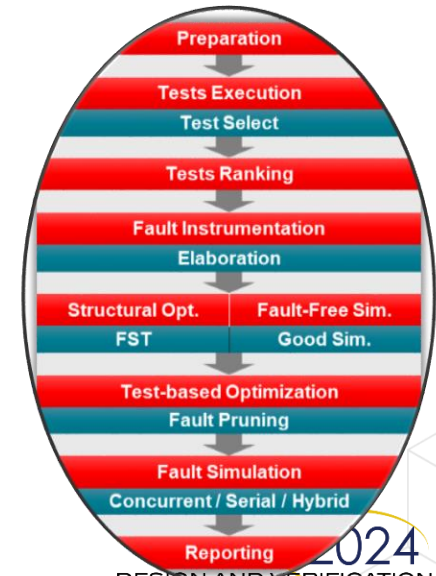
Sessions OR_GLS_HYB.HYBRID.ferlini.2023_09_07_02_22_19

Session Status	Name	Total Runs	#Passed	#Failed	#Waiting	#Running	#Other	Start Time	Owner
completed	OR_GLS_HYB.test_select.ferlini.2023_09_0...	3	3	0	0	0	0	9/7/23 2:23 AM	ferlini
completed	OR_GLS_HYB.elaboration.ferlini.2023_09_...	2	2	0	0	0	0	9/7/23 2:27 AM	ferlini
completed	OR_GLS_HYB.fst.ferlini.2023_09_07_02_29...	1	1	0	0	0	0	9/7/23 2:29 AM	ferlini
completed	OR_GLS_HYB.good_simulation.ferlini.2023...	2	2	0	0	0	0	9/7/23 2:29 AM	ferlini
completed	OR_GLS_HYB.fault_pruning.ferlini.2023_0...	2	2	0	0	0	0	9/7/23 2:35 AM	ferlini
completed	OR_GLS_HYB.fault.ferlini.2023_09_07_02_...	6	6	0	0	0	0	9/7/23 2:38 AM	ferlini
completed	OR_GLS_HYB.fault.ferlini.2023_09_07_02_...	94	54	0	0	0	40	9/7/23 2:43 AM	ferlini

Showing 7 items

cadence Regression Center Messages

Campaign Sub-Sessions



Dedicated Fault Analysis

Hierarchical Data



Verisium Manager | sjcvl-safety:44001 | 64b | ferlini [Analysis Center] (on sjfdcl1008)

Verisium Manager Regression **Analysis** Planning Composer (Beta) Tracking

Views Context Operations Scripts Planning Analyze Refinement Refinement F Safety Report Help

Verification Hierarchy

default

Name	Fault Node Detected Grade	Fault Node Total	Fault Sample Set Total	Fault NP	Fault S	Fault DD	Fault DU	Fault UD	Fault UU	Fz
(no filter)	!=-1.0	(no filter)	(no filter)	(no filter)	(no filter)	(no filter)	(no filter)	(no filter)	(no filter)	=
Verification Metrics	19.18%	3508	2184 / 3508 (62.26%)	0 / 2184 (0%)	0 / 2184 (0%)	25 / 2184 (1.14%)	13 / 2184 (0.6%)	648 / 2184 (29.67%)	1006 / 2184 (46.06%)	C
Types	n/a	0	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)	C
Instances	19.18%	3508	2184 / 3508 (62.26%)	0 / 2184 (0%)	0 / 2184 (0%)	25 / 2184 (1.14%)	13 / 2184 (0.6%)	648 / 2184 (29.67%)	1006 / 2184 (46.06%)	C
test_drink	19.18%	3508	2184 / 3508 (62.26%)	0 / 2184 (0%)	0 / 2184 (0%)	25 / 2184 (1.14%)	13 / 2184 (0.6%)	648 / 2184 (29.67%)	1006 / 2184 (46.06%)	C
top	19.18%	3508	2184 / 3508 (62.26%)	0 / 2184 (0%)	0 / 2184 (0%)	25 / 2184 (1.14%)	13 / 2184 (0.6%)	648 / 2184 (29.67%)	1006 / 2184 (46.06%)	C
coins	2.08%	866	866 / 866 (100%)	0 / 866 (0%)	0 / 866 (0%)	18 / 866 (2.08%)	0 / 866 (0%)	0 / 866 (0%)	533 / 866 (61.55%)	C
coins1	0%	866	0 / 866 (0%)	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)	C
diag	0%	160	84 / 160 (52.5%)	0 / 84 (0%)	0 / 84 (0%)	0 / 84 (0%)	13 / 84 (15.48%)	0 / 84 (0%)	50 / 84 (59.52%)	C
drinks	1.83%	382	382 / 382 (100%)	0 / 382 (0%)	0 / 382 (0%)	7 / 382 (1.83%)	0 / 382 (0%)	0 / 382 (0%)	231 / 382 (60.47%)	C
drinks1	0%	382	0 / 382 (0%)	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)	C
vending1	76.76%	284	284 / 284 (100%)	0 / 284 (0%)	0 / 284 (0%)	0 / 284 (0%)	0 / 284 (0%)	218 / 284 (76.76%)	63 / 284 (22.18%)	C
vending2	76.41%	284	284 / 284 (100%)	0 / 284 (0%)	0 / 284 (0%)	0 / 284 (0%)	0 / 284 (0%)	217 / 284 (76.41%)	63 / 284 (22.18%)	C
vending3	75%	284	284 / 284 (100%)	0 / 284 (0%)	0 / 284 (0%)	0 / 284 (0%)	0 / 284 (0%)	213 / 284 (75%)	66 / 284 (23.24%)	C

cadence Analysis Center Messages



Campaign Summary Report

Date, tool version, fault types, sampling, ...

```

---- CAMPAIGN : new_rprt_sample , permanent , CONCURRENT ----
Report Date      : 2023/02/13 02:49:17
Tool Version     : Xcelium 22.09-s005 , Verisium Manager 22.09-s002
Fault Types      : SA0+SA1
Sampling         : 50.00% of testable faults
    
```

Static instrumentation fault results

```

---- SUMMARY ----

```

	Total Faults		Total Prime		Sample Faults		Sample Prime	
----- INSTRUMENTATION -----	#	%	#	%	#	%	#	%
Faults	2626		2546		484		465	
Safe	1658	63.14	1658	65.12	0	0.00	0	0.00
Not Injected	479	18.24	458	17.99	35	7.23	35	7.53
Injected	489	18.62	430	16.89	449	92.77	430	92.47

Overall campaign(s) merged results

```

---- CLASSIFICATION ----

```

	#	%	#	%	#	%	#	%	
Fault Annotations	2626		2546		484		465		
SAFE	S	1666	63.44	1666	65.44	8	1.65	8	1.72
DANGEROUS DETECTED	DD	362	13.79	303	11.90	322	66.53	303	65.16
DANGEROUS UNDETECTED	DU	123	4.68	123	4.83	123	25.41	123	26.45
Not Classified		475	18.09	454	17.83	31	6.40	31	6.67
UNOBSERVED DETECTED	UD	0	0.00	0	0.00	0	0.00	0	0.00
UNOBSERVED UNDETECTED	UU	211	8.04	199	7.82	31	6.40	31	6.67
NOT SIMULATABLE	NS	0	0.00	0	0.00	0	0.00	0	0.00
INJECTION FAILED	IF	0	0.00	0	0.00	0	0.00	0	0.00
NOT PROCESSED	NP	101	3.85	92	3.61	0	0.00	0	0.00
Others		163	6.21	163	6.40	0	0.00	0	0.00

Sampled fault scope

Fault Disposition (user refinement)

```

---- REFINEMENT ----

```

	#	#
To S	8	8
From UU	4	4
From DU	3	3
From DD	1	1

Applicable client configuration

```

---- METRICS ----

```

	%	%
Fault Coverage	16.83	71.08
Test Coverage	74.64	72.36

```

---- PARAMETERS ----
Fault Coverage : 100 * (DD + D) / (DD + DU + S + D + U + P + U+U + U+D)
Test Coverage  : 100 * (DD + D) / (DD + DU + D + U + P)
Merge File    : default
Refinement    : /vols/vmanager t2b/ferlini/activities/2022/FCM tech_up_22.09/refine2.vRefine
    
```

Fault and Test Coverage results and formulas

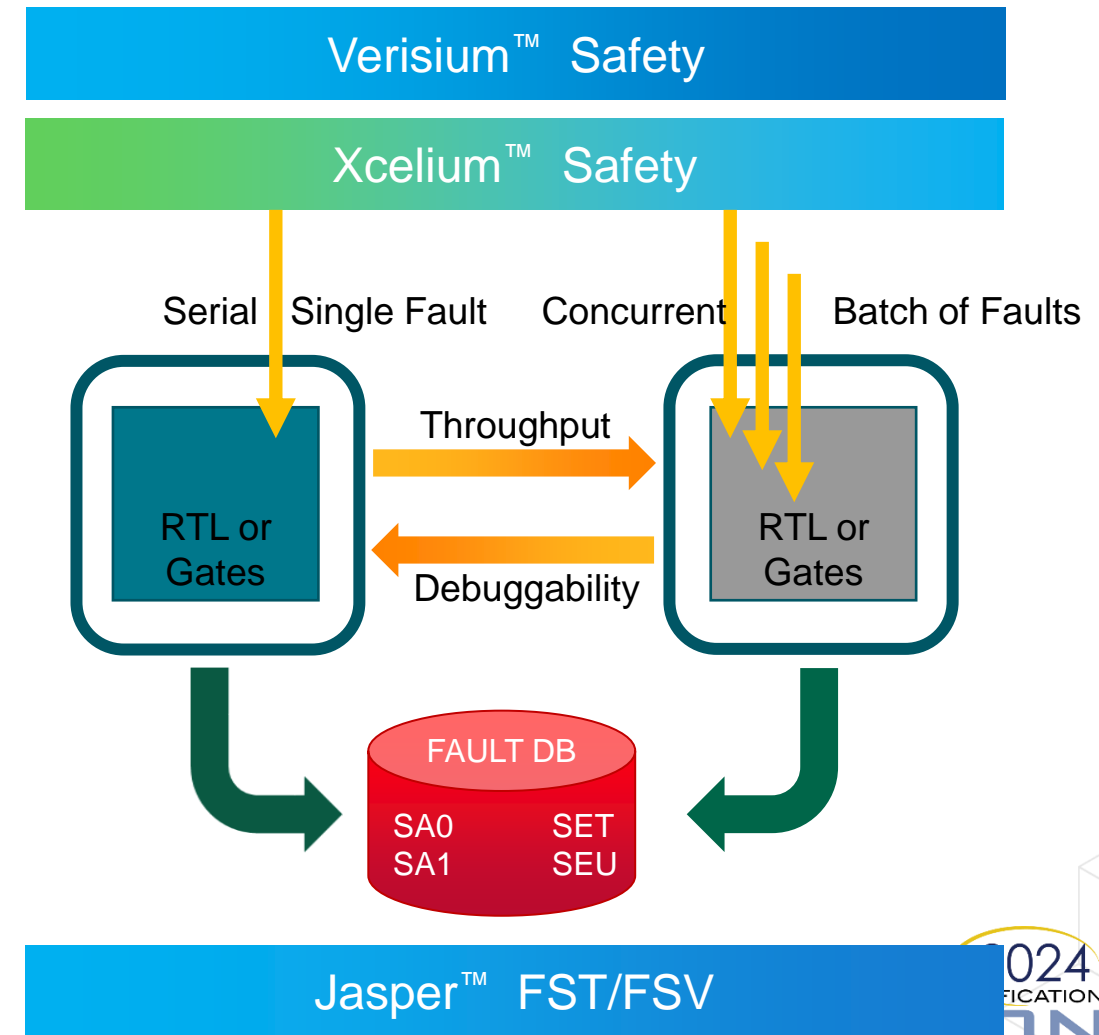


Fault Campaign Management – Safety Engines

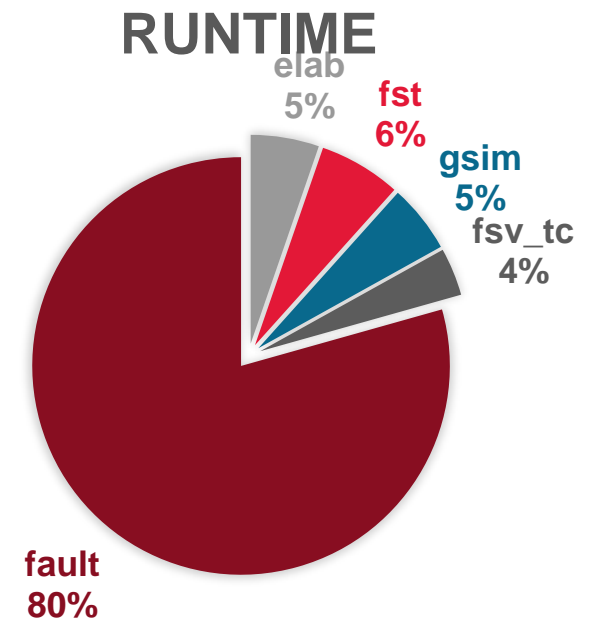
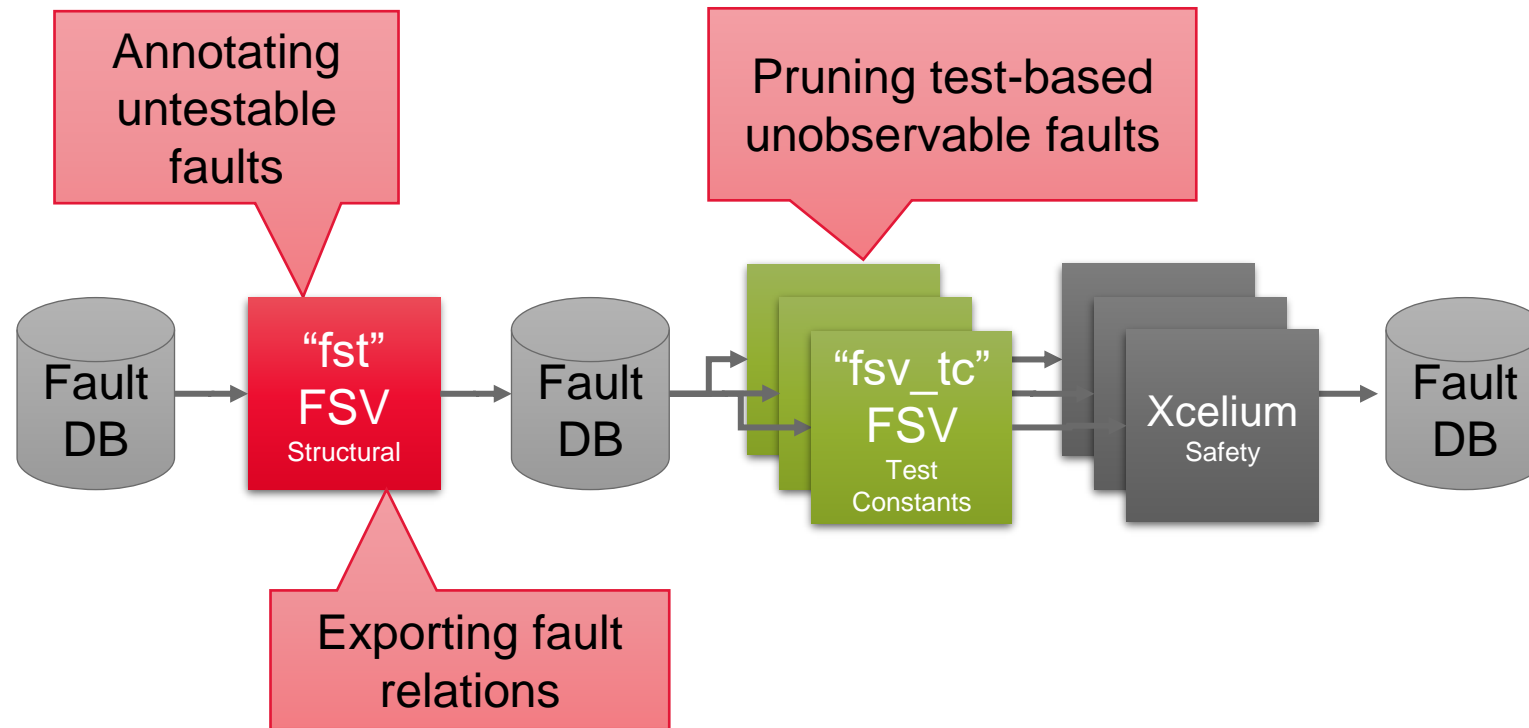


Xcelium Safety App

- The Xcelium Safety App provides native fault simulation by integrating Functional & Safety Engines
- Supports existing Xcelium Methodologies
 - Capture Replay, DSS (Dual Snapshot), Save Restore
- The Xcelium Safety App operates in 2 modes:
 - Serial mode: Flow setup and Debug
 - Concurrent mode:
 - Higher throughput
 - 5-100x faster than serial
 - Handles 2K to 20K faults in a single run (Single CPU Core)
- Supports Random Sampling as Sampling Percentage, Sampling Number
- Support Dual Strobe, Single Strobe Fault Classification
- Interoperable serial and concurrent fault simulation engines
- Both modes have identical flow and can easily switch back and forth
- The Xcelium Safety App simulates & annotates all faults in the fault DB
- Supports Fault Boundary to limit CoPF (Cone of Fault Propagation)



FCM – Optimizations from Jasper Safety (FSV)



A few minutes of optimization can save hours of simulation

- FSV exports fault relations → equivalent faults will be skipped
- FSV annotates untestable faults → Safe faults will be ignored
- FSV annotates faults as unobservable by test → Pruned faults will be dropped

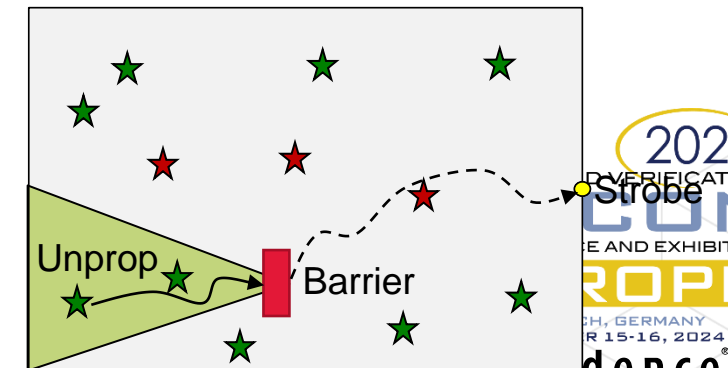
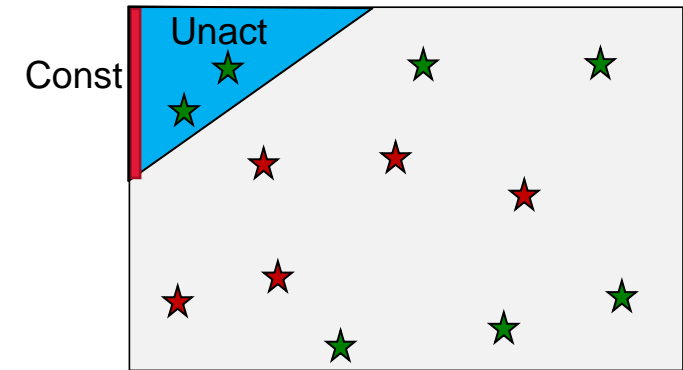
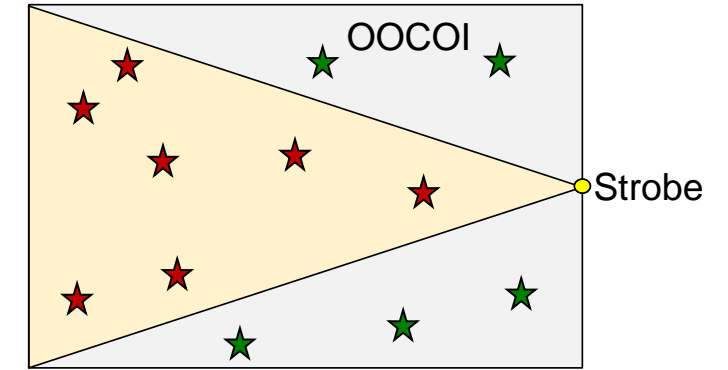


FSV Structural Analysis Check Types

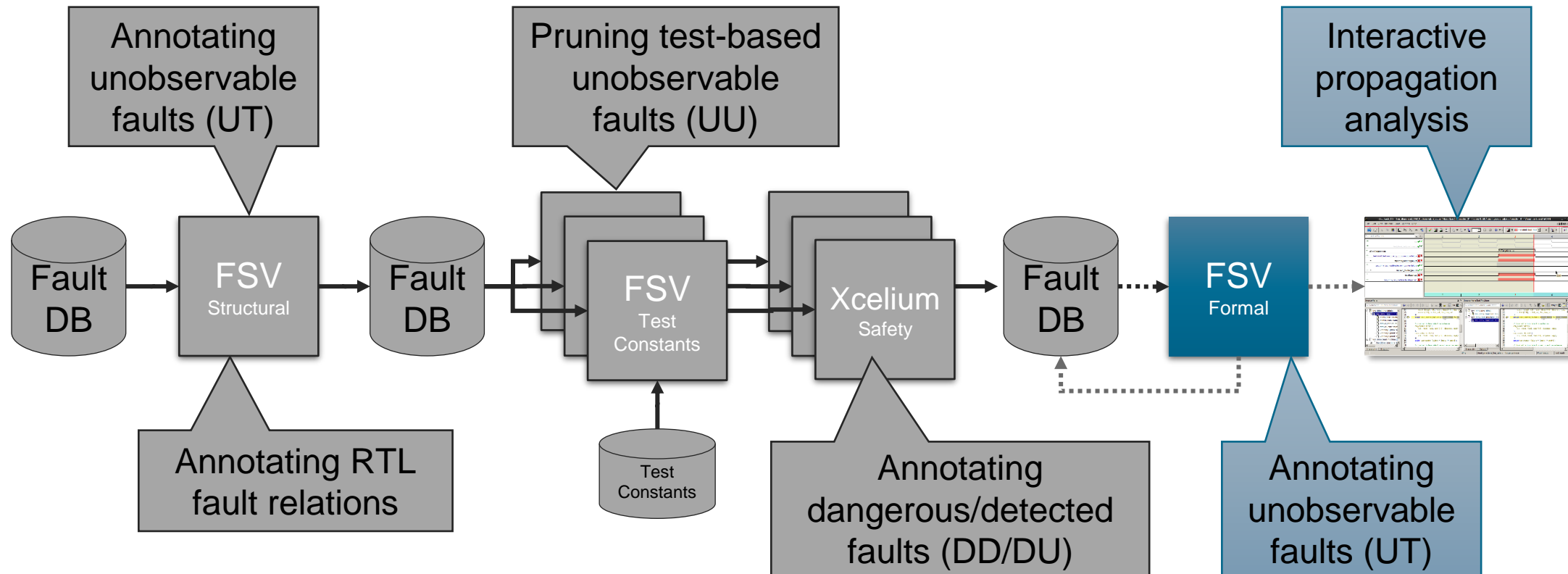


- Out-of-COI Analysis
 - A fault node outside the Cone-of-Influence (COI) has no physical connection to the functional strobe(s)
 - **Fault is Out-of-COI = Safe**
- Activatability Analysis
 - A SA0/1 fault injected on a node which is constant 0/1 cannot be activated
 - **Fault is Unactivatable = Safe**
- Propagatability Analysis
 - A fault that is activated and in COI, but cannot propagate to the functional strobe
 - **Fault is Unpropagatable = Safe**

★ Dangerous Fault
★ Safe Fault



FSV Integration with Xcelium Safety Simulator



- FSV Structural automatically annotates unobservable faults and RTL fault relations in database
- FSV TC prunes faults not exercisable by particular simulation test
- Xcelium Safety simulates and annotates all remaining faults in database
- FSV Formal annotates unobservable faults and provides interactive propagation analysis

FSV Formal – Debugging Visualize Waveforms

- Visualize for detection traces and unobservable analysis
 - Use Right-Mouse-Button Menu over an item in the Fault Table

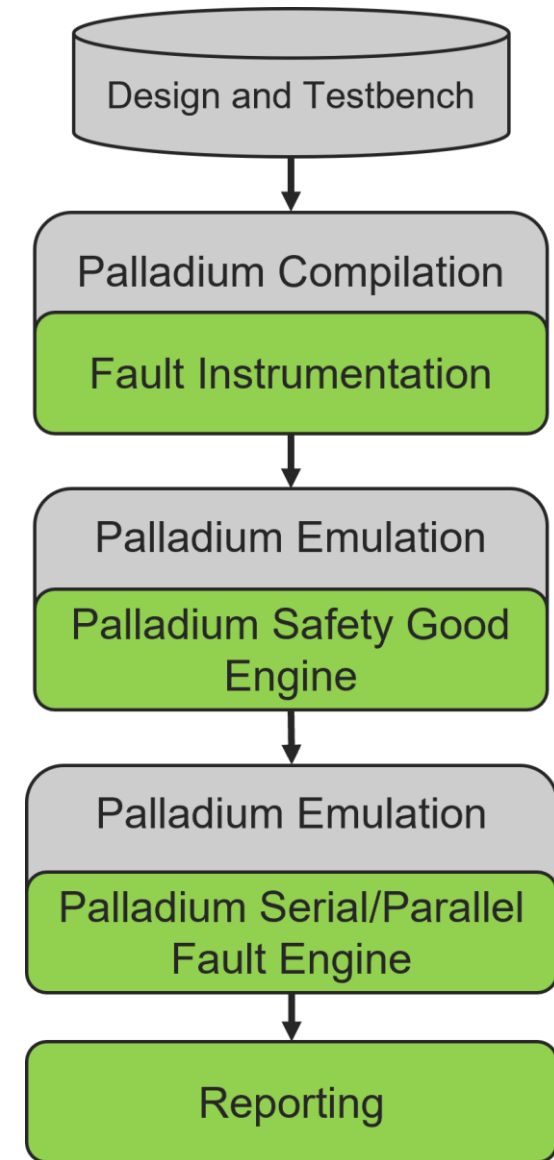
The screenshot shows the 'Fault Table' window with a table of fault entries. The row for 'top.nickel_out3' (ID 508) is selected. A right-click context menu is open over this row, with the 'Visualize' option highlighted. A red arrow points to the 'Visualize' option. The table has columns for ID, Node, Type, Injection, FCOI, CCOI, Constant, Activatability, FO Propagatab, and CO Detectabili. The status of the selected row is 'Activated' and 'Detected'.

ID	Node	Type	Injection	FCOI	CCOI	Constant	Activatability	FO Propagatab	CO Detectabili
504	top.nickel_out1	SA0	0	In	In	Unknown	Unprocessed	Unknown	Unprocessed
506	top.nickel_out2	SA0	0	In	In	Unknown	Unprocessed	Unknown	Unprocessed
508	top.nickel_out3	SA0	0	In	In	Unknown	Activated	Unpropagat...	Detected
510	top.nickels[0]	SA0	0	In	In	Unknown	Unproces	Unknown	Unprocessed
512	top.nickels[1]	SA0	0	In	In	Unknown	Unproces	Unknown	Unprocessed
514	top.nickels[2]	SA0	0	In	In	Unknown	Unproces	Unknown	Unprocessed
516	top.nickels[3]	SA0	0	In	In	Unknown	Unproces	Unknown	Unprocessed
518	top.nickels[4]	SA0	0	In	In	Unknown	Unproces	Unknown	Unprocessed
520	top.nickels[5]	SA0	0	In	In	Unknown	Unproces	Unknown	Unprocessed
522	top.nickels[6]	SA0	0	In	In	Unknown	Unproces	Unknown	Unprocessed
524	top.nickels[7]	SA0	0	In	In	Unknown	Unproces	Unknown	Unprocessed
526	top.quarter_in	SA0	0	In	In	Unknown	Unproces	Unknown	Unprocessed
530	top.two_dime_out	SA0	0	In	In	Unknown	Unproces	Unknown	Unprocessed
532	top.two_dime_out1	SA0	0	In	In	Unknown	Unproces	Unknown	Unprocessed
534	top.two_dime_out2	SA0	0	In	In	Unknown	Unproces	Unknown	Unprocessed
536	top.two_dime_out3	SA0	0	In	In	Unknown	Unproces	Unknown	Unprocessed
538	top.vendino1_clk	SA0	0	In	In	Unknown	Unproces	Unknown	Unprocessed

Palladium Safety

User Flow

- Easy to migrate from Functional verification flows to Fault Injection
 - Some files and option to be added to Palladium compilation
 - Faults are identified and instrumented during compilation
- Fault-free circuit emulation (Good Emulation)
 - Process strobe points and capture good waveform
- Fault Emulation Flows
 - Serial Fault Injection
 - Parallel Fault Injection
 - Interactive Fault Injection
- Fault Detection
 - Post-processing
 - Compares good and fault waveforms after each run
 - Inline
 - Detects the fault during the run using detection system
- Reporting
 - Standalone or using Xcelium utility (xfr)

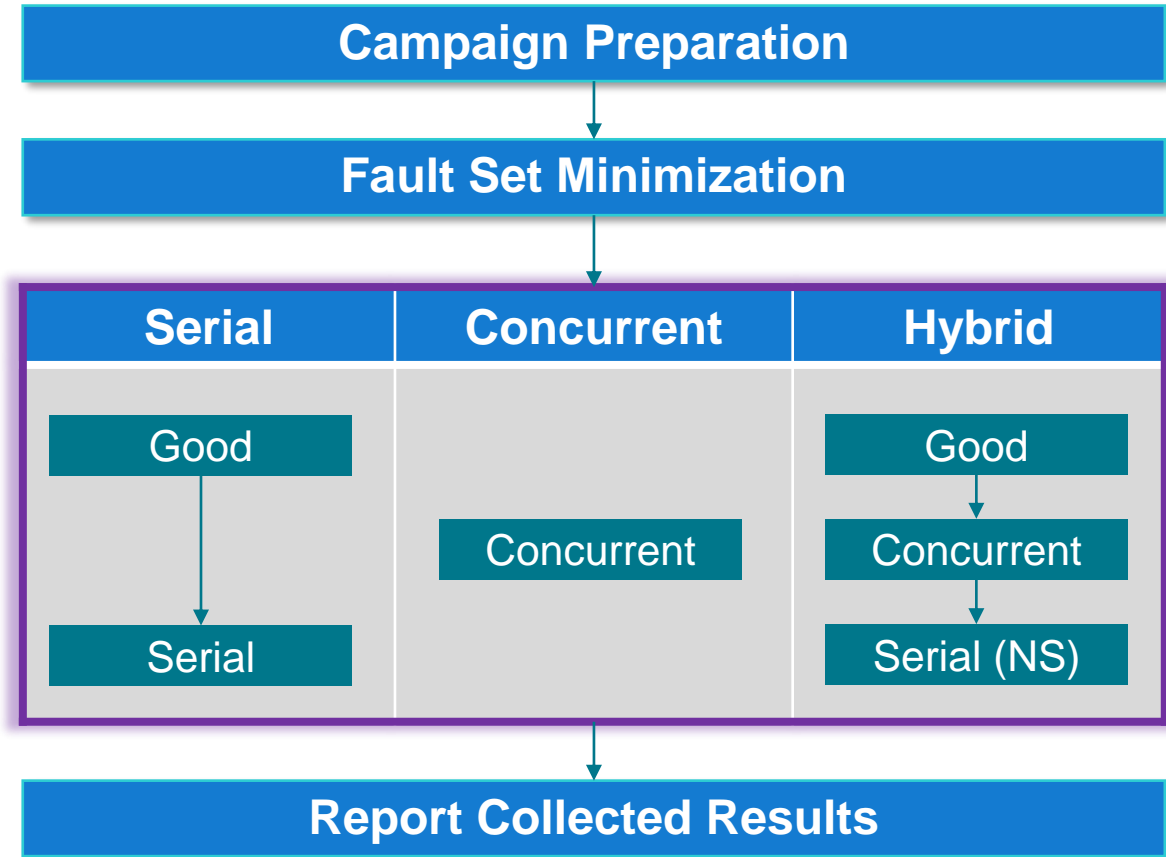
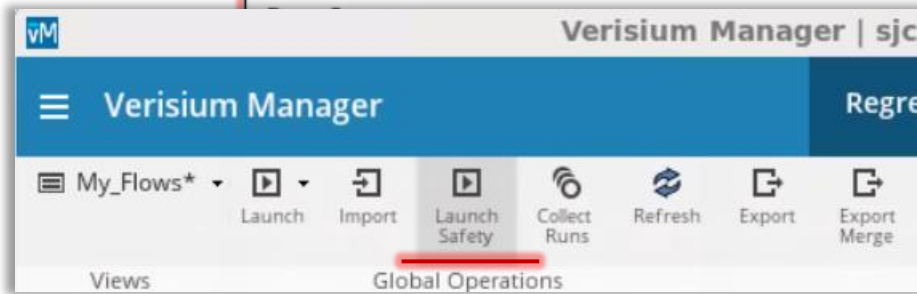
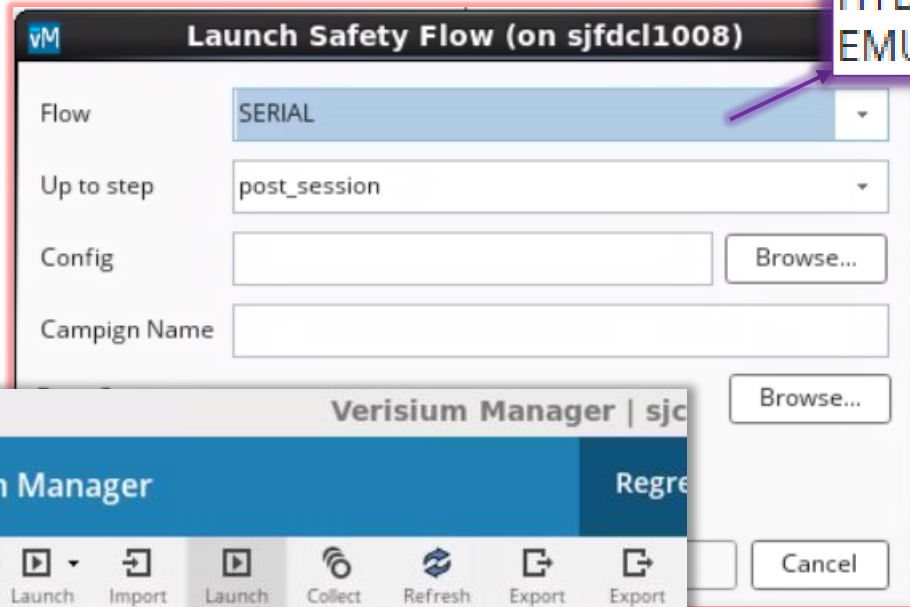




Fault Campaign Automation

Campaign Invocation

- GUI



- CLI

```
vmanager -safety \
  -execcmd "fi_campaign -launch <...> -flow_type <...> -cfg <...>"
```



Campaign Preparation

Organize Data

- Campaign directory

```
fs_exec_myc.HYBRID.usr...
├── flowData
├── input
├── report
├── sessions
│   ├── myc.HYBRID.usr...
│   ├── myc.test_select.usr...
│   ├── myc.elaboration.usr...
│   ├── myc.fst.usr...
│   ├── myc.good_simulation.usr...
│   ├── myc.fault_pruning.usr...
│   ├── myc.fault.usr...
│   └── myc.fault.usr...
```

Translate Inputs

- User-input (e.g., stobes)

```
strobe functional top.dut.o
strobe checker top.sm.alarm
```

- Xcelium syntax

```
fs_strobe -functional top.dut.o
fs_strobe -checker top.sm.alarm
```

- Jasper syntax

```
strobe functional dut.out
strobe checker sm.alarm
```

Prune Tests (optional)

- Remove redundant tests
 - 0% additional coverage
- Order per cov/time
- Customizable heuristic
 - Coverage type and contribution threshold
- Permanent campaigns
 - Select functional tests



Prepare

Minimize

Execute

Report

Analyze

Campaign Parameters

```
fault_target top... -type sa0+sa1
```

Fault spec.

```
strobe functional top.dut.o
strobe checker top.sm.alarm
```

Strobe list

```
session dv {};
Group tests {
  test t1 : {};
  test t2 : {};
  ...
}
```

Test List

```
xrun -64bit \
  $FS_SIM_PARAM \
  ...
```

Sim script



Customizations

Configuration file

```

FS_EXEC_FAULT_TYPE           : permanent
FS_FAULT_LIST_FILE_NAME     : ../faults.list
FS_STROBE_LIST_FILE_NAME    : ../strokes.list
FS_SAMPLING_PERCENT         :
FS_SAMPLING_...             :
FS_REGR_TESTS_VSIF         : ../tests.vusif
FS_TOP_DIR                  : ../sessions
FS_FAULT_TOP                : tb.top
FS_REGR_TESTS_REFINE       :
FS_FSIM_SCRIPT              : ../fsim.csv
FS_STROBE_DEFAULT_EVENT    :
FS_FAULT_INJECT_CONDITION  :
FS_ENABLE_TEST_SELECTION   : FALSE
FS_FAULT_STOP_SEVERITY     : 3
FS_FAULT_REDUCTION_LEVEL   : FSV_FST_ONLY
FS_FAULT_RELATION_LEVEL    : FSV_FST_ONLY
FS_FAULT_PRUNING_LEVEL     : FSV_TC_ONLY
FS_FAULT_USE_TEST_CONST    : unobservable
FS_FST_SCRIPT               :
...

```

Domain driven configuration



Midas

FMEDA Analysis



Verification Environment



Fault Set Minimization

Verisium Manager Safety



Legend:
 - Mandatory parameters
 - Midas overridden

2024

Fault Set Minimization

Design Structure

Testability Analysis

Identify faults:
- Uncontrollable
- Unobservable

Fault Collapsing

Group equivalent faults and consider only their prime representative

Statistics

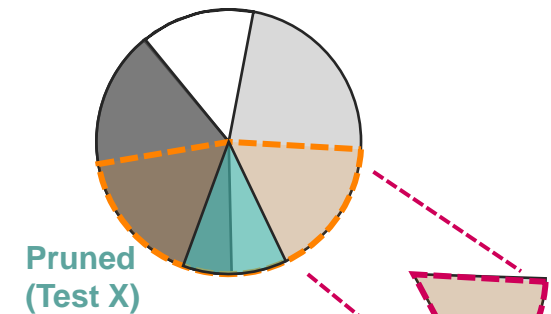
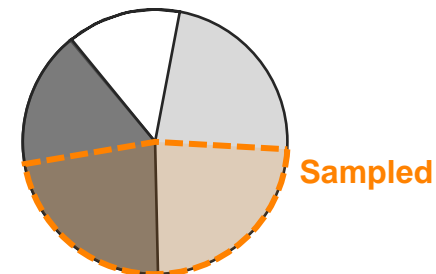
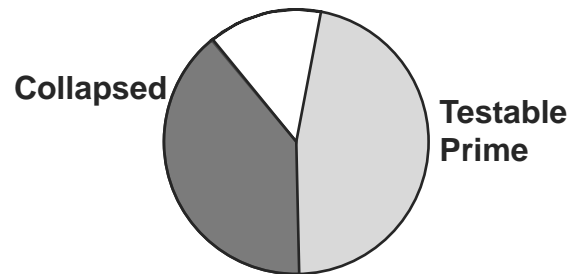
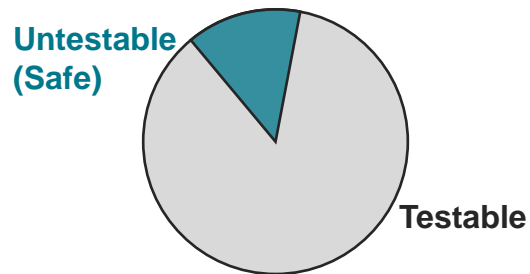
Random Sampling

Estimate the overall results based on a representative sample

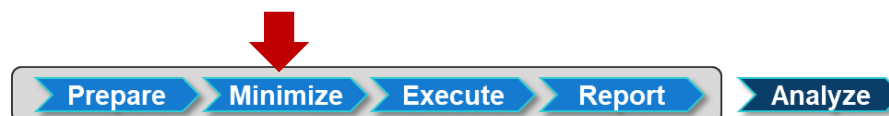
Test Stimulus

Fault Pruning

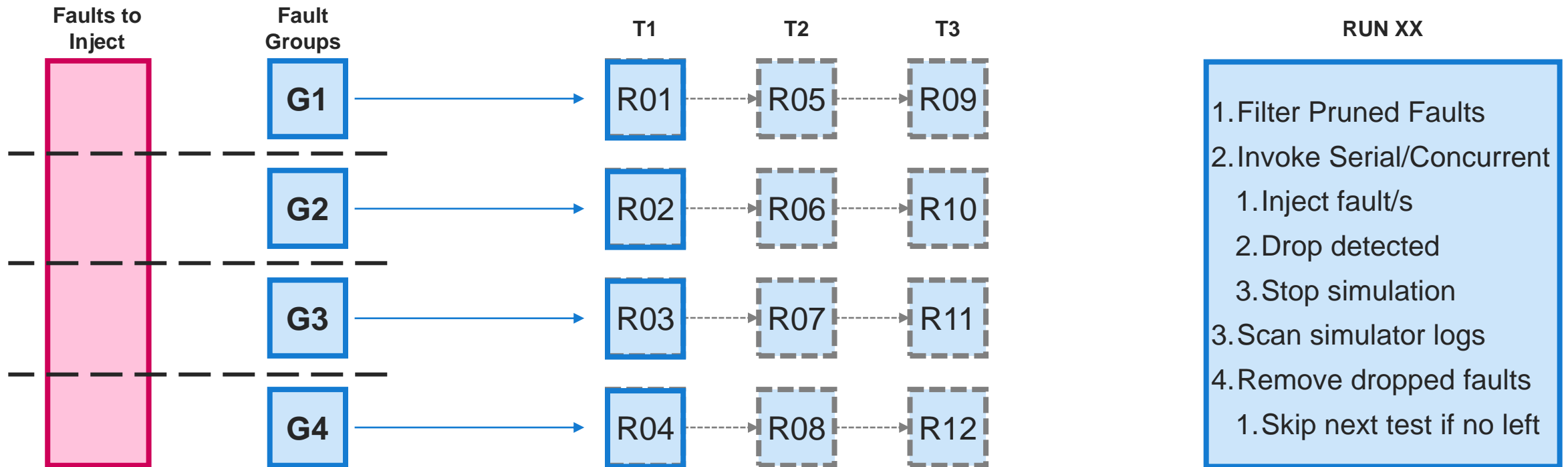
Find extra untestable faults by constraining testability based on stimulus patterns



Sampled Testable Prime Faults to Inject



Fault Injection Execution



Fault Grouping

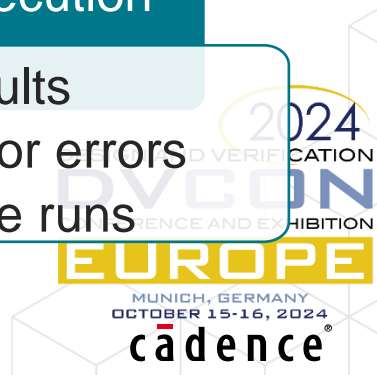
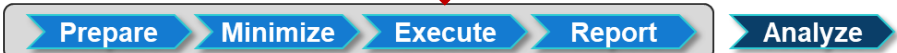
- Configurable
- Max F per G (Serial = 1)

Fault Session Build

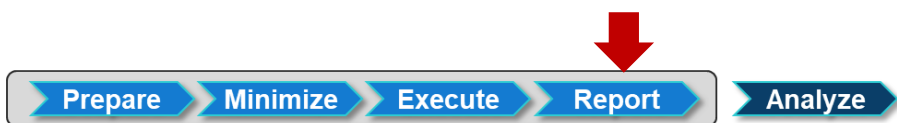
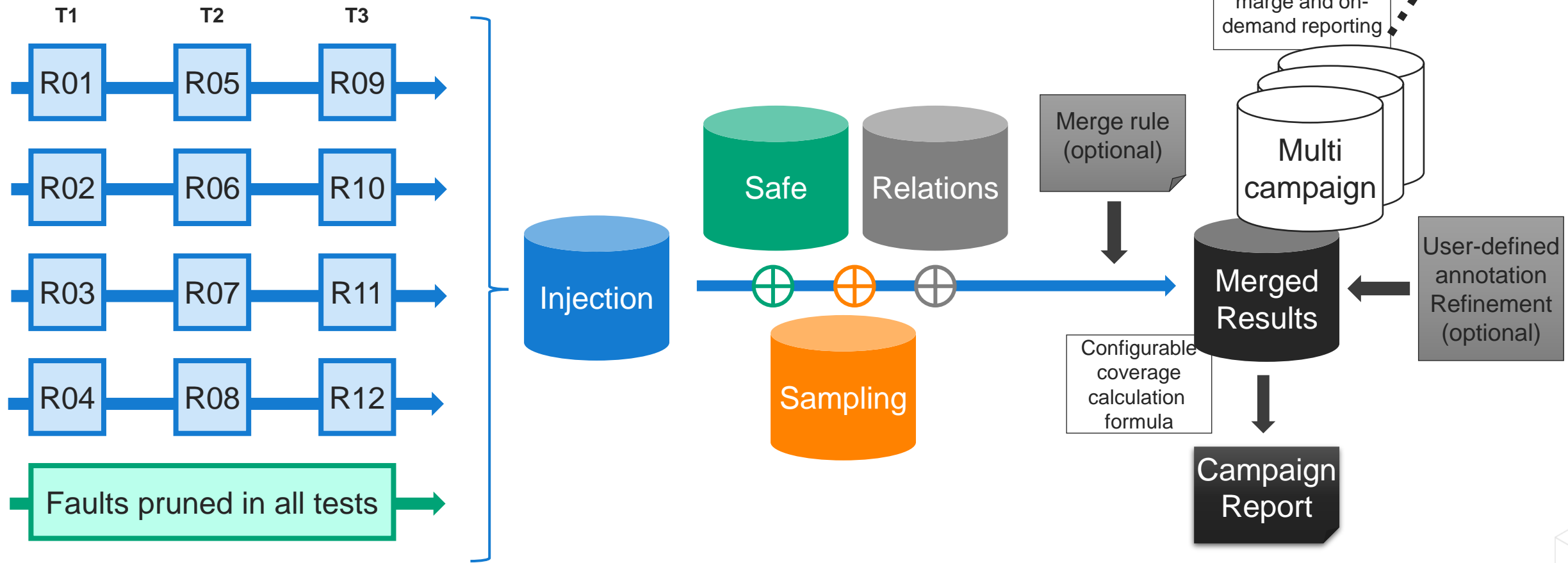
- Test dependency
- Submit runs to the computer farm

Fault Run Execution

- Filter faults
- Check for errors
- Optimize runs



Reporting Campaign Results





Fault Campaign Analysis

Fault Simulation Results

Run generated fault annotation

Verisium Manager | sjcvl-safety:44001 | 64b | ferlini [Analysis Center] (on sjfdcl1008)

Verisium Manager | Regression | Analysis | Planning | Composer (Beta) | Tracking

Default* | Runs Query | Metrics | Tests | vPlan | Reload vPlan | Reload Coverage | Refresh Runs | Scripts Manager | MyAction | New vPlan | Edit vPlan | Failures | Runs | Formal Prop. | Correlate Runs | Rank Runs | Edit all at once | Edit each | Edit all Runs | History | Report | Help

Views | Context Operations | Scripts | Planning | Analyze | Runs | Report | Help

Runs | /Risc_core_tests/short_test_Faults_Group_0

Index	Name	Status	Duration (sec.)	Fault Node	Fault Type	Fault Annotation	Fault Inject Time
1	/Risc_core_tests/short_test_Faults_Group_0	passed	71	or1200_cpu.or1200_if.g...	sa0	DD	10us
2	/Risc_core_tests/short_test_Faults_Group_1	passed	71	or1200_cpu.or1200_if.g...	sa0	DD	10us
3	/Risc_core_tests/fpu_test_Faults_Group_0	passed	126	or1200_cpu.or1200_if.g...	sa0	DD	10us
4	/Risc_core_tests/fpu_test_Faults_Group_1	passed	126	or1200_cpu.or1200_if.g...	sa1	DD	10us

Showing 4 items

Fault Node	Fault Type	Fault Annotation	Fault Inject Time
or1200_cpu.or1200_if.g...	sa0	DD	10us
or1200_cpu.or1200_if.g...	sa0	DD	10us
or1200_cpu.or1200_if.g...	sa0	DD	10us
or1200_cpu.or1200_if.g...	sa1	DD	10us
or1200_cpu.or1200_if.g...	sa0	DD	10us
or1200_cpu.or1200_if.g...	sa1	DD	10us
or1200_cpu.or1200_if.g...	sa0	DD	10us
or1200_cpu.or1200_if.g...	sa0	DD	10us
or1200_cpu.or1200_if.g...	sa1	DD	10us
or1200_cpu.or1200_if.g...	sa0	DD	10us
or1200_cpu.or1200_if.g...	sa1	DD	10us
or1200_cpu.or1200_if.g...	sa0	DD	10us
or1200_cpu.or1200_if.g...	sa1	DD	10us
or1200_cpu.or1200_if.g...	sa1	DD	10us
or1200_cpu.or1200_if.g...	sa1	DD	10us

Showing 469 items

cadence Analysis Center | Messages



Fault Campaign Results – Hierarchical View

Merged annotation

Verisium Manager | sjcvi-safety:44001 | 64b | ferlini [Analysis Center] (on sjfdcl1008)

Verisium Manager | Regression | Analysis | Planning | Composer (Beta) | Tracking

Block Expression Toggle Statement FSM Cover Group Assertion Fault Advanced Fault Correlate Runs Rank Runs Show Contributing Runs Read Save Reload Unload Calculate DC and S Set merge Rule Safety Report Reports Help

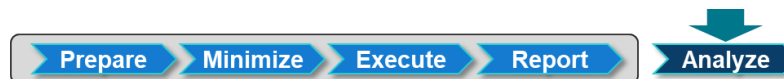
Views Analyze Refinement Files Safety Report Help

Verification Hierarchy

default

Name	Fault Node Detected Grade	Fault Node Total	Fault Sample Set Total	Fault NP	Fault S	Fault DD	Fault DU	Fault UD	Fault UU
(no filter)	!=-1.0	(no filter)	(no filter)	(no filter)	(no filter)	(no filter)	(no filter)	(no filter)	(no filter)
Verification Metrics	19.18%	3508	2184 / 3508 (62.26%)	0 / 2184 (0%)	0 / 2184 (0%)	25 / 2184 (1.14%)	13 / 2184 (0.6%)	648 / 2184 (29.67%)	1006 / 2184 (46.06%)
Types	n/a	0	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)
Instances	19.18%	3508	2184 / 3508 (62.26%)	0 / 2184 (0%)	0 / 2184 (0%)	25 / 2184 (1.14%)	13 / 2184 (0.6%)	648 / 2184 (29.67%)	1006 / 2184 (46.06%)
test_drink	19.18%	3508	2184 / 3508 (62.26%)	0 / 2184 (0%)	0 / 2184 (0%)	25 / 2184 (1.14%)	13 / 2184 (0.6%)	648 / 2184 (29.67%)	1006 / 2184 (46.06%)
top	19.18%	3508	2184 / 3508 (62.26%)	0 / 2184 (0%)	0 / 2184 (0%)	25 / 2184 (1.14%)	13 / 2184 (0.6%)	648 / 2184 (29.67%)	1006 / 2184 (46.06%)
coins	2.08%	866	866 / 866 (100%)	0 / 866 (0%)	0 / 866 (0%)	18 / 866 (2.08%)	0 / 866 (0%)	0 / 866 (0%)	533 / 866 (61.55%)
coins1	0%	866	0 / 866 (0%)	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)
diag	0%	160	84 / 160 (52.5%)	0 / 84 (0%)	0 / 84 (0%)	0 / 84 (0%)	13 / 84 (15.48%)	0 / 84 (0%)	50 / 84 (59.52%)
drinks	1.83%	382	382 / 382 (100%)	0 / 382 (0%)	0 / 382 (0%)	7 / 382 (1.83%)	0 / 382 (0%)	0 / 382 (0%)	231 / 382 (60.47%)
drinks1	0%	382	0 / 382 (0%)	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)
vending1	76.76%	284	284 / 284 (100%)	0 / 284 (0%)	0 / 284 (0%)	0 / 284 (0%)	0 / 284 (0%)	218 / 284 (76.76%)	63 / 284 (22.18%)
vending2	76.41%	284	284 / 284 (100%)	0 / 284 (0%)	0 / 284 (0%)	0 / 284 (0%)	0 / 284 (0%)	217 / 284 (76.41%)	63 / 284 (22.18%)
vending3	75%	284	284 / 284 (100%)	0 / 284 (0%)	0 / 284 (0%)	0 / 284 (0%)	0 / 284 (0%)	213 / 284 (75%)	66 / 284 (23.24%)

cadence Analysis Center | Messages



Fault Campaign Results – Annotated Fault List

Merged annotation

The screenshot displays the Verisium Manager interface. The top navigation bar includes 'Regression', 'Analysis', 'Planning', 'Composer (Beta)', and 'Tracking'. The 'Analysis' tab is active, showing a toolbar with various tools like 'Scripts Manager', 'MyAction', 'New vPlan', 'Edit vPlan', 'Correlate Runs', 'Rank Runs', 'Fault Advanced', 'Show Local', 'Refine Annotation', 'Un Refine', 'Read', 'Save', 'Reload', 'Unload', 'Report', and 'Help'.

The main area shows 'Instance (default):' with a red box highlighting 'Current instance accumulated metrics'. Below this, a summary bar displays: 'Fault Node Total: 3508.0 | Fault Sample Set Total: 2184.0 | Fault NP: 0 / 2184 (0%) | Fault S: 0 / 2184 (0%) | Fault DD: 25 / 2184 (1.14%) | Fault DU: 13 / 2184 (0.6%) | Fault UD: 648 / 2184 (29.67%)'. A red box highlights '60+ fault attributes' in the top right of the table area.

The main table lists faults with columns: Fault Node, Fault Type, Fault Annotation, Fault Inject Time, Name, Col #, Name, and Value. A red box highlights the 'Refine Annotation' menu in the bottom left corner, which includes options like 'Refine Annotation', 'Un Refine', 'Refine Tag', 'Remove Refine Tag', 'Exclude Type', 'Un-Exclude Type', 'Exclude Resilience', 'Review', 'Un Review', and 'Clear exclusion mark'.

Ex:	Fault Node	Fault Type	Fault Annotation	Fault Inject Time	Name	Col #	Name	Value
	(no filter)	(no filter)	(no filter)	(no filter)	(no filter)		(no filter)	(no filter)
	test_drink.top.coins.g92...	sa1	UU		test_drink.top.coins.g922.A.sa1	3	Fault Annotation	DD
	test_drink.top.coins.g92...	sa0	DD	500ns	test_drink.top.coins.g922.A.sa0		Fault Approach	
	test_drink.top.coins.g92...	sa1	UU		test_drink.top.coins.g922.B.sa1		Fault Classification	DD
	test_drink.top.coins.g92...	sa0	DD	500ns	test_drink.top.coins.g922.B.sa0		Fault Detect Time	500ns
	s.g92...	sa0	UU		test_drink.top.coins.g922.Y.sa0		Fault Engine Type	XFS Concurrent
	s.g92...	sa1	DD	500ns	test_drink.top.coins.g922.Y.sa1		Fault Functional Detect Time	500ns
	s.g92...	sa1	UU		test_drink.top.coins.g923.A.sa1		Fault Hold Time	
	s.g92...	sa0	UU		test_drink.top.coins.g923.A.sa0	4	Fault Inject Time	500ns
	s.g92...	sa0	UU		test_drink.top.coins.g923.B.sa0	1	Fault Node	test_drink.top.c...
	s.g92...	sa1	DD	500ns	test_drink.top.coins.g923.B.sa1		Fault Node Average Grade	100%
	s.g92...	sa0	DD	500ns	test_drink.top.coins.g923.Y.sa0		Fault Node Detected	1.0
	s.g92...	sa1	UU		test_drink.top.coins.g923.Y.sa1		Fault Node Detected Grade	100%
	ems						Fault Node Excluded	0.0

Fault Annotation Traceability

Result per each test

The screenshot displays the Verisium Manager Analysis Center interface. The top navigation bar includes 'Regression', 'Analysis', 'Planning', 'Composer (Beta)', and 'Tracking'. The 'Analysis' tab is active, showing a toolbar with various actions like 'Edit all at once', 'Edit each', 'History', 'Rerun', 'Create Context', 'Open dir', 'Exclude', 'Un-Exclude', 'Read', 'Save', 'Edit Comment', 'Report', and 'Help'.

The main workspace is divided into several panels:

- Groups of Faults of:** A summary table showing test results.
- Runs:** A table showing the execution of a specific run.
- Details:** A table showing the attributes of the selected run.
- Context Menu:** A menu is open over the 'Runs' table, showing options like 'Exclude', 'Edit all selected at once', 'Edit each', 'Rerun', 'Create Context', 'Open run directory', 'Clear Filters', and 'Undo Sort'.

Groups of Faults of:

Test Name	Fault Annotation	Number Of Entities	Fault Type
(no filter)	(no filter)	(no filter)	(no filter)
nickel_random	UD	1	sa0
quarter_random	UD	1	sa0
dime_random	UU	1	sa0

Runs:

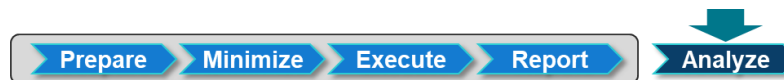
Name	Index	Status	Duration (sec.)
(no filter)	(no filter)	(no filter)	(no filter)
/tests/nickel_Faults_Group_0	1	passed	

Details:

Col #	Name	Value
(no filter)	(no filter)	(no filter)
	AbstractCO Annotation	CD
	AbstractFO Annotation	FU
	Enclosing Entity	test_drink.top.vend
	Exclusion Rule Type	None
	Fault Annotation	UD

Context Menu Options:

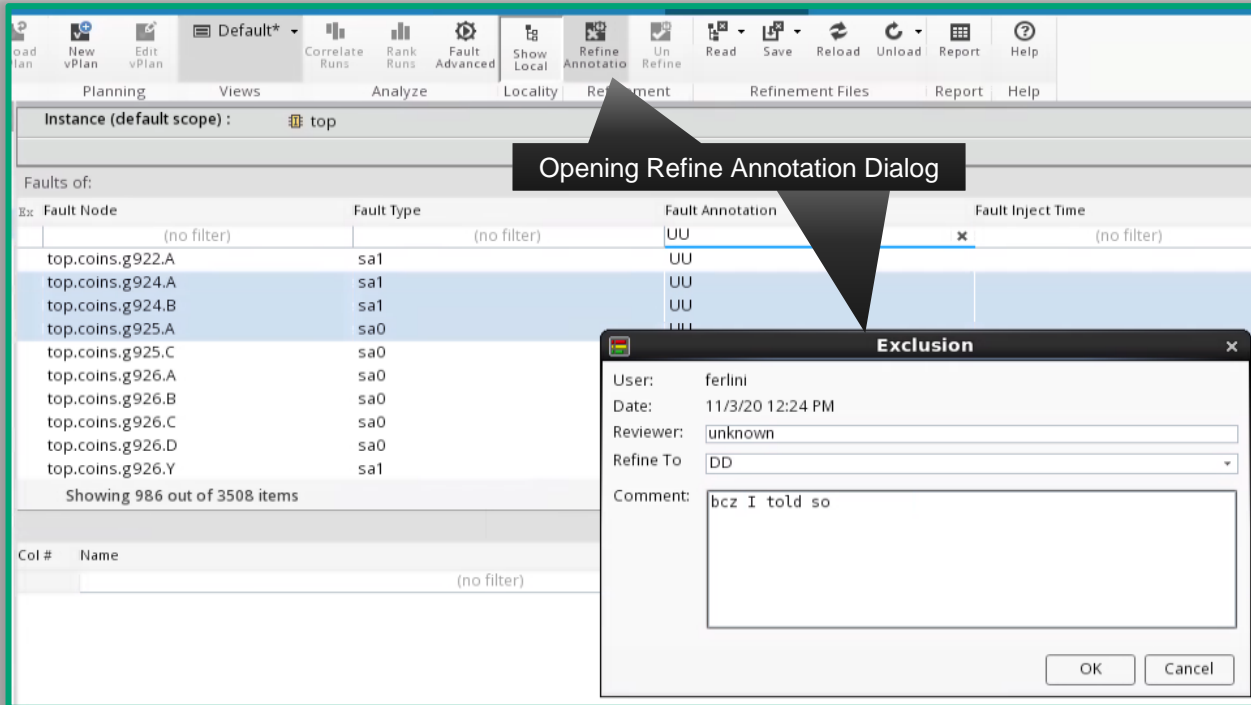
- Filter...
- Exclude
- Edit all selected at once
- Edit each
- Rerun
- Create Context
- Open run directory
- Clear Filters
- Undo Sort
- Copy Cell
- Copy Row



Fault Annotation Refinement

Dispositioning unclassified faults

- GUI and CLI →



```

vmanager> refine_annotation -faults {top.vending1.\\current_state_reg\[3\] } \
    -fault_type SEU -refineTo S -comment {bcz I want}
...
A total of 8 faults were refined to S
vmanager> save -refinement fcm_refinement.vRefine
vmanager> fi_campaign -report -overwrite -output fcm_refined_report
Writing report to: fcm_refined_report/faultsim_stat_summary.report
    
```



	Total Faults	Total Prime	Sample Faults	Sample Prime
	#	%	#	%
INSTRUMENTATION				
Faults	2626		2546	465
Safe	1658 63.14	1658 65.12	0 0.00	0 0.00
Not Injected	479 18.24	458 17.99	35 7.23	35 7.53
Injected	489 18.62	430 16.89	449 92.77	430 92.47
CLASSIFICATION				
Fault Annotations	2626	2546	484	465
SAFE	S 1666 63.44	1666 65.44	8 1.65	8 1.72
DANGEROUS DETECTED	DD 362 13.79	303 11.90	322 66.53	303 65.16
DANGEROUS UNDETECTED	DU 123 4.68	123 4.83	123 25.41	123 26.45
Not Classified	475 18.09	454 17.83	31 6.40	31 6.67
UNOBSERVED DETECTED	UD 0 0.00	0 0.00	0 0.00	0 0.00
UNOBSERVED UNDETECTED	UU 211 8.04	199 7.82	31 6.40	31 6.67
NOT SIMULATABLE	NS 0 0.00	0 0.00	0 0.00	0 0.00
INJECTION FAILED	IF 0 0.00	0 0.00	0 0.00	0 0.00
NOT PROCESSED	NP 101 3.85	92 3.61	0 0.00	0 0.00
Others	163 6.21	163 6.40	0 0.00	0 0.00
REFINEMENT				
To S	8	8		
From UU	4	4		
From DU	3	3		
From DD	1	1		
METRICS				
Fault Coverage	16.83	71.08		
Test Coverage	74.64	72.36		
PARAMETERS				
Fault Coverage	: 100 * (DD + D) / (DD + DU + S + D + U + P + U+U + U+D)			
Test Coverage	: 100 * (DD + D) / (DD + DU + D + U + P)			
Merge File	: default			
Refinement	: /vols/vmanager_t2b/ferlini/activities/2022/FCM_tech_up_22.09/refine2.vRefine			

Fault Tagging

- What?
 - User-editable (string) attribute per fault metric element
- Why?
 - Support post-campaign analysis (debug, refinement, etc.) by tagging relevant faults
 - Logically gather faults even if they do not share a common attribute value (e.g., hierarchy, annotation)
- How?

Fault Tag (on sjfhw636)

Tag text: TAG1

OK Cancel

Refinement

- Read Refinement
- Read Mapping Refinement With Condition
- Read Tag Refinement file
- Save All Refinements
- Save Refinement file...
- Save updates in loaded refinement files
- Save Tag Refinement file

Reuse stored "Fault Tag" like with "Refinement"

No impact on annotation or campaign results

"Fault Tag" is kept consistent across all equivalent faults automatically

Fault Node	Fault Type	Fault Inject Time	Fault Annotation	Is Prime	Prime Node	Prime Type	Prime Inject Time
(no filter)	(no filter)	(no filter)	(no filter)		(no filter)	(no filter)	(no filter)
dut_inst.mem2_i.mem_with_crc_i.g39.S0	sa1	45ns	DD	true	dut_inst.mem2_i.mem_with_crc_i.g39.S0	sa1	45ns
dut_inst.mem2_i.mem_with_crc_i.g39.S0	sa0	45ns	DD	true	dut_inst.mem2_i.mem_with_crc_i.g39.S0	sa0	45ns
dut_inst.mem1_i.mem_with_crc_i.mem_crc_reg[7].D	sa1		S	true	dut_inst.mem1_i.mem_with_crc_i.mem_crc_reg[7].D	sa1	
dut_inst.mem1_i.mem_with_crc_i.g118.Y	sa1		S	false	dut_inst.mem1_i.mem_with_crc_i.mem_crc_reg[7].D	sa1	
dut_inst.mem1_i.mem_data_ff_tmp_reg[17].RN	sa0	45ns	DU	true	dut_inst.mem1_i.mem_data_ff_tmp_reg[17].RN	sa0	45ns
dut_inst.mem1_i.mem_data_ff_tmp_reg[17].RN	sa1	45ns	UU	true	dut_inst.mem1_i.mem_data_ff_tmp_reg[17].RN	sa1	45ns
dut_inst.mem1_i.mem_with_crc_i.g118.S0	sa0		S	true	dut_inst.mem1_i.mem_with_crc_i.g118.S0	sa0	
dut_inst.mem1_i.mem_with_crc_i.g117.S0	sa0	45ns	DD	true	dut_inst.mem1_i.mem_with_crc_i.g117.S0	sa0	45ns
dut_inst.mem1_i.mem_with_crc_i.g117.S0	sa1	45ns	DD	true	dut_inst.mem1_i.mem_with_crc_i.g117.S0	sa1	45ns
dut_inst.mem1_i.mem_with_crc_i.g117.Y	sa0	45ns	DD	false	dut_inst.mem1_i.mem_with_crc_i.mem_reg[6].D	sa0	45ns

Showing 2626 items

Prepare Minimize Execute Report Analyze

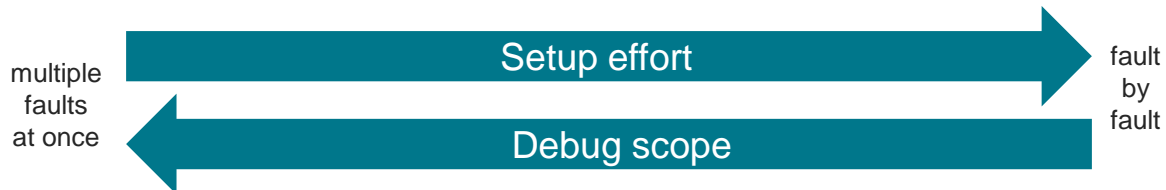
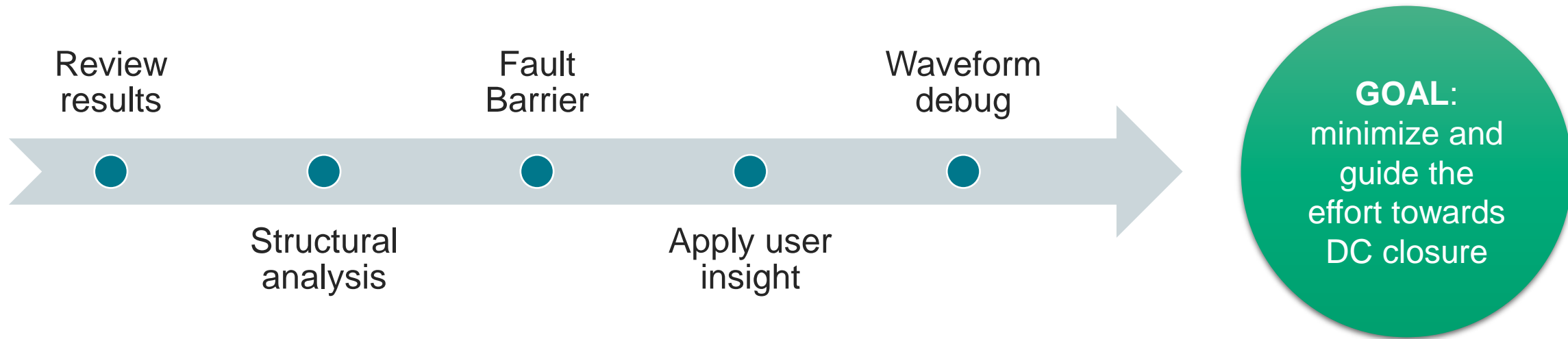
© 2024 Cadence Design Systems, Inc. All rights reserved.

cadence



Fault Campaign Debug

Fault Campaign Closure



FSV Formal – Visualize Fault Detection Traces

The screenshot displays the Cadence JasperGold interface for formal verification. The main window shows a trace visualization with the following components:

- Reset Signals:** Includes `reset` and `test_drink_bad_machine.reset`, both marked with green checkmarks.
- Fault Signal:** Includes `top.nickel_in` and `test_drink_bad_machine.top.nickel_in`, both marked with red X's.
- FD Strokes:** Includes `top.dispense` and `test_drink_bad_machine.top.dispense` (marked with red X's), and `top.dime_out` and `test_drink_bad_machine.top.dime_out` (marked with green checkmarks).
- Other Signals:** `top.exact_change` and `test_drink_bad_machine.top.exact_change` are marked with green checkmarks.

Key annotations in the trace include:

- Twin Signals from good and bad machine:** Points to the `reset` and `exact_change` signals.
- SA0 Fault injected:** Points to a red box highlighting the `top.nickel_in` signal transition.
- SA0 Fault activated:** Points to a red box highlighting the `test_drink_bad_machine.top.nickel_in` signal transition.
- Fault observed at strobe:** Points to a red box highlighting the `top.dispense` signal transition.
- Twin source code with value annotation from good and bad machine:** Points to the state machine logic in the source panes.

The source panes show the state machine logic for `test_drink` and `test_drink_bad_machine`. The `test_drink` pane shows the state machine logic, and the `test_drink_bad_machine` pane shows the state machine logic with a value annotation for `current_state`.

Fault Merged Annotation Per Each Test

Add / Remove tests

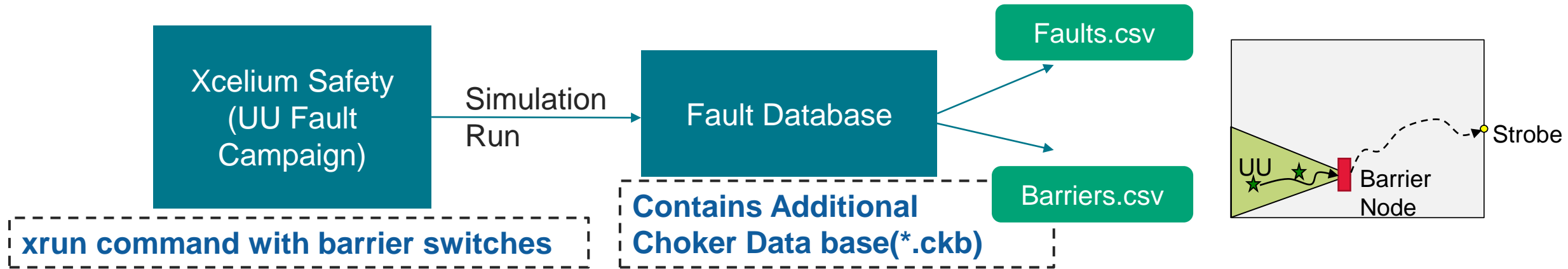
- Test 1 – Nickel

Name	Fault Node Detected Grade	Fault Node Total	Fault Sample Set Total	Fault NP	Fault S	Fault DD
(no filter)	-1.0	(no filter)	(no filter)	(no filter)	(no filter)	(no filter)
Verification Metrics	16.9%	3508	2184 / 3508 (62.3%)	0 / 3508 (0%)	1324 / 3508 (3...)	25 / 3508 (0.7...)
Types	n/a	0	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)
Instances	16.9%	3508	2184 / 3508 (62.3%)	0 / 3508 (0%)	1324 / 3508 (3...)	25 / 3508 (0.7...)
top	16.9%	3508	2184 / 3508 (62.3%)	0 / 3508 (0%)	1324 / 3508 (3...)	25 / 3508 (0.7...)
coins	2.1%	866	866 / 866 (100%)	0 / 866 (0%)	0 / 866 (0%)	18 / 866 (2.1%)
coins1	0%	866	0 / 866 (0%)	0 / 866 (0%)	866 / 866 (10...)	0 / 866 (0%)
diag	3.8%	160	84 / 160 (52.5%)	0 / 160 (0%)	76 / 160 (47.5...)	0 / 160 (0%)
drinks	1.8%	382	382 / 382 (100%)	0 / 382 (0%)	0 / 382 (0%)	7 / 382 (1.8%)

- Test 2 - Quarter

Name	Fault Node Detected Grade	Fault Node Total	Fault Sample Set Total	Fault NP	Fault S	Fault DD
(no filter)	-1.0	(no filter)	(no filter)	(no filter)	(no filter)	(no filter)
Verification Metrics	11.3%	3508	2184 / 3508 (62.3%)	25 / 3508 (0.7...)	1324 / 3508 (3...)	0 / 3508 (0%)
Types	n/a	0	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)	0 / 0 (n/a)
Instances	11.3%	3508	2184 / 3508 (62.3%)	25 / 3508 (0.7...)	1324 / 3508 (3...)	0 / 3508 (0%)
top	11.3%	3508	2184 / 3508 (62.3%)	25 / 3508 (0.7...)	1324 / 3508 (3...)	0 / 3508 (0%)
coins	0%	866	866 / 866 (100%)	18 / 866 (2.1%)	0 / 866 (0%)	0 / 866 (0%)
coins1	0%	866	0 / 866 (0%)	0 / 866 (0%)	866 / 866 (10...)	0 / 866 (0%)
diag	3.8%	160	84 / 160 (52.5%)	0 / 160 (0%)	76 / 160 (47.5...)	0 / 160 (0%)
drinks	0%	382	382 / 382 (100%)	7 / 382 (1.8%)	0 / 382 (0%)	0 / 382 (0%)

Functional Safety Flow: Barrier Analysis Details



- Barrier Analysis executed on UU Faults to debug/identify block points
- Xcelium Safety supports barrier Analysis “-fault_barrier” switch to dump the data in Fault DB for every Fault Simulation
- Cadence developed Python utility is executed on fault_db to generate two files faults.csv and barrier.csv
 - **barrier.csv** -> captures the barriers and the associated blocked faults
 - contains the instance ; file name and line number of the code which block the fault propagation
 - **faults.csv** -> contains fault set and associated barriers for each of the fault nodes

Snippet of barriers.csv (Barrier to Fault Relation)

```
Barrier ID,Barrier Node,FanIn Strength,Faults
1,test_drink.top.coins.g1824__4547.Y,2,{1 2}
2,test_drink.top.coins.g1823__1474.Y,2,{1 2}
3,test_drink.top.coins.g1822__3772.Y,2,{1 2}
4,test_drink.top.coins.g2634__7675.CO,1,{2}
5,test_drink.top.coins.g2588__1474.Y,1,{2}
```

Snippet of faults.csv (Faults to Barrier Mapping)

```
Fault ID,Fault Node,Fault Type,Fault Injection Time,FanOut Strength,Barriers
1,test_drink.top.coins.RC_CG_HIER_INST1.RC_CGIC_INST.E,SA1,402NS,3,{1 2 3}
2,test_drink.top.coins.RC_CG_HIER_INST1.RC_CGIC_INST.ECK,SA1,402NS,5,{1 2 3 4 5}
```

Waveform Debug Incremental Campaign FCM integration

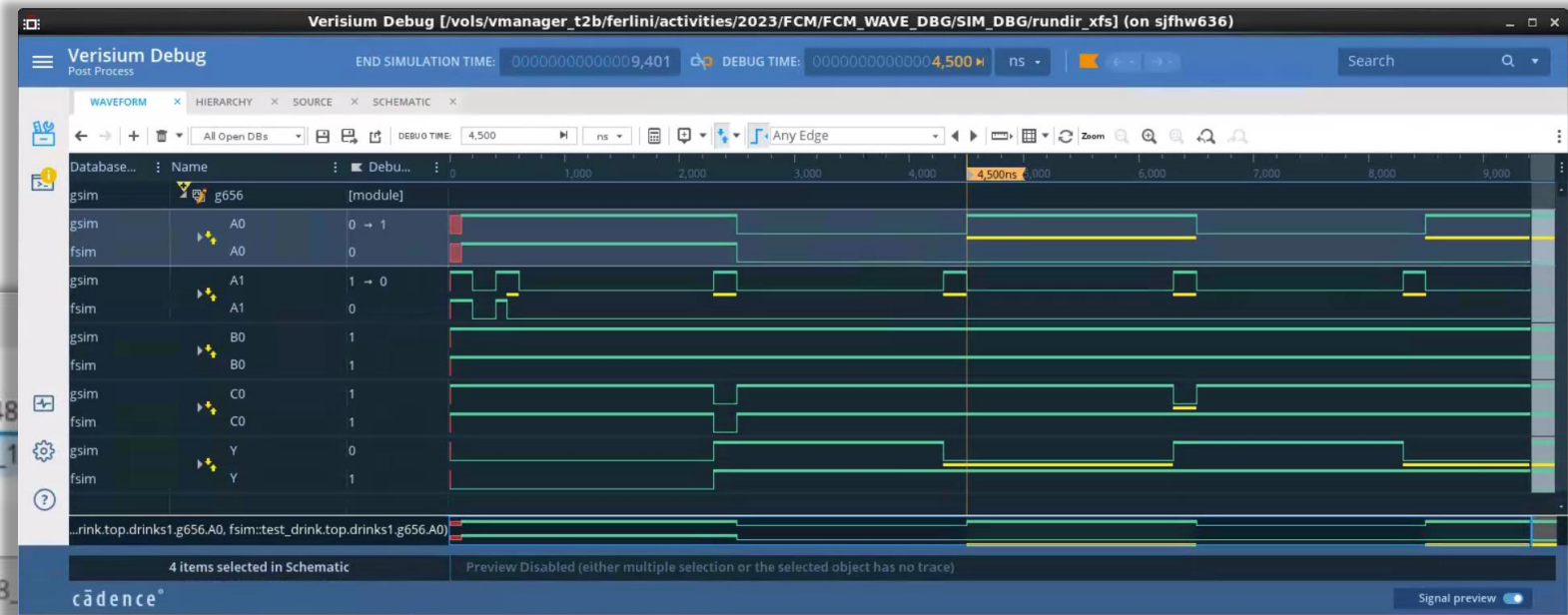
Flow Sessions

Session Status	Name
(no filter)	ve_multi.HYBRID.ferlini.2023_10_05_13_08_48
completed	dbg_inc_wave_multi.HYBRID.ferlini.2023_10_05_13_08_48

Showing 1 out of 245 items

Sessions | dbg_inc_wave_multi.HYBRID.ferlini.2023_10_05_13_08_48

Session Status	Name	Total Runs	#Passed
(no filter)	(no filter)	(no filter)	(no filter)
completed	dbg_inc_wave_multi.good_simulation.ferlini.2023_10_05_13_08_48	1	1
completed	dbg_inc_wave_multi.fault.ferlini.2023_10_05_13_08_48	7	7



Default | Runs Query | Metrics | Tests | vPlan | Reload vPlan | Reload Coverage | Refresh Runs | Scripts Manager | VDebug | Failures | Runs | Formal Prop. | Correlate Runs | Rank Runs | Edit all at once | Edit each | Edit all Runs | History | Rerun | Create Context | Stop Run | Open dir | Stop Sessions | Report | Help

Views | Context Operations | Scripts | Invoke Verisium Debug with good and fault sim | Runs | List Tabs | /tests/all_Faults_Group_4

Index	Name	Status	Duration (sec.)
1	/tests/all_Faults_Group_0	passed	4
2	/tests/all_Faults_Group_1	passed	4
3	/tests/all_Faults_Group_2	passed	4
4	/tests/all_Faults_Group_3	passed	4
5	/tests/all_Faults_Group_4	passed	4
6	/tests/all_Faults_Group_5	passed	4
7	/tests/all_Faults_Group_6	passed	4

Fault Node	Fault Type	Fault Annotation	Fault Inject Time
top.drinks1.g656.A1	sa0	DD	500ns



Digital Safety Verification Summary

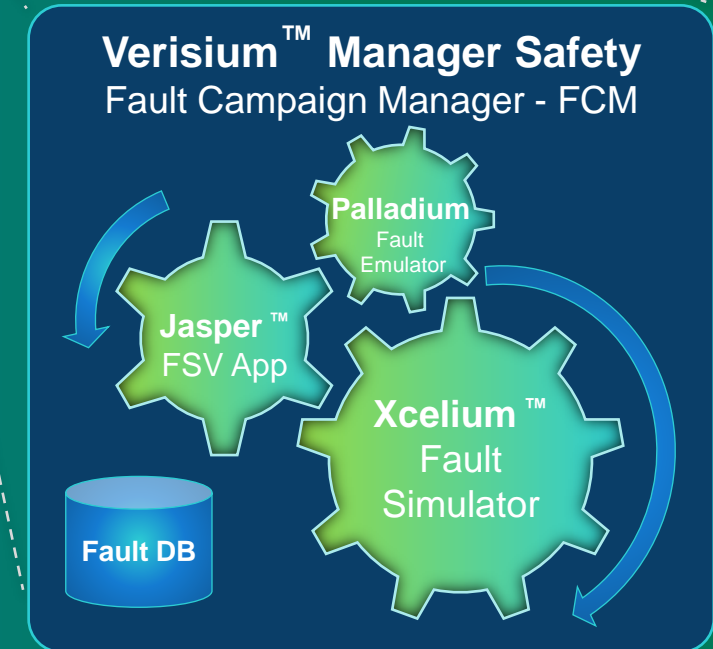
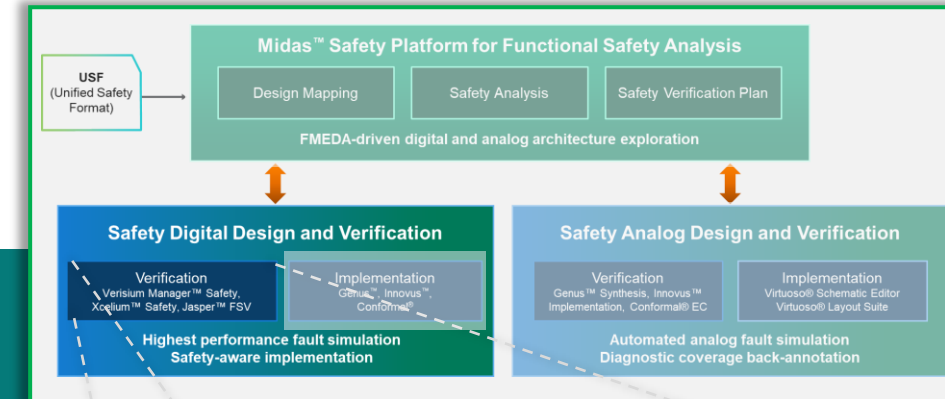
✓ Fault Campaign Automation

- ✓ Same verification environment (Verisium Manager add-on)
- ✓ Single front-end campaign configuration
- ✓ Jasper and both Xcelium fault engines orchestration
 - ✓ Data exchange via the proprietary unified fault database
- ✓ Dedicated fault coverage analysis (GUI and reports)

✓ Multi-Domain Fault Analysis support

- ✓ Permanent and Transient fault campaigns
- ✓ Diagnostic Coverage and Safeness
 - ✓ Software-based Self-Test Library (STL) assessment
 - ✓ Safety Mechanism (integration) Verification (+Detection Time Interval)
- ✓ Fault / Test Grading (DFT) + Architectural Vulnerability (RadHard)

✓ ISO26262 tool qualification – up to ASIL D

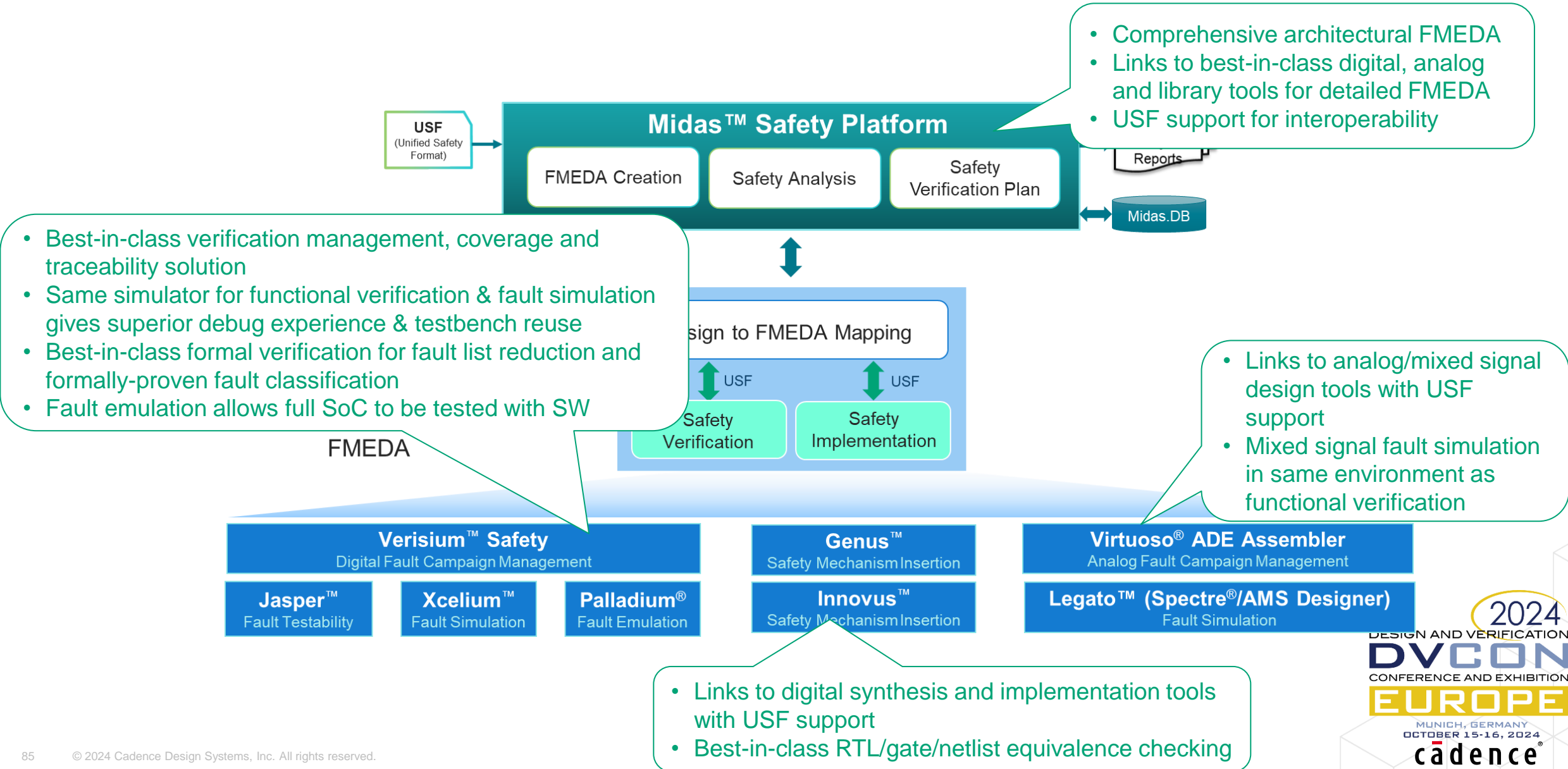




Summary



Advantages of the Cadence Functional Safety Solution



- Comprehensive architectural FMEDA
- Links to best-in-class digital, analog and library tools for detailed FMEDA
- USF support for interoperability

- Best-in-class verification management, coverage and traceability solution
- Same simulator for functional verification & fault simulation gives superior debug experience & testbench reuse
- Best-in-class formal verification for fault list reduction and formally-proven fault classification
- Fault emulation allows full SoC to be tested with SW

- Links to analog/mixed signal design tools with USF support
- Mixed signal fault simulation in same environment as functional verification

- Links to digital synthesis and implementation tools with USF support
- Best-in-class RTL/gate/netlist equivalence checking





cādence[®]

© 2024 Cadence Design Systems, Inc. All rights reserved worldwide. Cadence, the Cadence logo, and the other Cadence marks found at <https://www.cadence.com/go/trademarks> are trademarks or registered trademarks of Cadence Design Systems, Inc. Accellera and SystemC are trademarks of Accellera Systems Initiative Inc. All Arm products are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All MIPI specifications are registered trademarks or service marks owned by MIPI Alliance. All PCI-SIG specifications are registered trademarks or trademarks of PCI-SIG. All other trademarks are the property of their respective owners.