2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE
MUNICH, GERMANY
OCTOBER 15-16, 2024

USF-based FMEDA-driven Functional Safety Verification

Francesco Lertora, Software Engineering Group Director, SVG
Frederico Ferlini, Sr. Principal Product Engineer, SVG

15 September 2024

cadence®

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

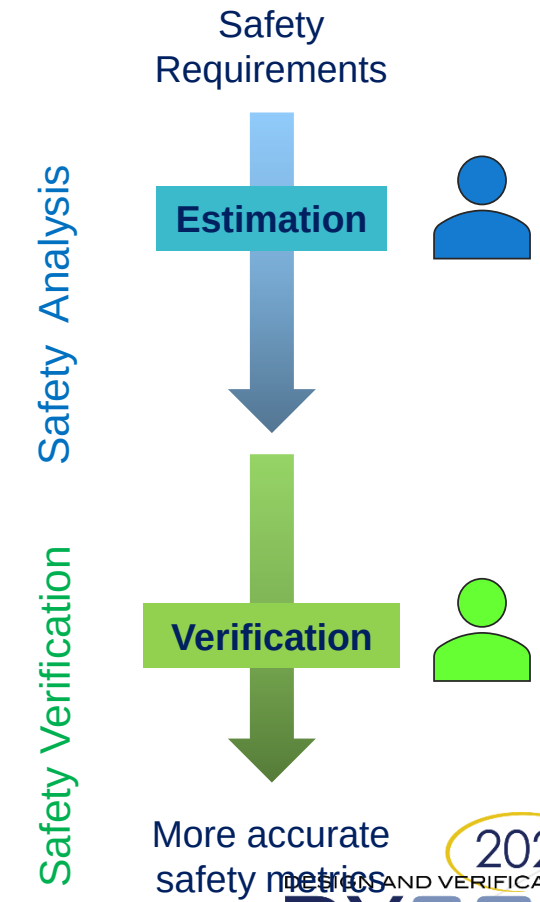
- Lack of formalism, standards ambiguity, differentiated assessors approach, lead to customer-specific methodologies + widespread usage of Spreadsheets
  - «consulting-driven» market side-effects:
    - ‘keep it obscure’
    - ‘this is *my* (certified) methodology’
    - ‘(only) We will tell you what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

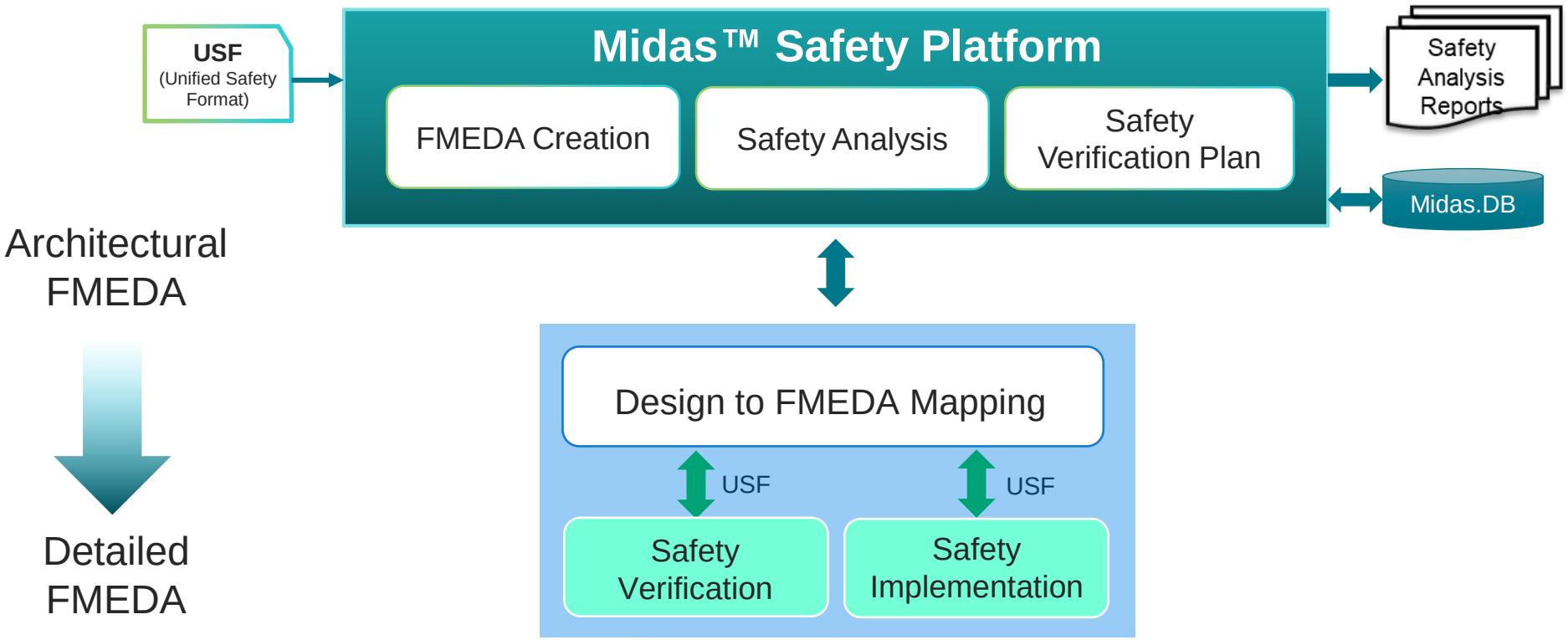
- Cadence is committed to adopt and support the IEEE 2851 family of standards

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



Verisium™ Safety
Digital Fault Campaign Management

Jasper™
Fault Testability

Xcelium™
Fault Simulation

Palladium®
Fault Emulation

Genus™
Safety Mechanism Insertion

Innovus™
Safety Mechanism Insertion

Virtuoso® ADE Assembler
Analog Fault Campaign Management

Legato™ (Spectre®/AMS Designer)
Fault Simulation

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

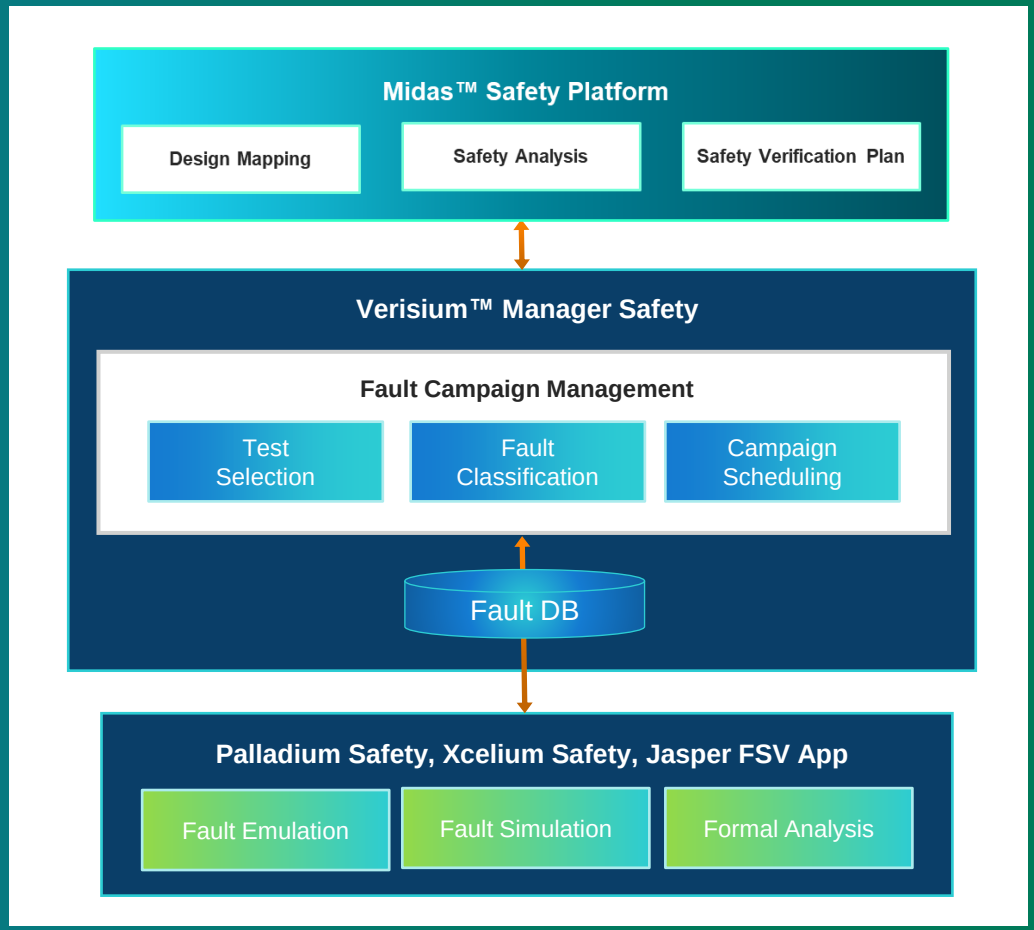
- Structural analysis to reduce the fault list
- Formal analysis for accurate fault classification

Fault Simulation – Xcelium Safety

- Native serial and concurrent fault verification
- Same simulator for functional verification (GOOD machine) and fault simulation (BAD machine)

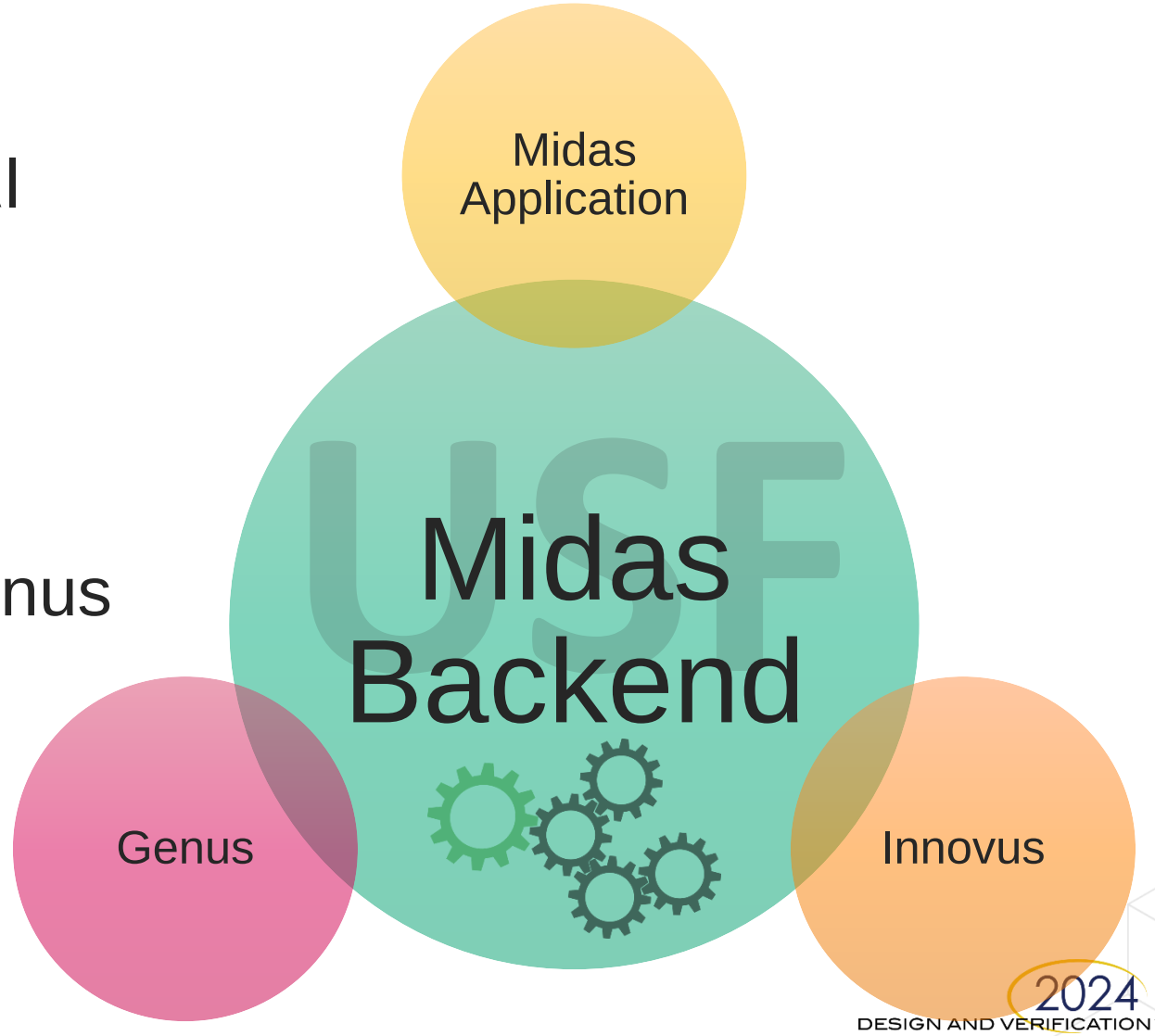
Fault Emulation – Palladium Safety

- Run full SoC with SW or STLs



Midas Safety Platform Modularity

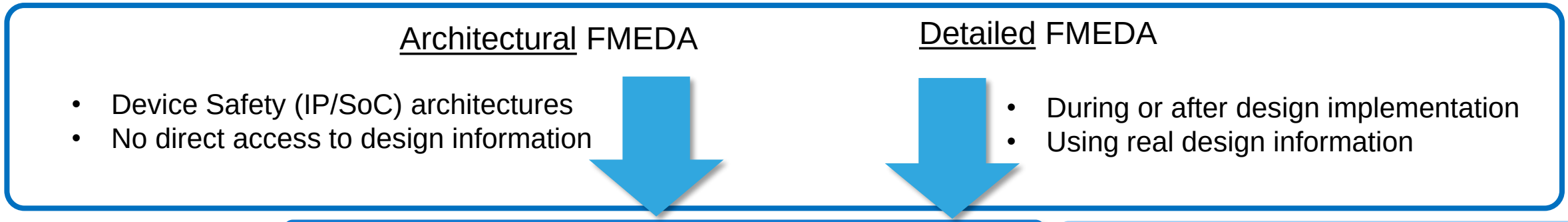
- The Midas backend is the ‘functional safety engine’
  - Support for Midas command line interface
  - ISO26262; IEC61508
  - BFR
- Same backend is integrated into Genus and Innovus
- Core features can be made easily available in different contexts



Functional Safety Analysis Overview

2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE
MUNICH, GERMANY
OCTOBER 15-16, 2024
cadence

Functional Safety Analysis



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)
- 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 -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}
```

- 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

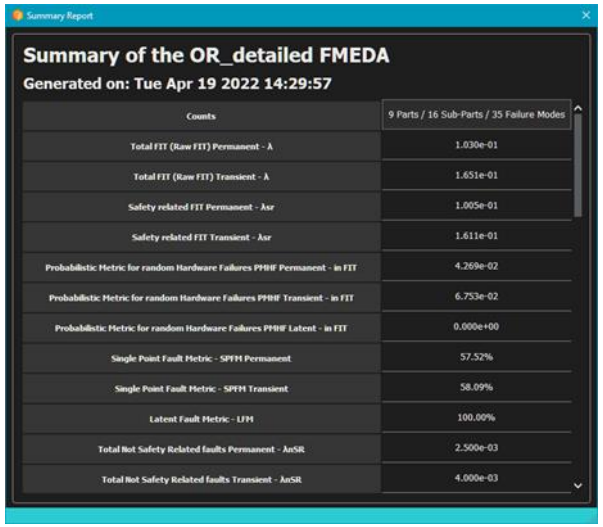
```
create_safety_mechanism SM-IF -desc "Instruction Fetch
redundancy" -type Custom -class HW
apply_safety_mechanism  SM-IF -to FM_ARCH_1 -fmeda myFMEDA -
dcperm 95 -dctrans 0 -dclat 100
```

- 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
```

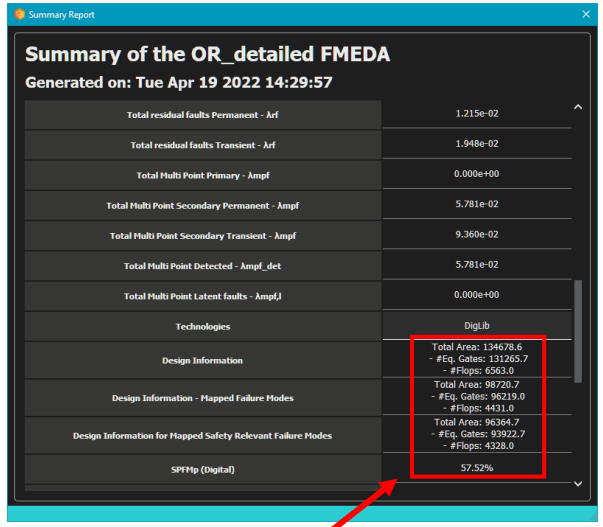
Refine FMEDA Data for Optimized Safety Design

Architectural FMEDA



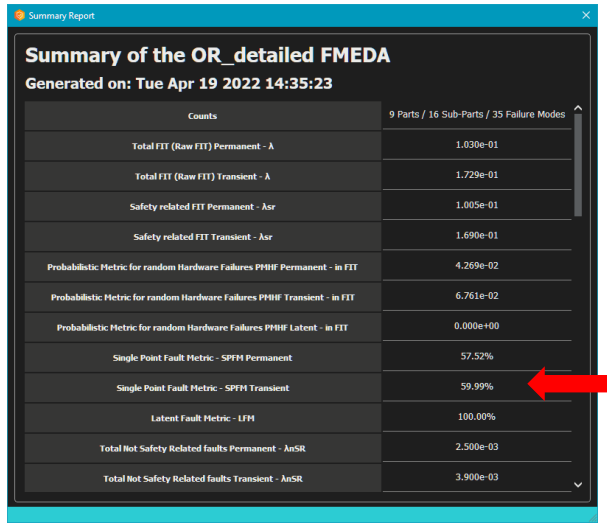
- No design data available
- FMEDA hierarchy only
- Failure rates and distribution solely based on early estimations

Detailed FMEDA



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

Optimized Safety Design



2% higher SPFM

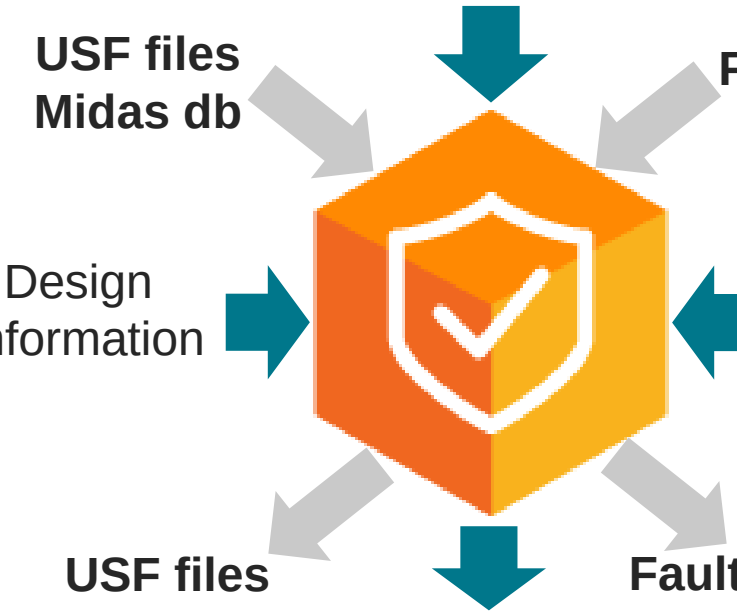
- With design & simulation data
- Design to FMEDA hierarchy mapping
- HW safety metric based on design & simulation data

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



USF files
Midas db

Fault Simulation Results

Estimated

Design Information

Xcelium

Genus

Spectre

dhedb → Real

Reliability

Basic Failure Rates by technology

Reference Areas

USF files
Midas db

Fault Simulation Setup (Campaing Order)

Reports

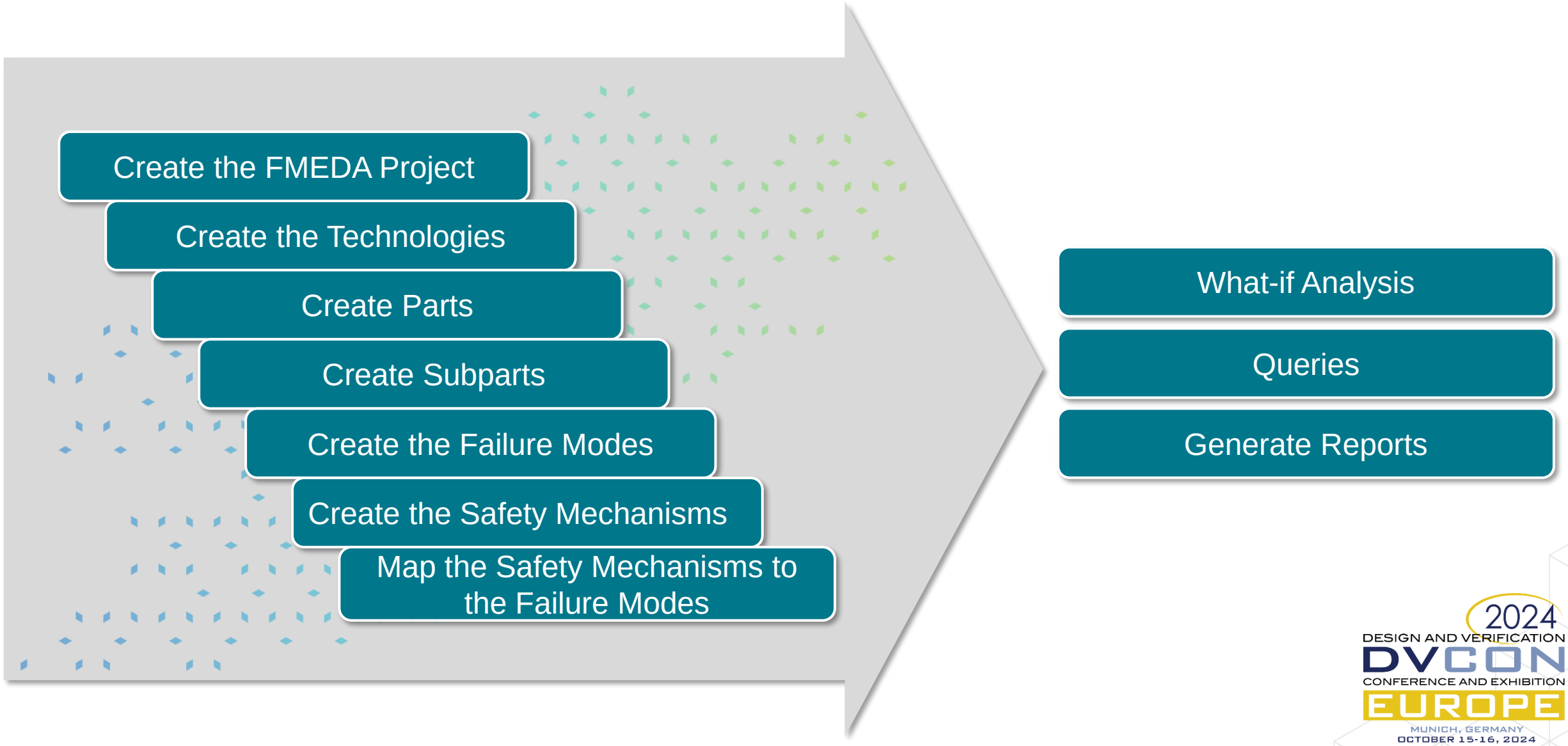
- FMEDA (Permanent+Transient)
- Summary
- SoC Report

DEEP DIVE

Architectural FMEDA

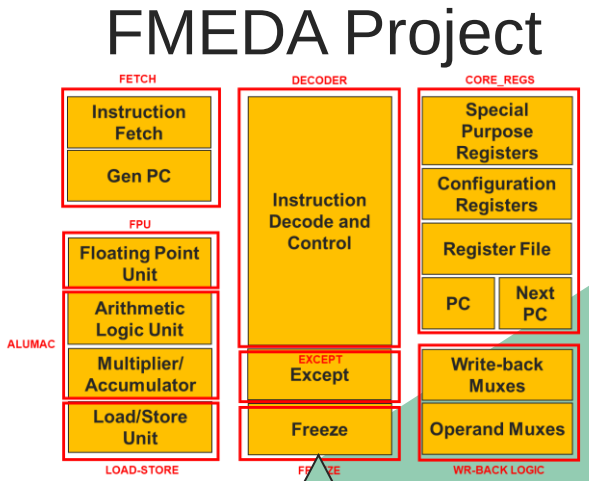
2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE
MUNICH, GERMANY
OCTOBER 15-16, 2024
cadence

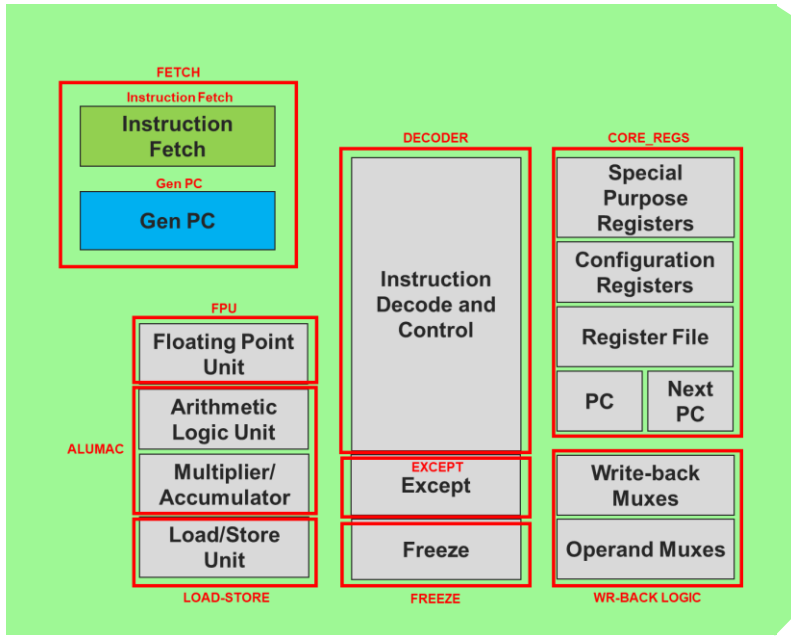
Architectural FMEDA Authoring Steps

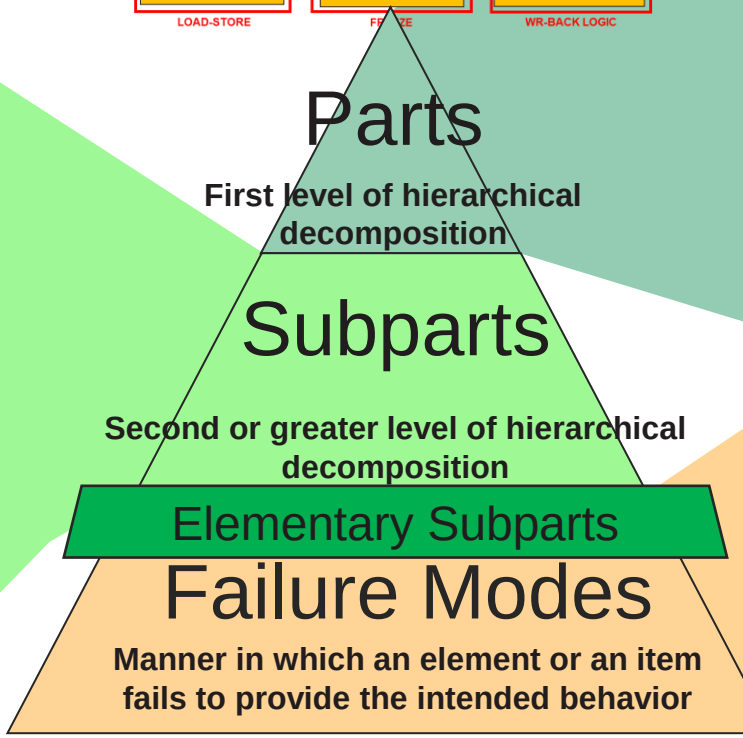


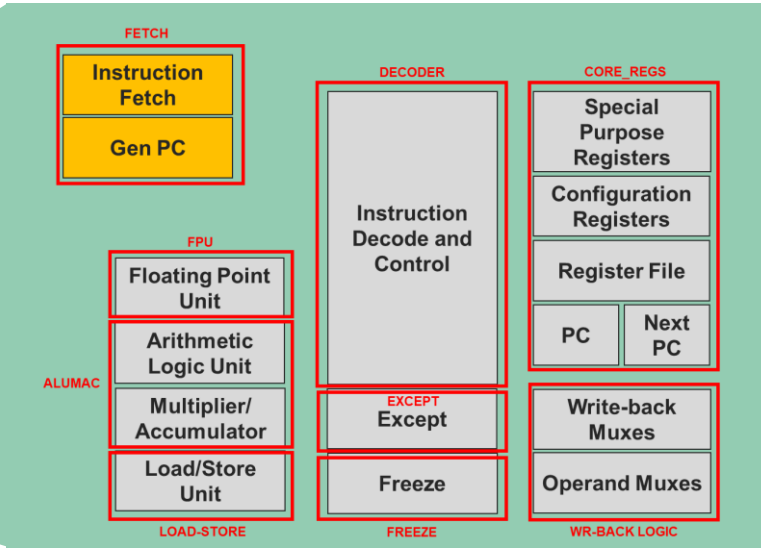
Design Decomposition

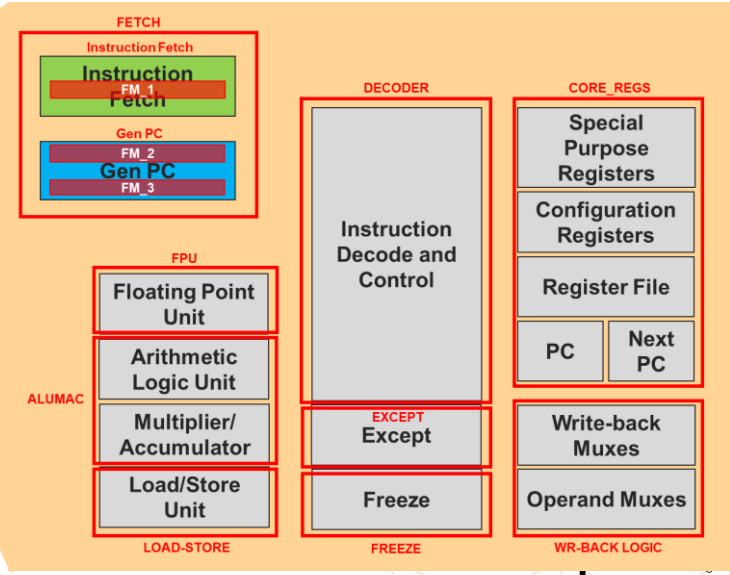
- Safety analysis are typically performed with a reduced number of hierarchical levels compared with the design hierarchy







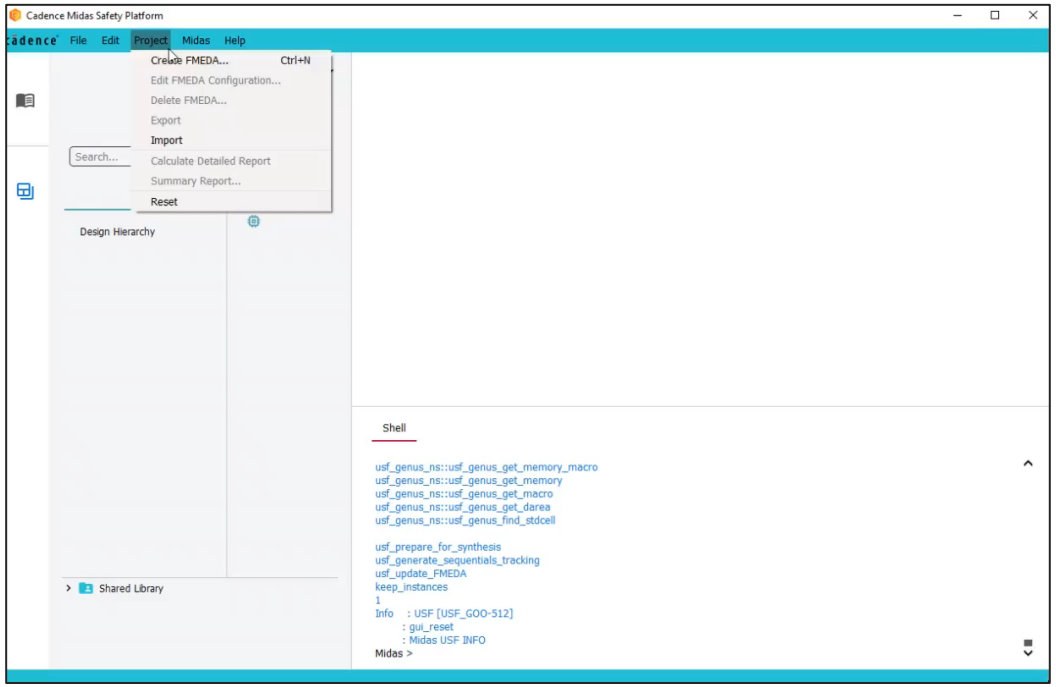


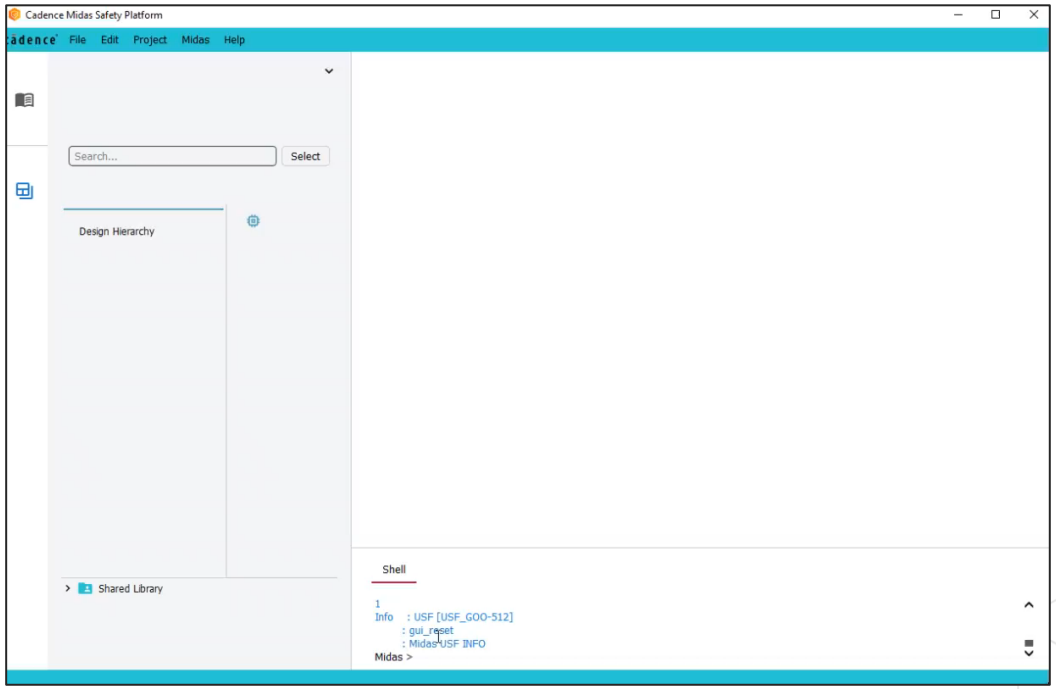


- The target is to define the failure modes, not to describe the circuit functionalities

Functional Safety Authoring

- The GUI provides an user-friendly FMEDA authoring enviroment
- Safety objects can also be created with USF commands

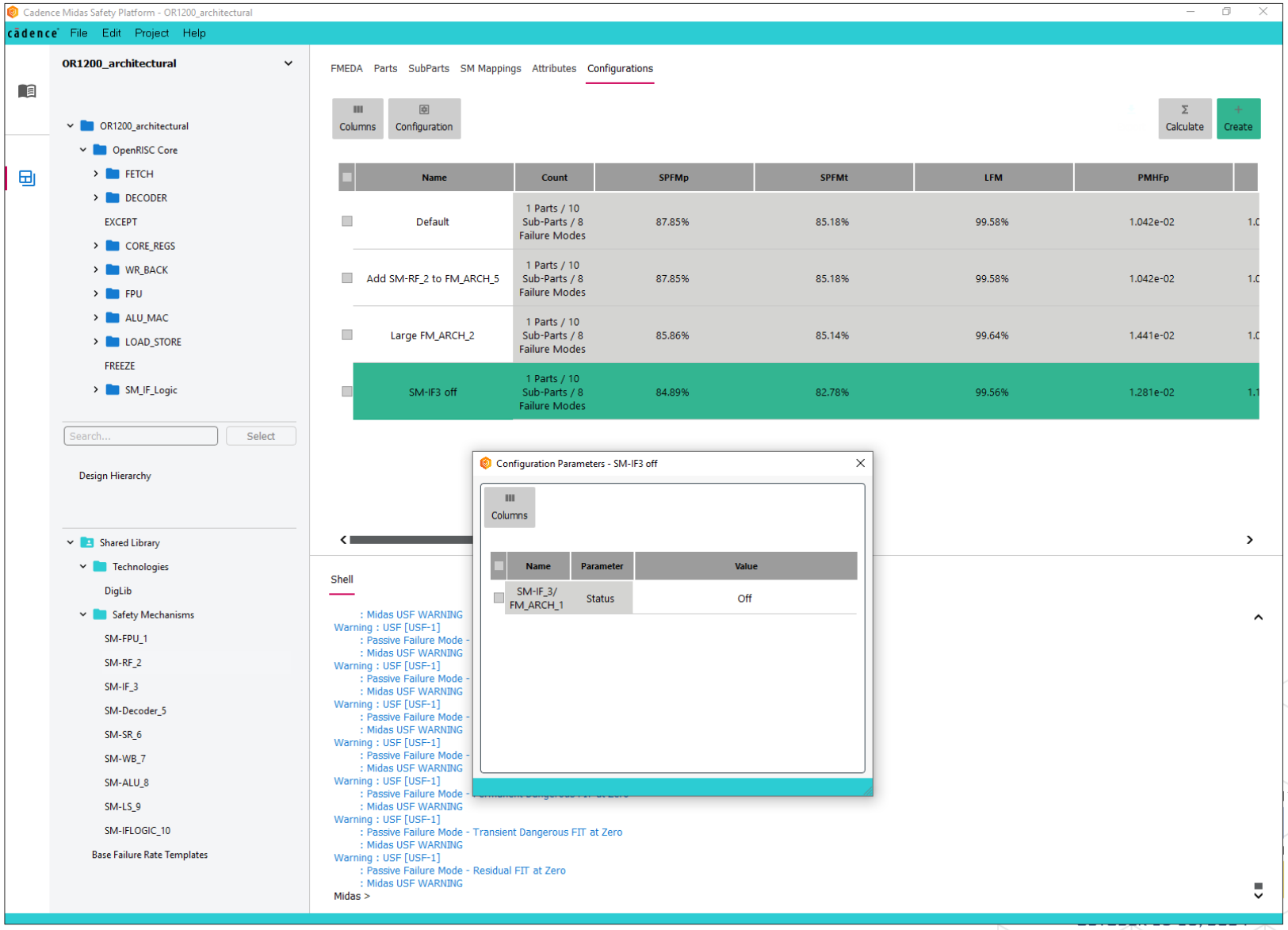




- The solution is fully scriptable
- Mixing GUI and scripted-automations is further possibile

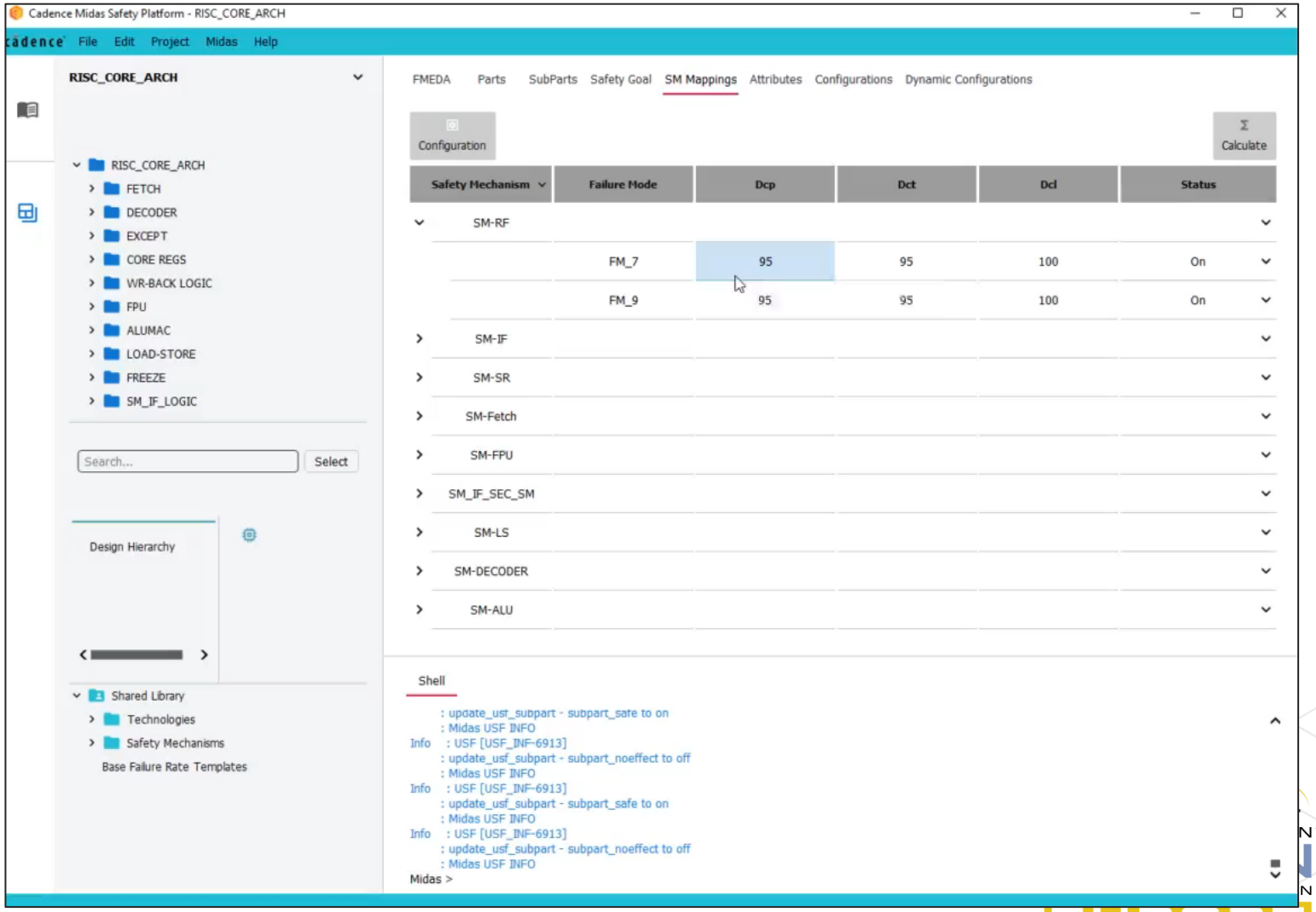
What-if Analysis: FMEDA Static Configurations

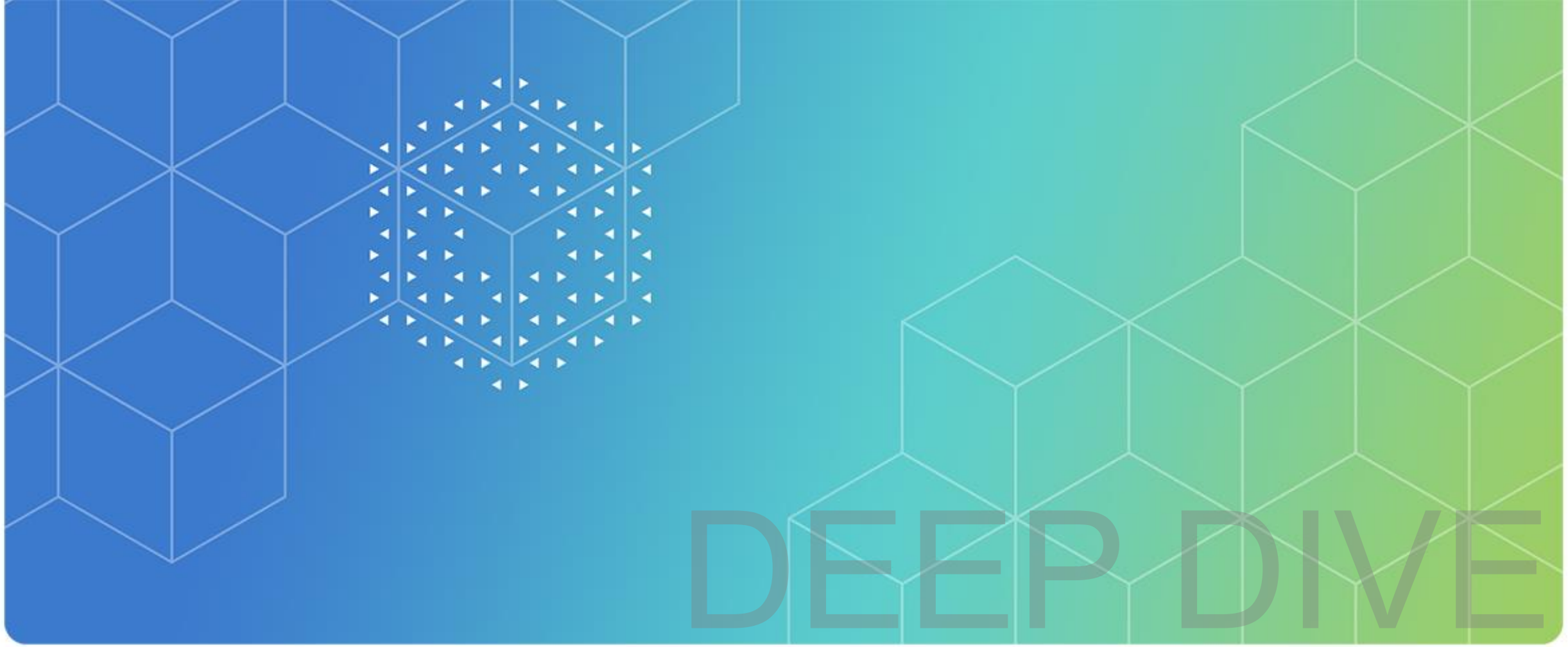
- Create configurations changing values in the FMEDA (e.g., design info., SM DCs)
- Each configuration generates safety metrics to be compared
- The configurations can be saved and restored



What-if Analysis: FMEDA Dynamic Configurations

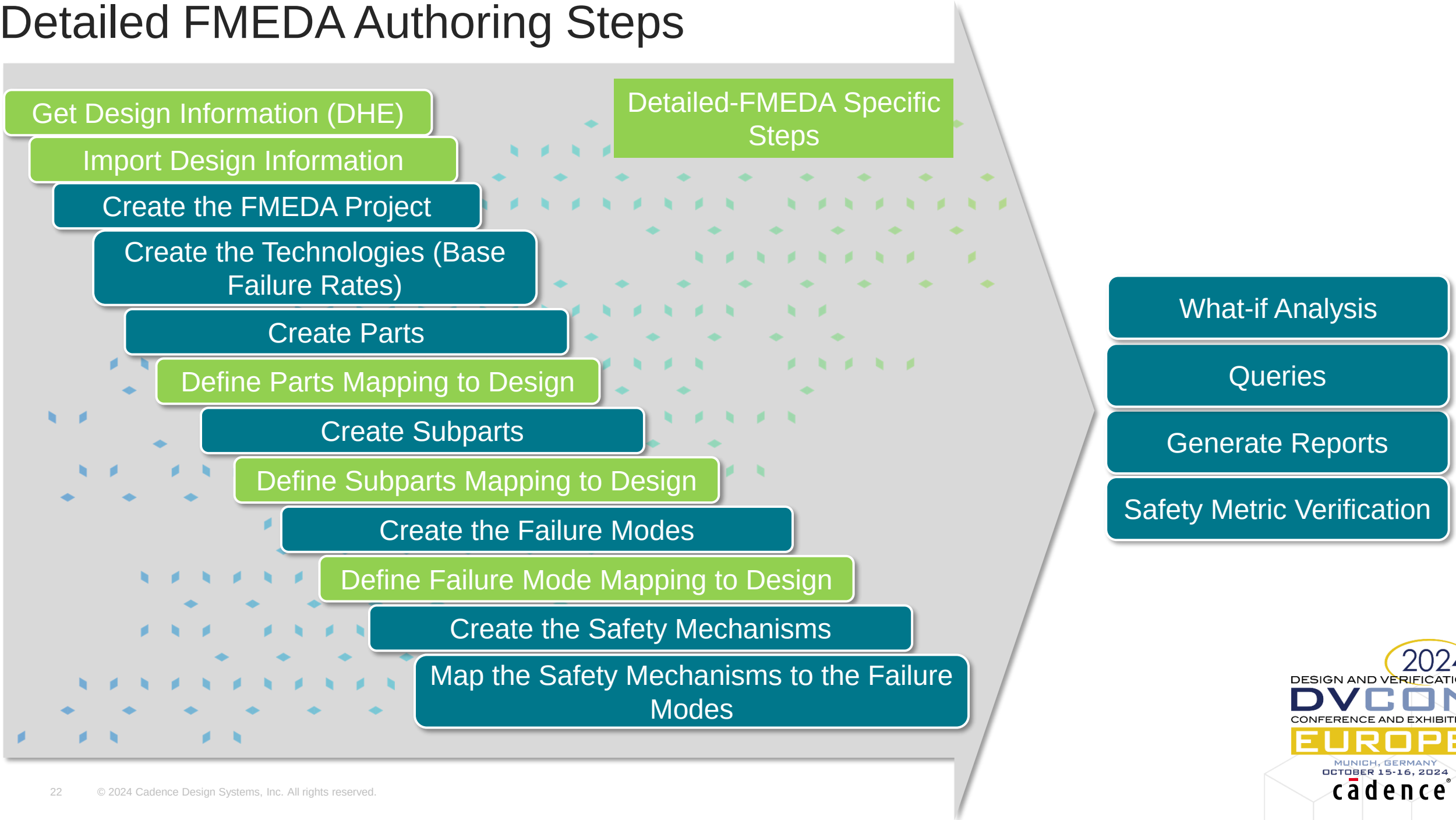
- It is possibile to select one parameter (e.g, DC), define the interval and an output metric to be reported
- By leveraging the USF backend Midas provides the result of the simulation
- Graphs, and values, can be saved and restored





Detailed FMEDA

Detailed FMEDA Authoring Steps



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
cfmom    : 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 :

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

set_bfr

die	package	overstress
set_IEC62380_lDIE	set_IEC62380_lPackage	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_cpackage; get_IEC62380_cpackage`

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

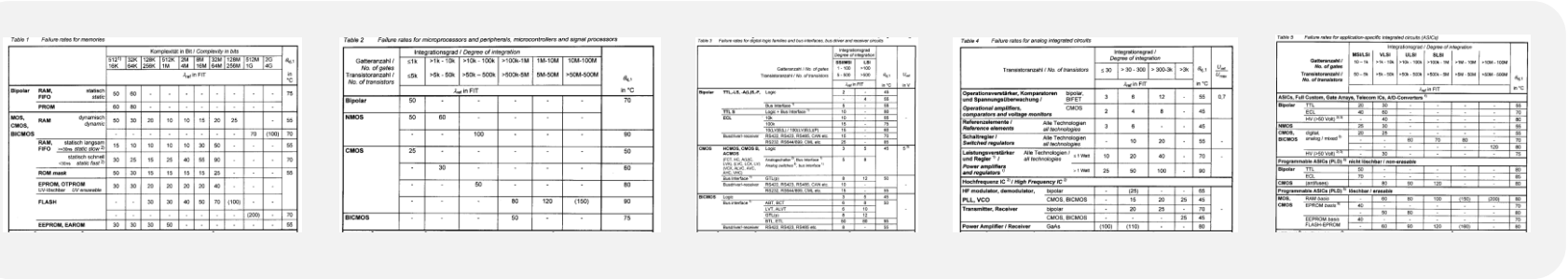
Drift sensitivity factor

$\lambda = \lambda_{ref} \times \pi_U \times \pi_T$ Digital CMOS-B

Temperature sensitivity factor

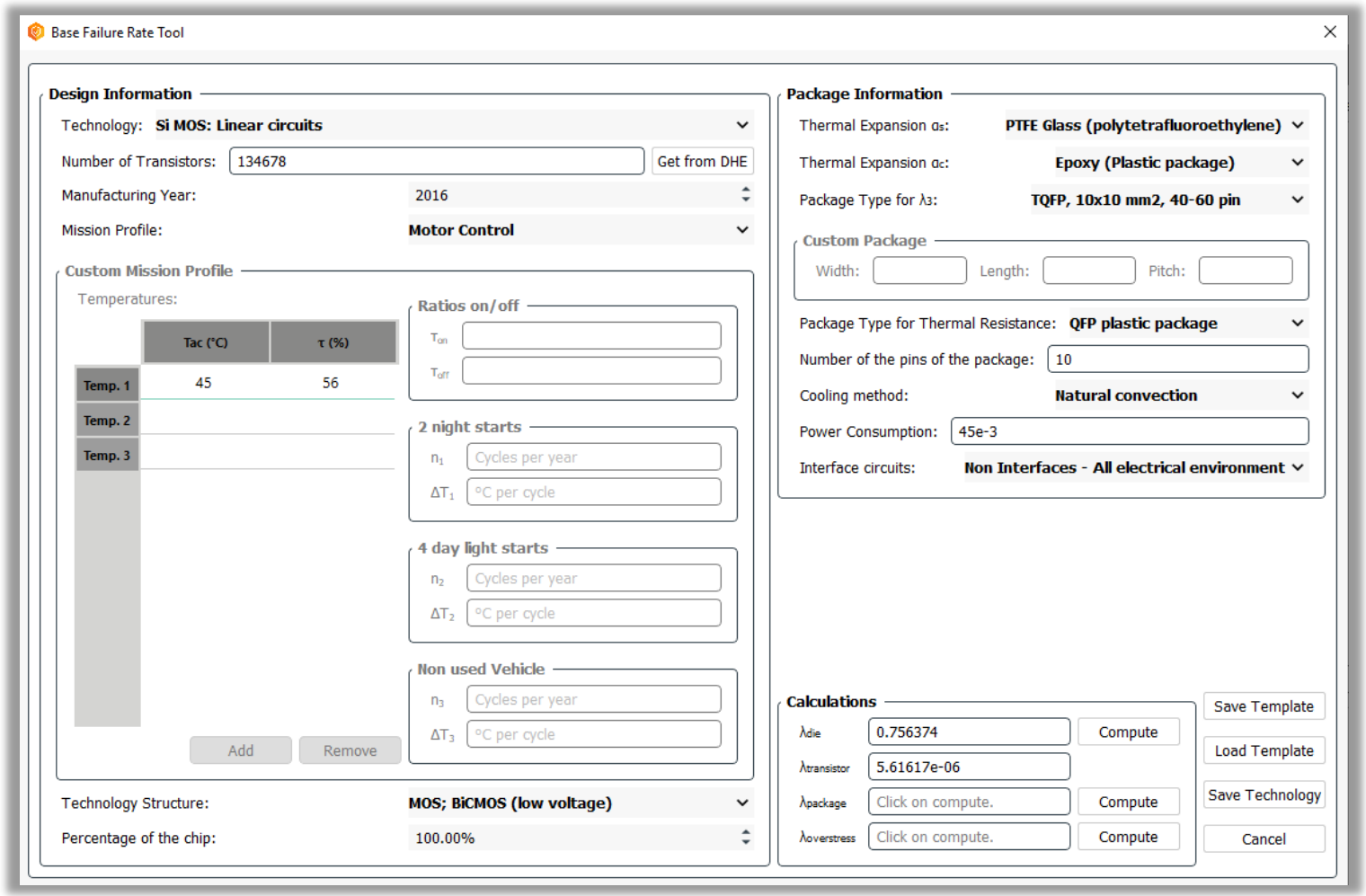
$\lambda = \lambda_{ref} \times \pi_T$ For all other integrated circuits

Voltage Dependence factor

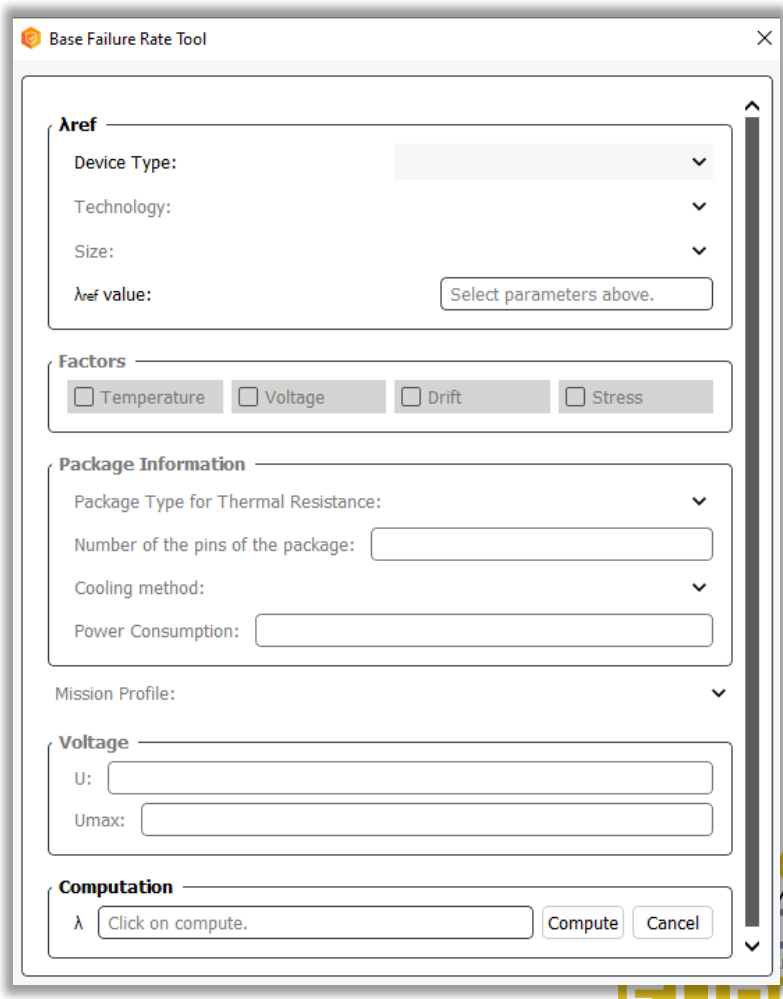


Midas GUI BFR Tools

ISO 26262-11 / IEC TR 62380

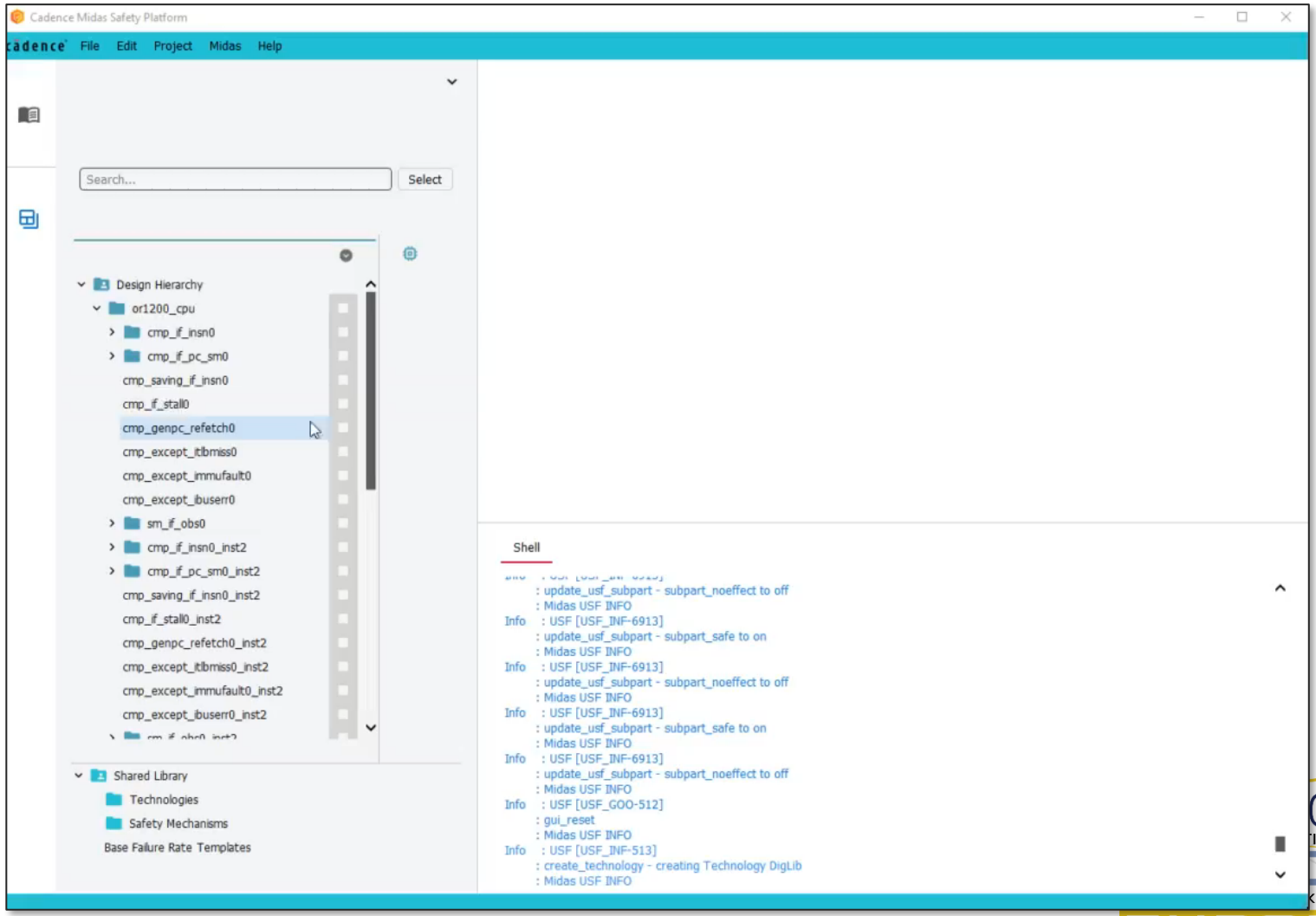


SN29500



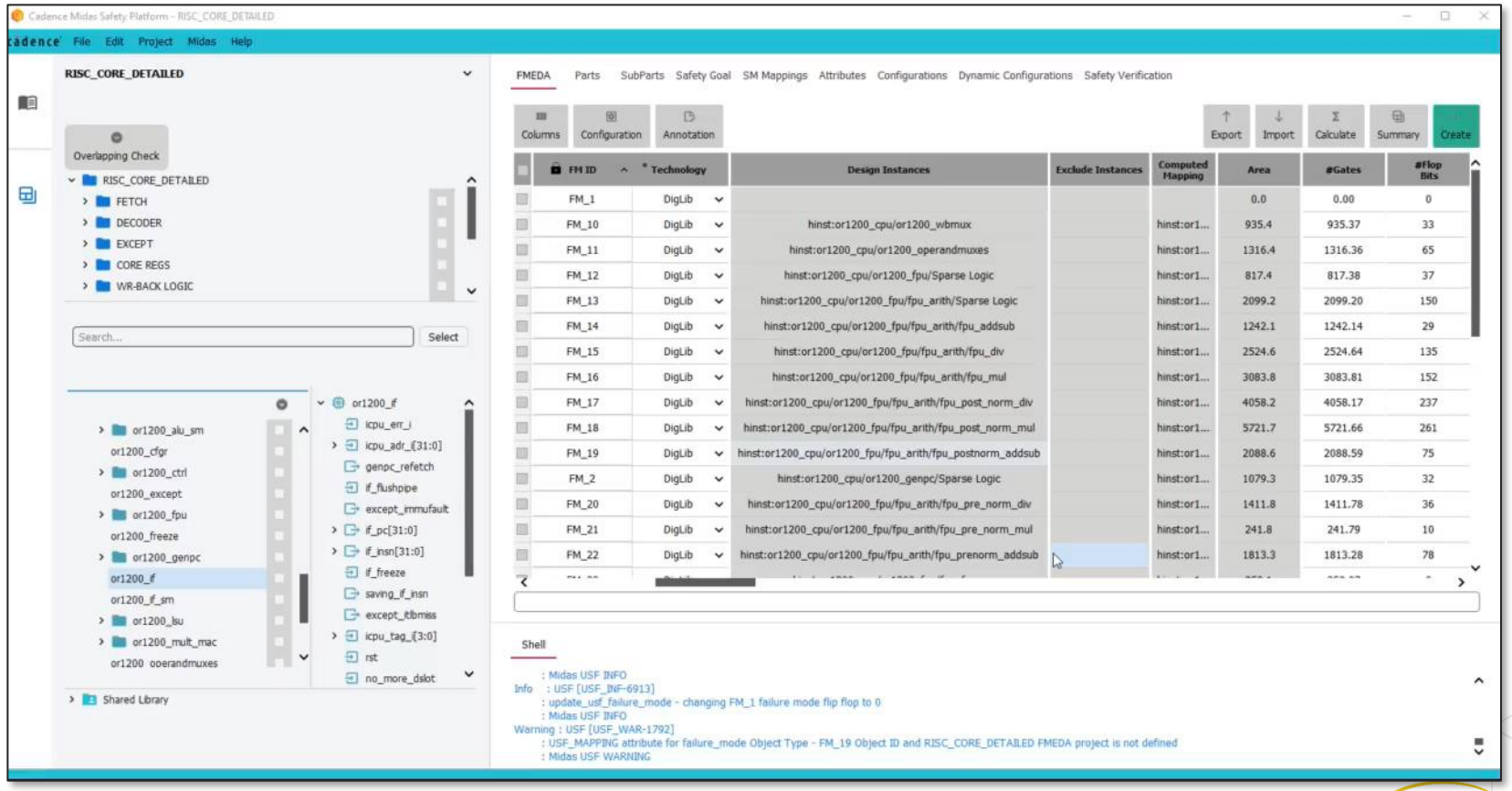
Leverage Design Information in the BFR Computation

- 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



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

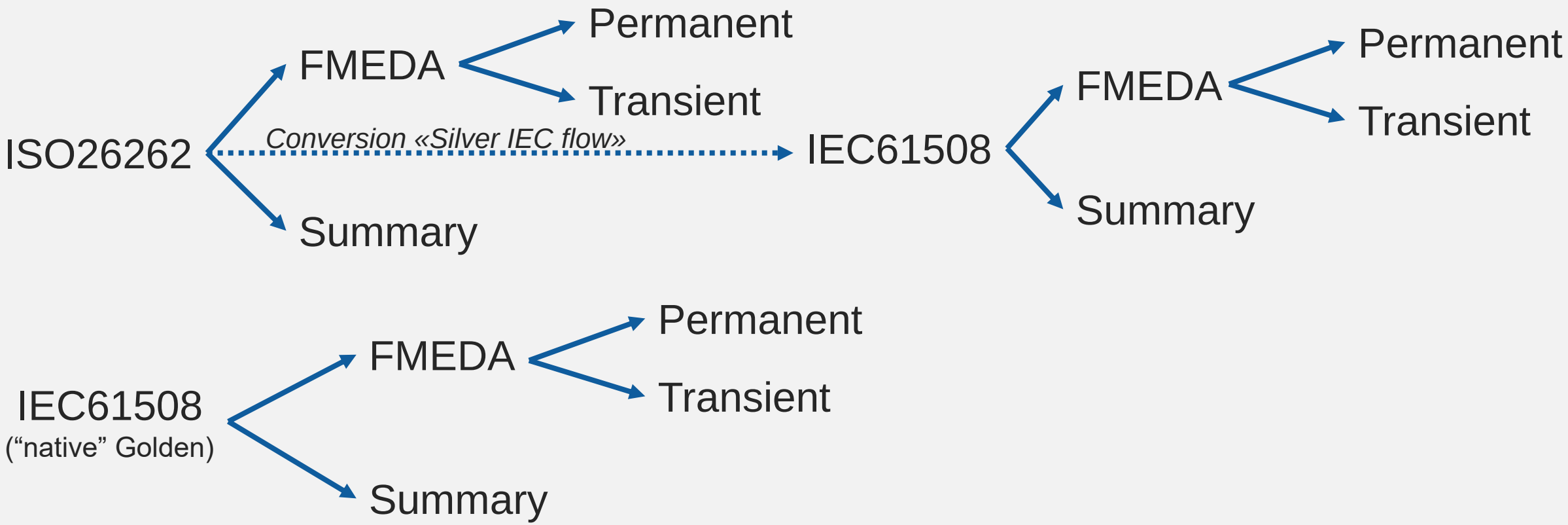
query_usf USF Relational Queries

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

LEVEL 0	query_usf *	Listing available information
LEVEL 1	query_usf {fmeda} {-obj_id id} {-obj_type type}	Direct query
LEVEL 2	query_usf {fmeda} {-obj_id id} {-obj_type type} [-ref_type RefType] [-ref_id refid]	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% -- -- --}

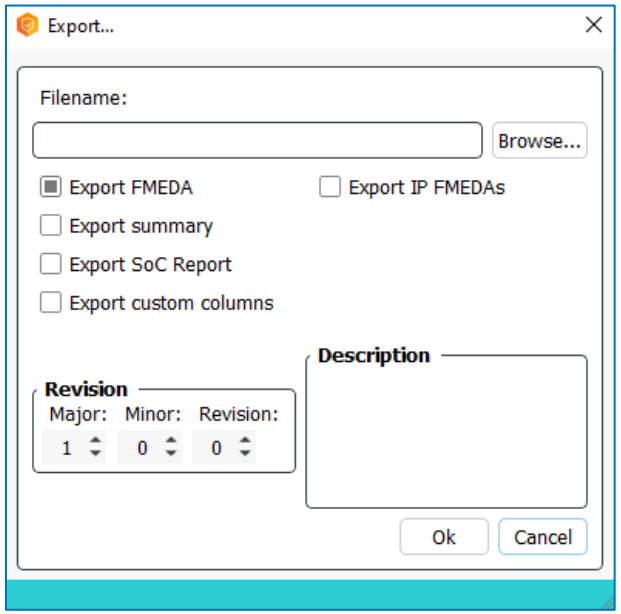
USF Reports: ISO26262 and IEC61508

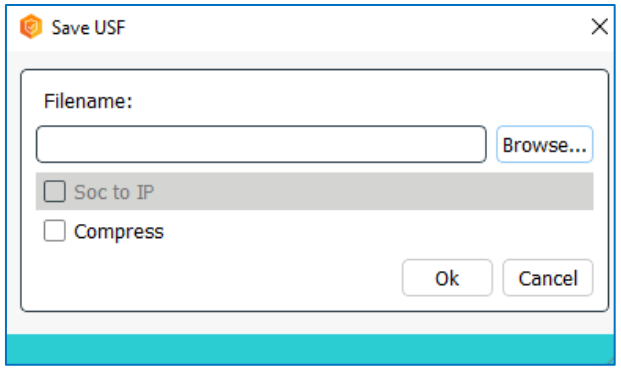


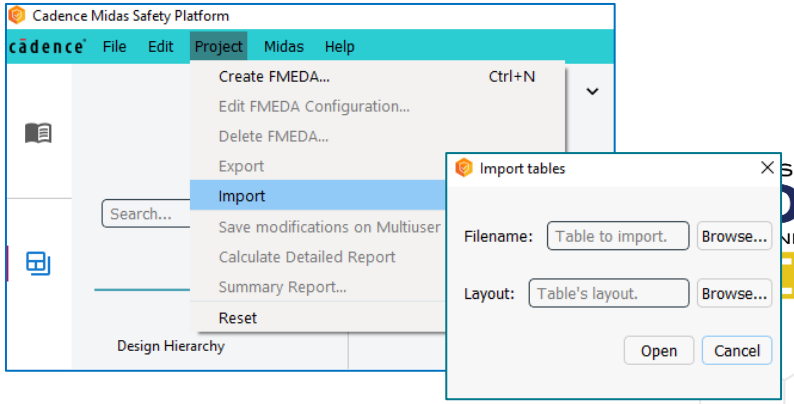

```
report_safety -fmeda myFMEDA permanent html "reports/ISO_PERMANENT.html"
report_safety -fmeda myFMEDA transient html "reports/ISO_TRANSIENT.html"
report_safety -fmeda myFMEDA report    html "reports/ISO_SUMMARY.html"

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"
```

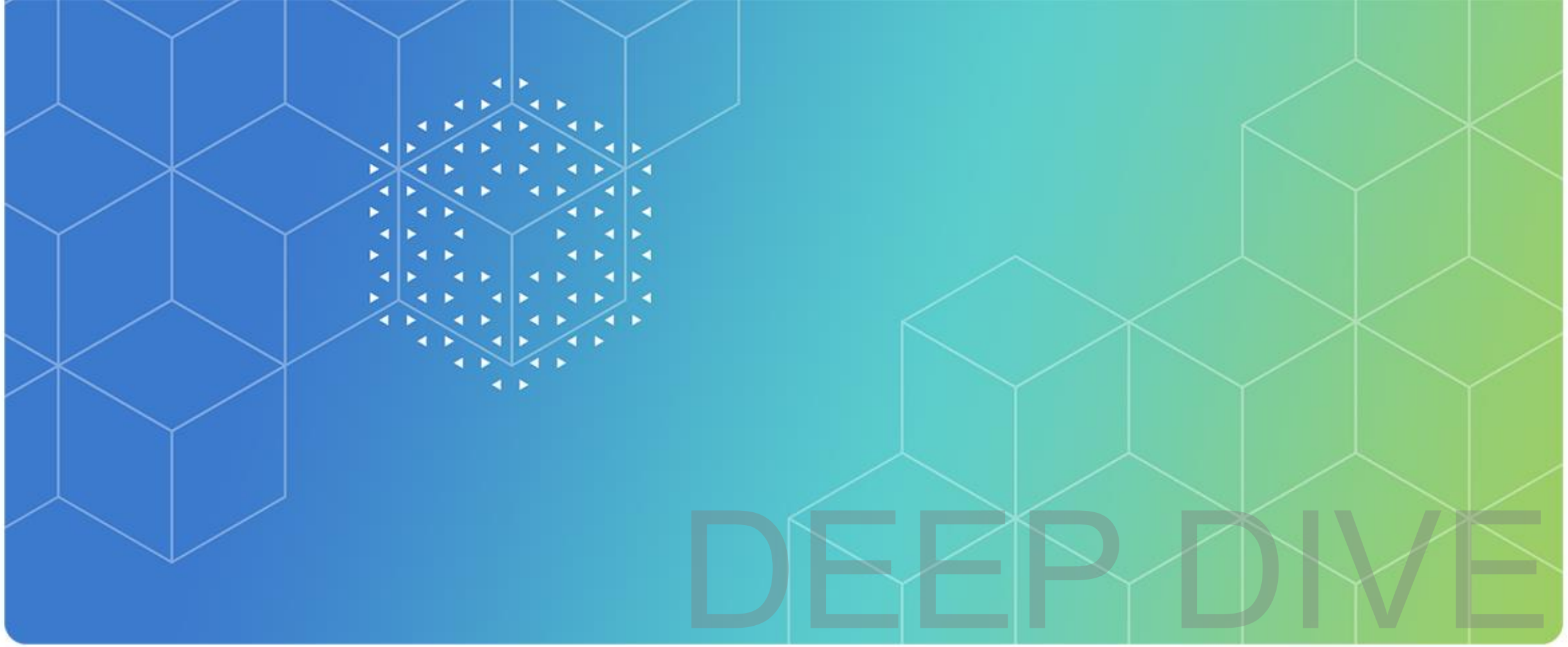
Midas Application Import-Export

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







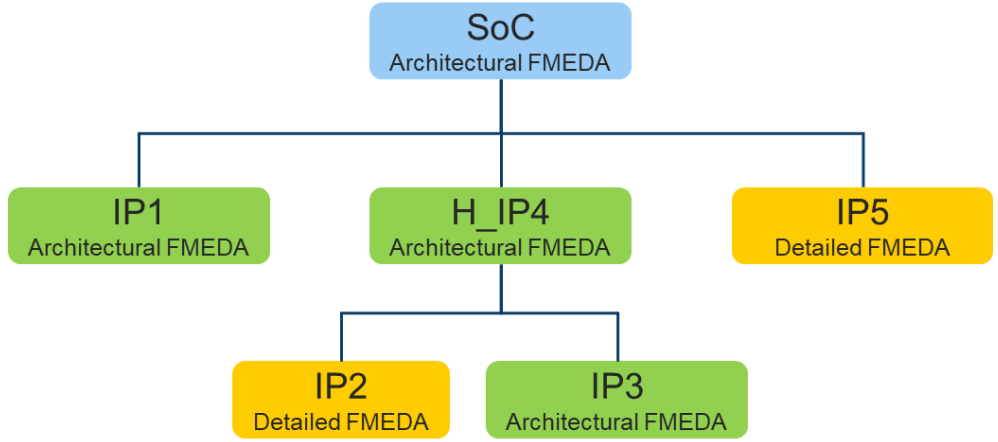


SoC Safety Analysis

2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE
MUNICH, GERMANY
OCTOBER 15-16, 2024
cadence

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
```

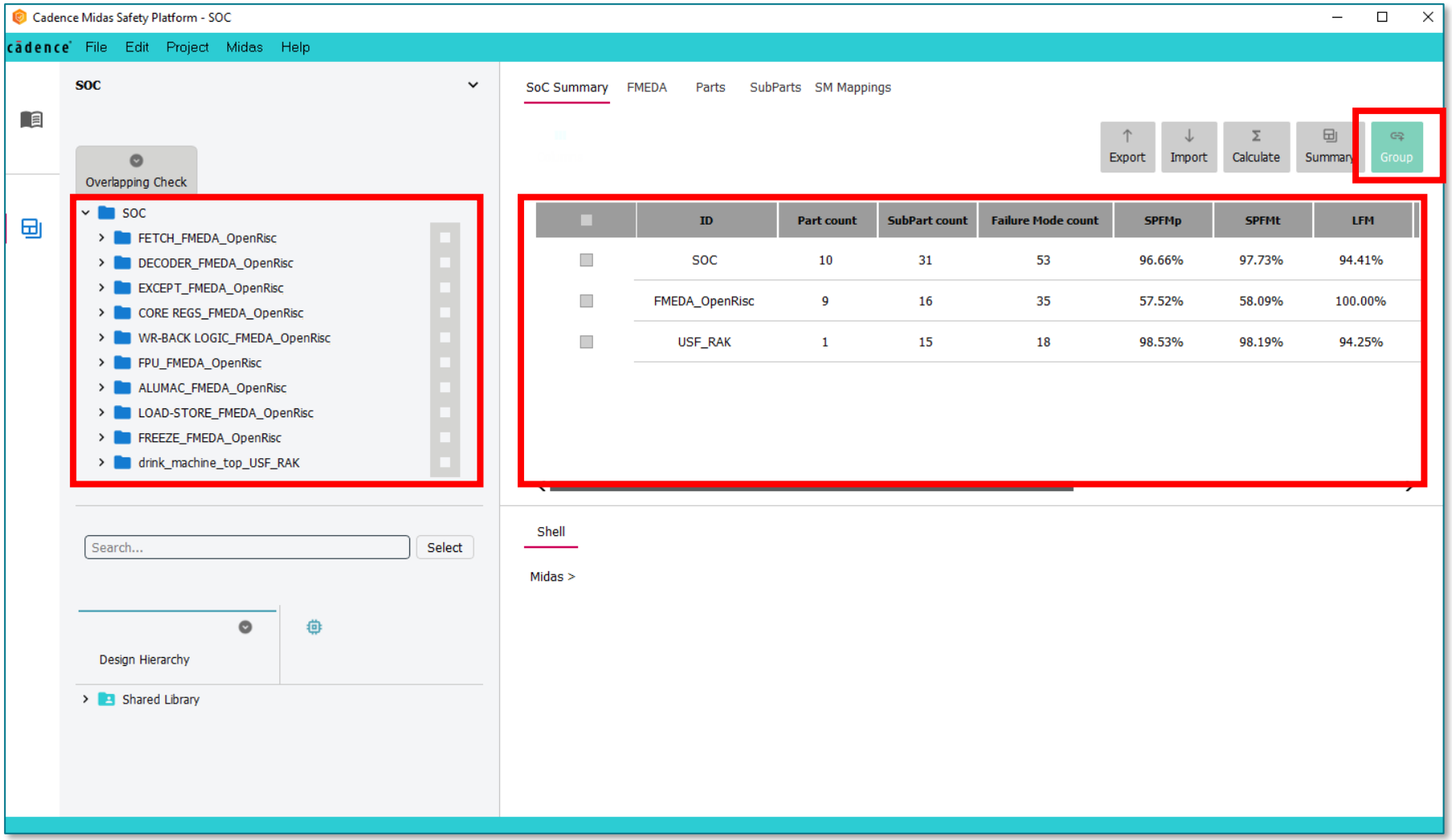
Grouping IP FMEDAs into a SoC FMEDA: USF Command

group_fmeda {-fmeda_list} [-fmeda_file] {-to fmeda_soc} [-linkonly]	
-fmeda_list FMEDA_tags_list	Specify a list of FMEDA to link to a SoC FMEDA. With the format FMEDAIP(*num_replica*), automatically creates replicas of the same FMEDA
-fmeda_file 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 save_usf commands.
-to 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 set_fmeda command using the -soc option.
-linkonly	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



- 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
```

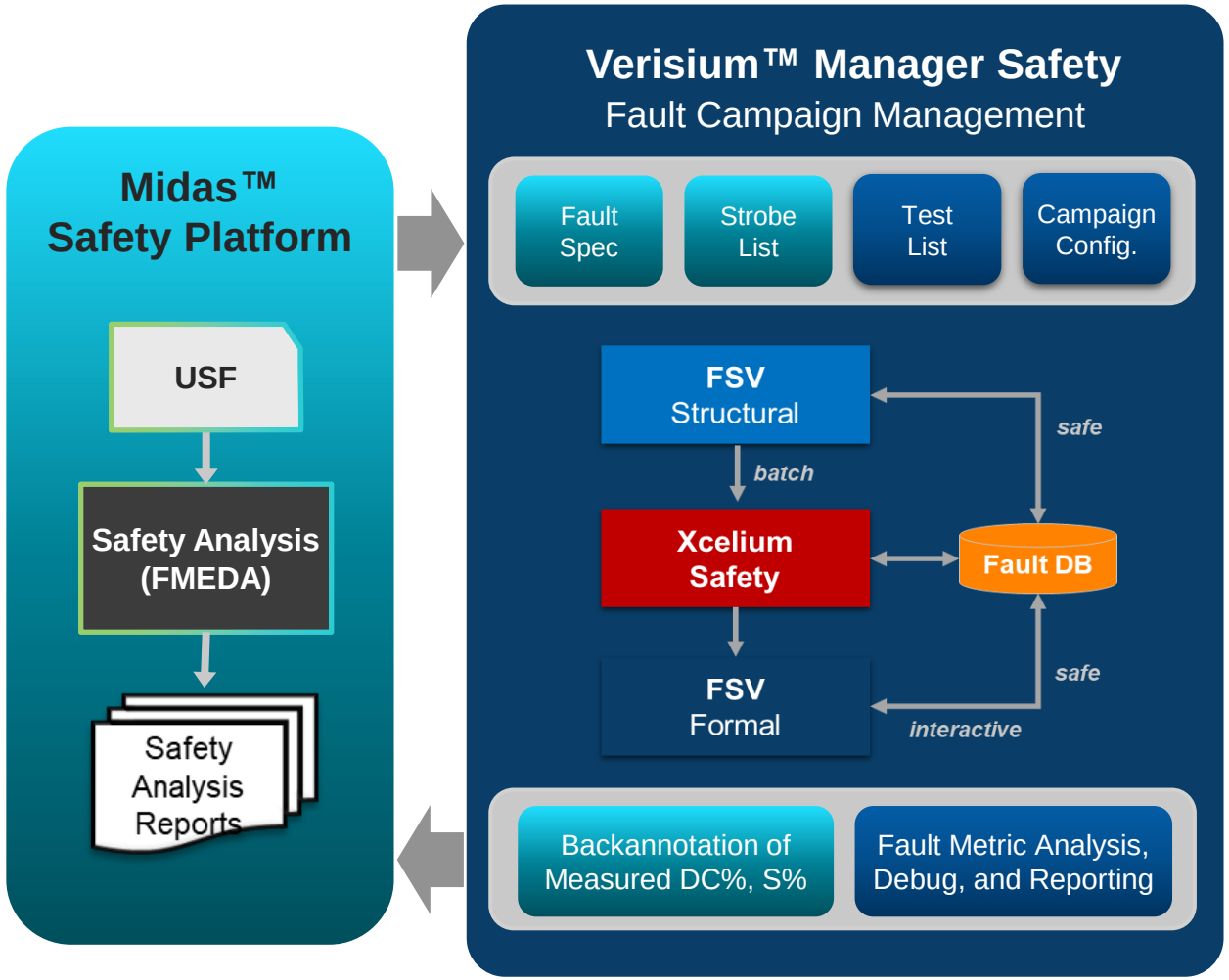
- It is possibile to export the Safety Goals metrics into a report

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

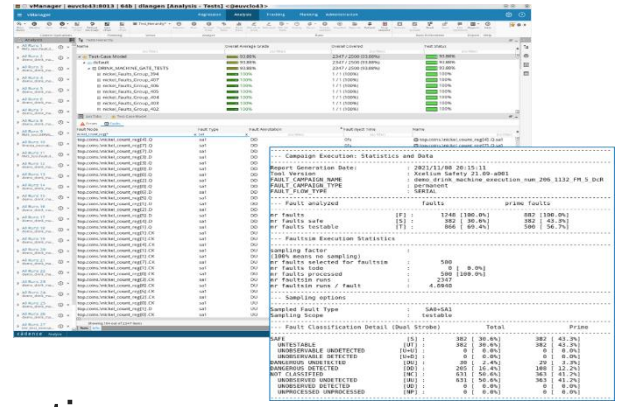
SG ID	FMEDA	Safety Goal Violations	SPFMp	SPFMt	LFM	PMHFp	PMHFp%	PMHFt	PMHFt%	Design Failure Rate Permanent (FIT)	Design FITp%	Design Failure Rate Transient (FIT)	Design FITt%
SG_1	FMEDA_DTFM	My safety goal 1	0.00%	0.00%	--	4.542e-04	58.3%	1.106e-02	57.9%	4.542e-04	58.3%	1.106e-02	57.9%
SG_2	FMEDA_DTFM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

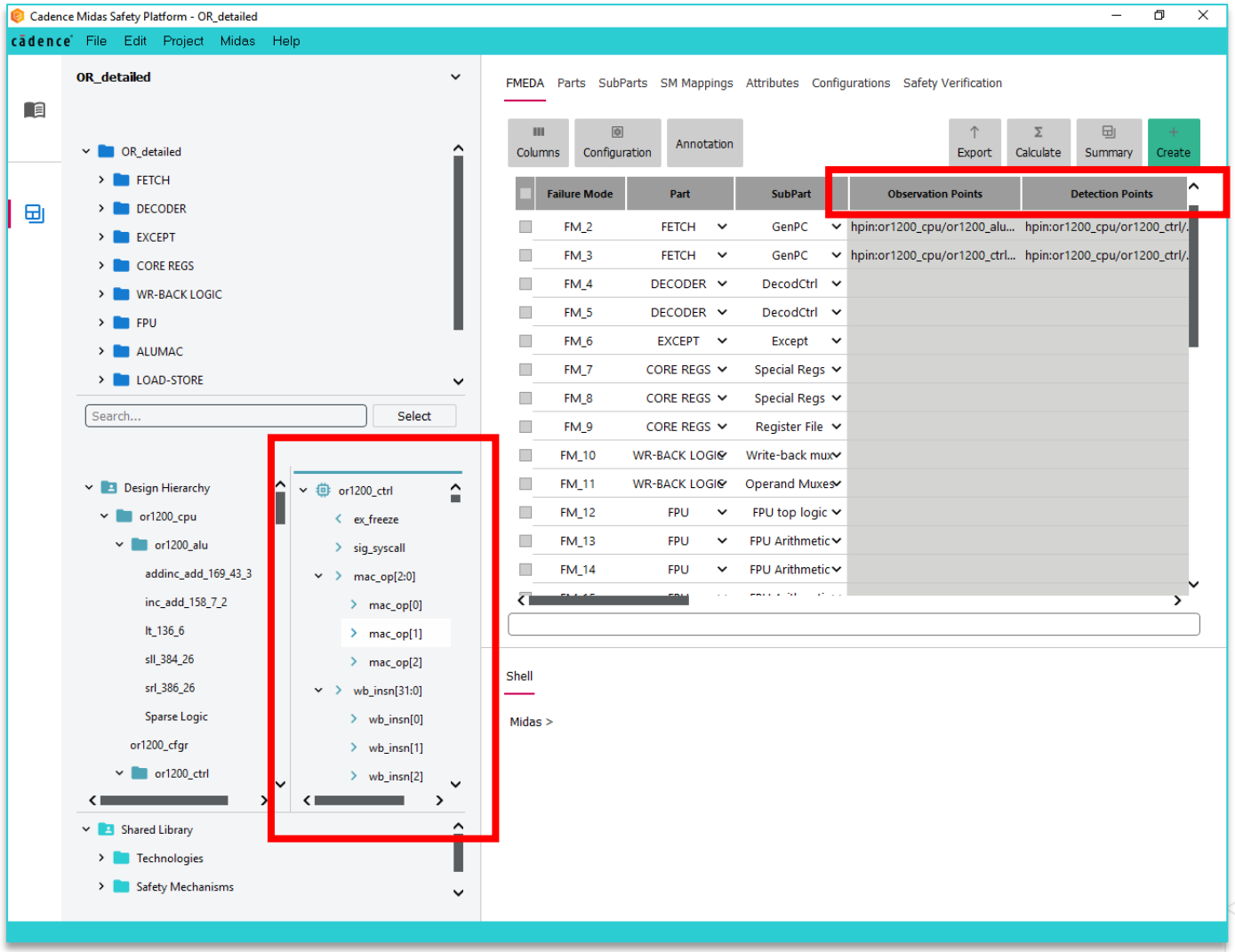


- Test selection and ranking
  - Coverage-based test selection
  - Customizable ranking criteria
- Fault list reduction
  - Fault sampling
  - Fault collapsing
  - Testability analysis
  - Test Dropping
- Fault campaigns execution
  - Measured Diagnostic Coverage and Safeness
  - Backannotation of results 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

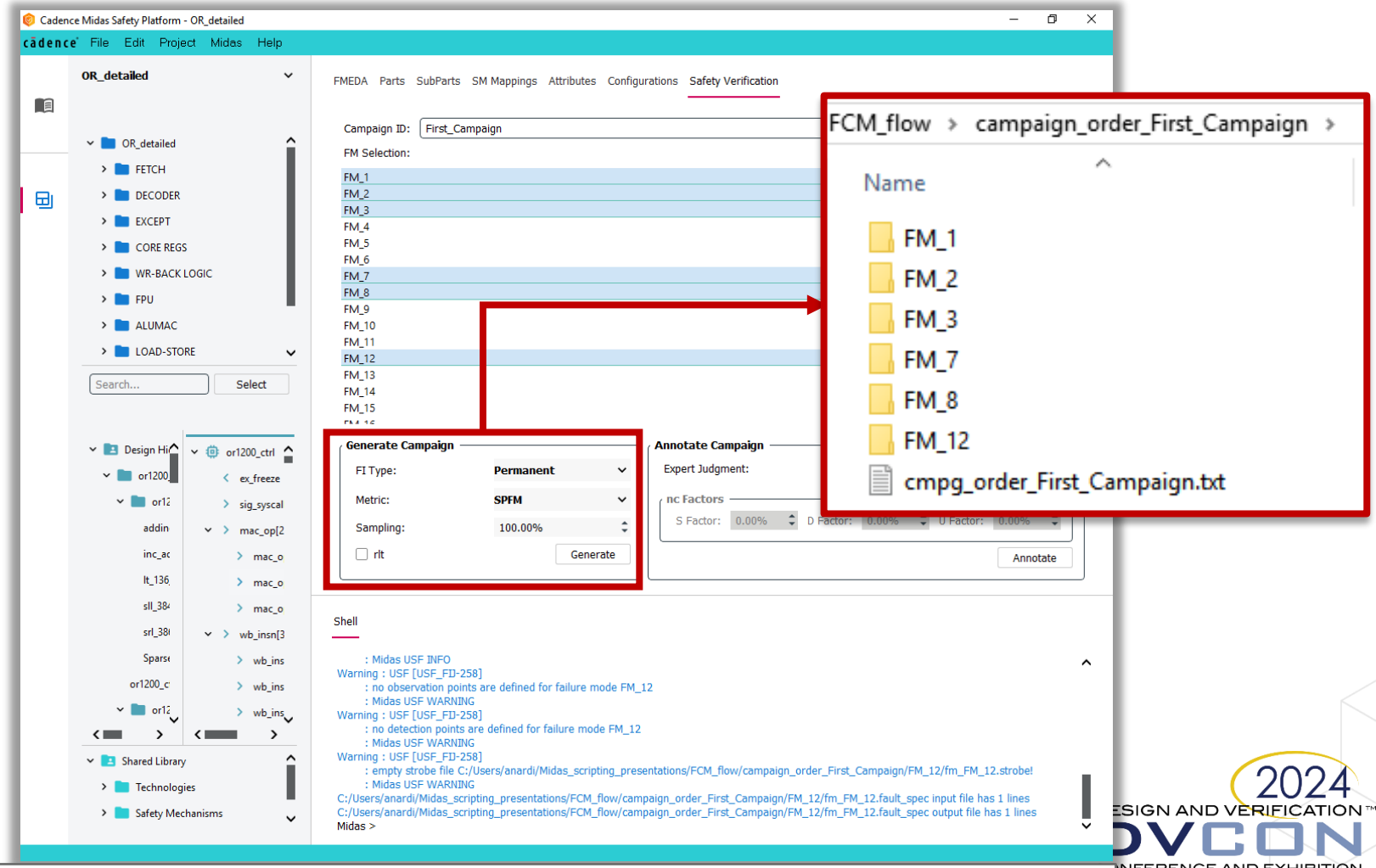



```
create_failure_mode ... [-observation_points obspoints] [-diagnostic_points detpoints]

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

Driving Fault Simulation Campaign for DC Validation

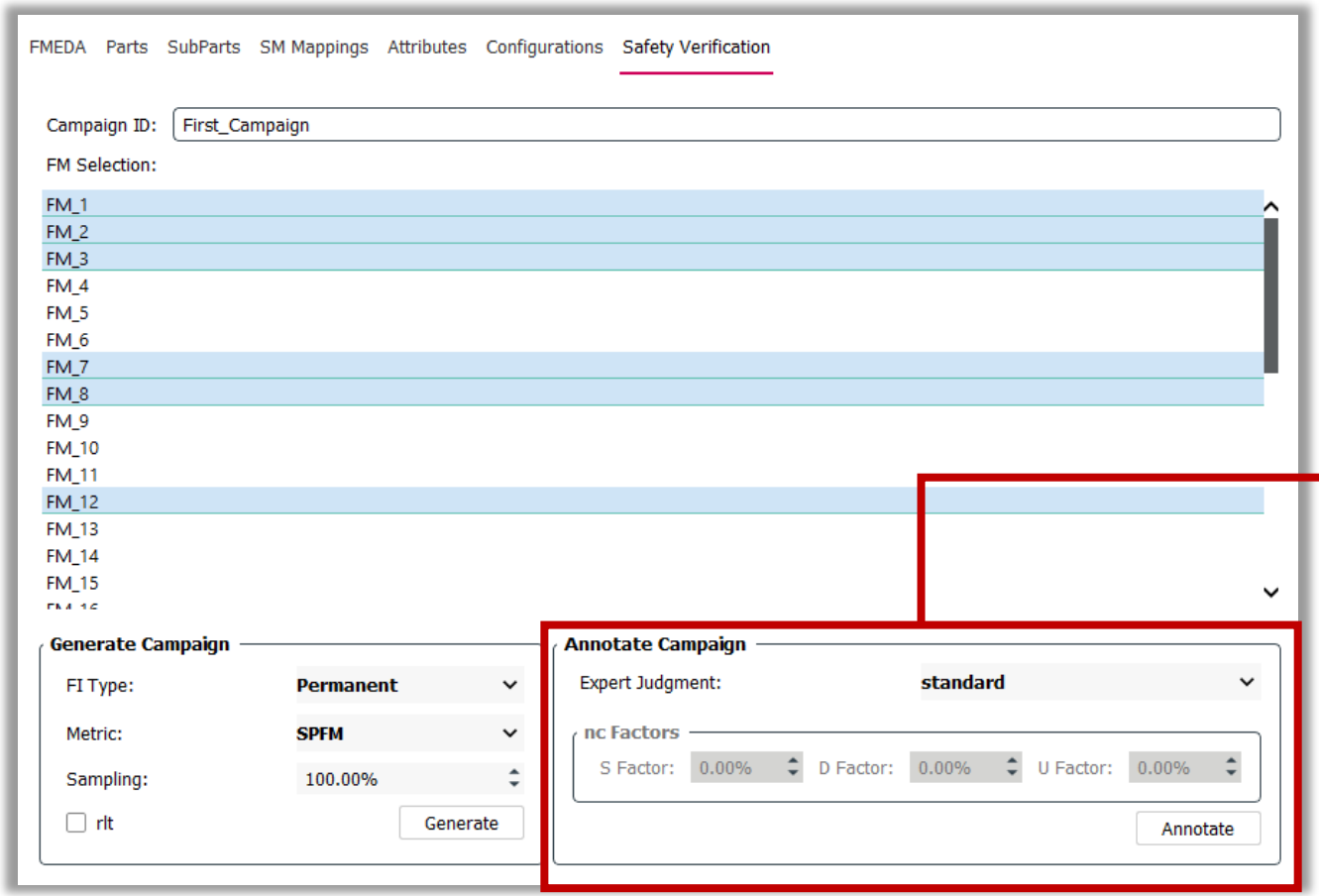
Fault Injection Campaign Order Generation

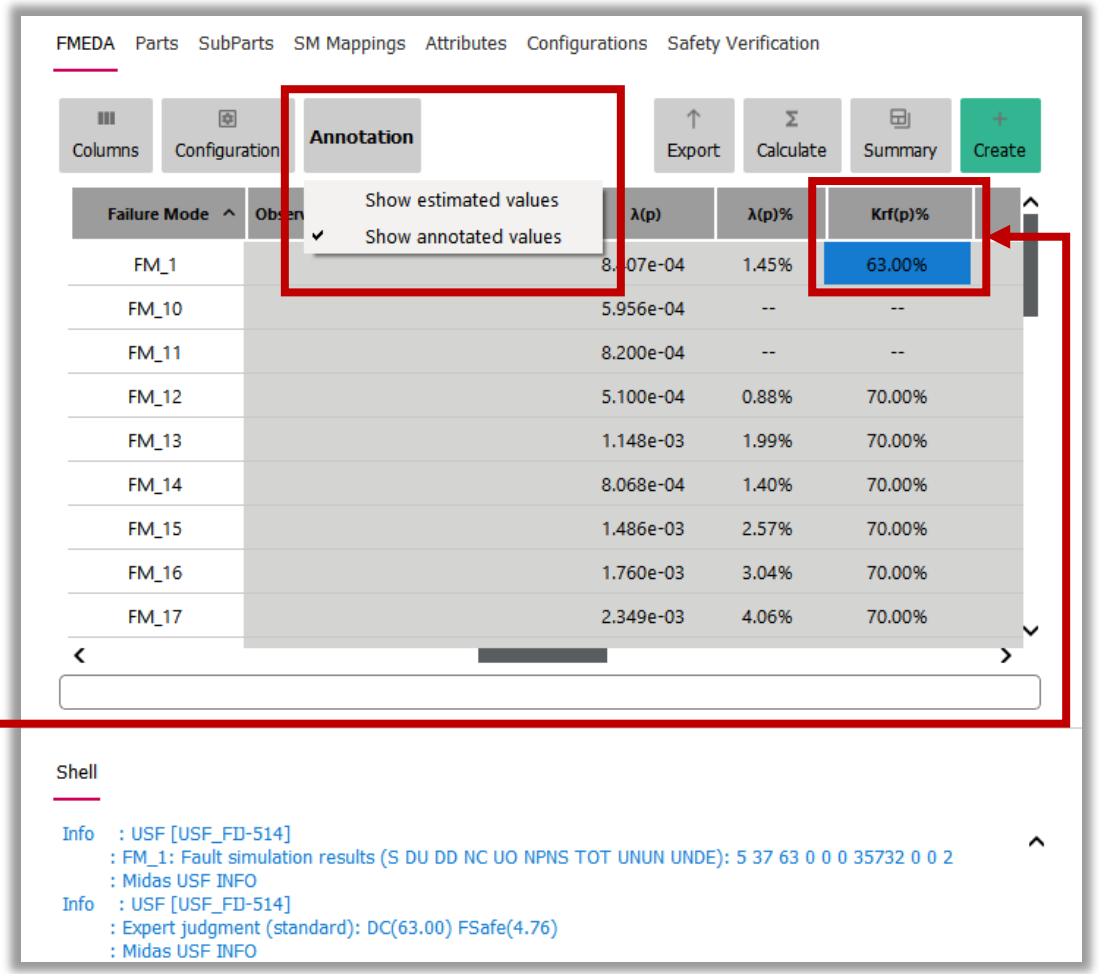
- Generation of the campaign order
  - Summary of the Fault Injection Campaign
  - Fault specification file
  - Strobe specification
  - Verisium Manager configuration




```
usf_campaign_order CAMPAIGN_FI1 -fmeda MYFMEDA -fm {I2C0_FM_1 I2C0_FM_2} \
                   -p -generate -sampling 100 -spfm
```

Back-annotation of the Fault Injection Campaign Results

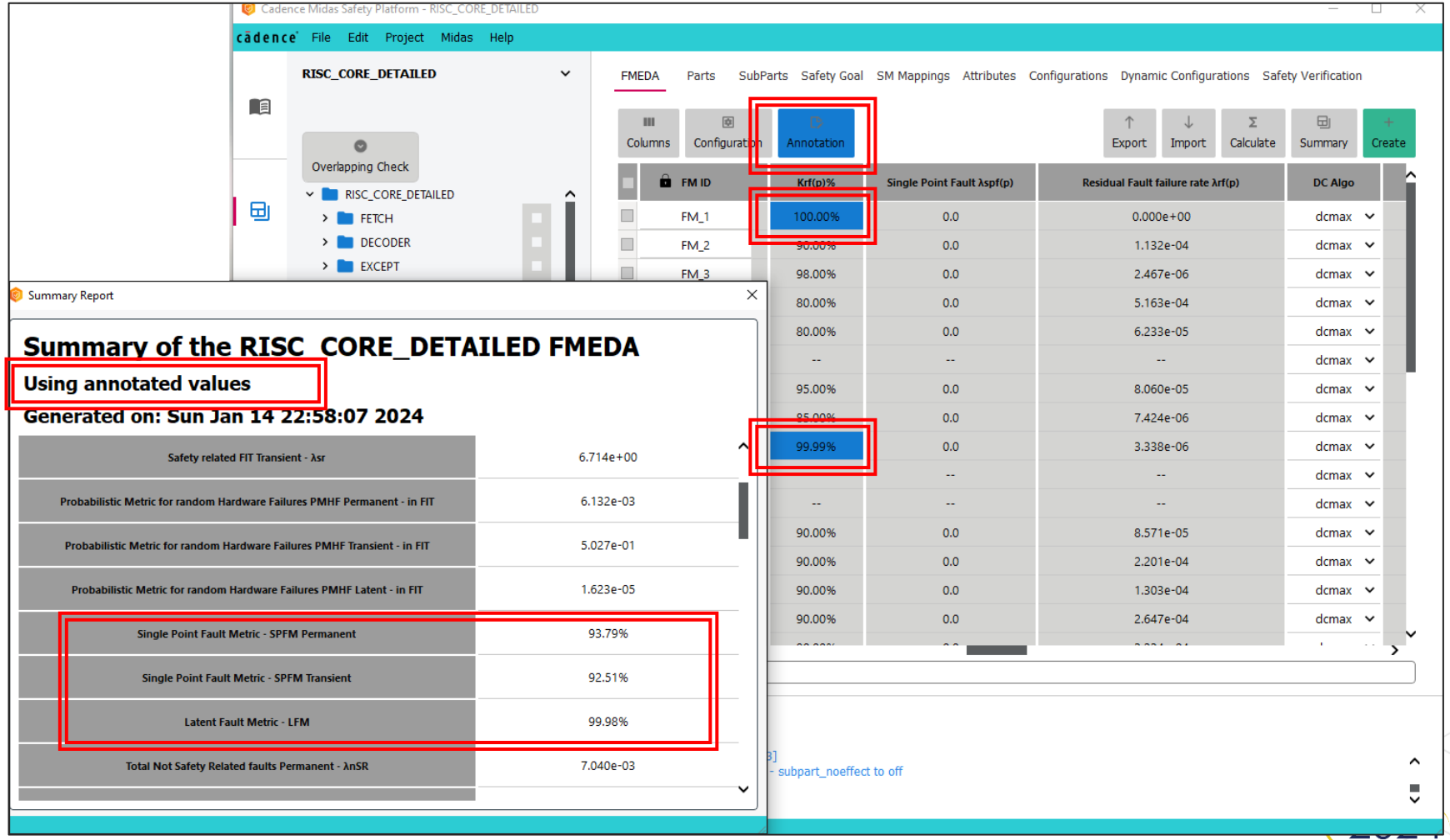





```
usf_campaign_order CAMPAIGN_FI1 -fmeda MYFMEDA -fm {I2C0_FM_1 I2C0_FM_2 I2C0_FM_3 I2C0_FM_4} -p -annotate \
                   -expert standard
```

Generate Final Reports

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




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

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

2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE
MUNICH, GERMANY
OCTOBER 15-16, 2024

USF-based FMEDA-driven Functional Safety Verification

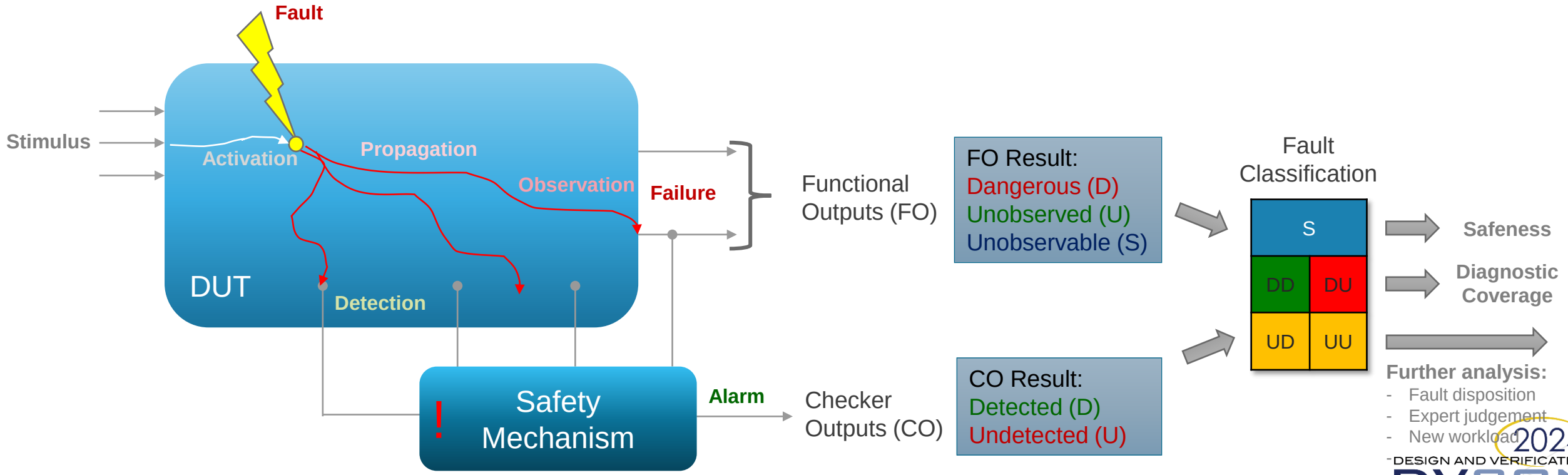
Fault Campaign Management (Verisium Manager Safety + Xcelium Safety + Jasper Safety)

Frederico Ferlini, Sr Principal Product Engineer, SVG

cadence®

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

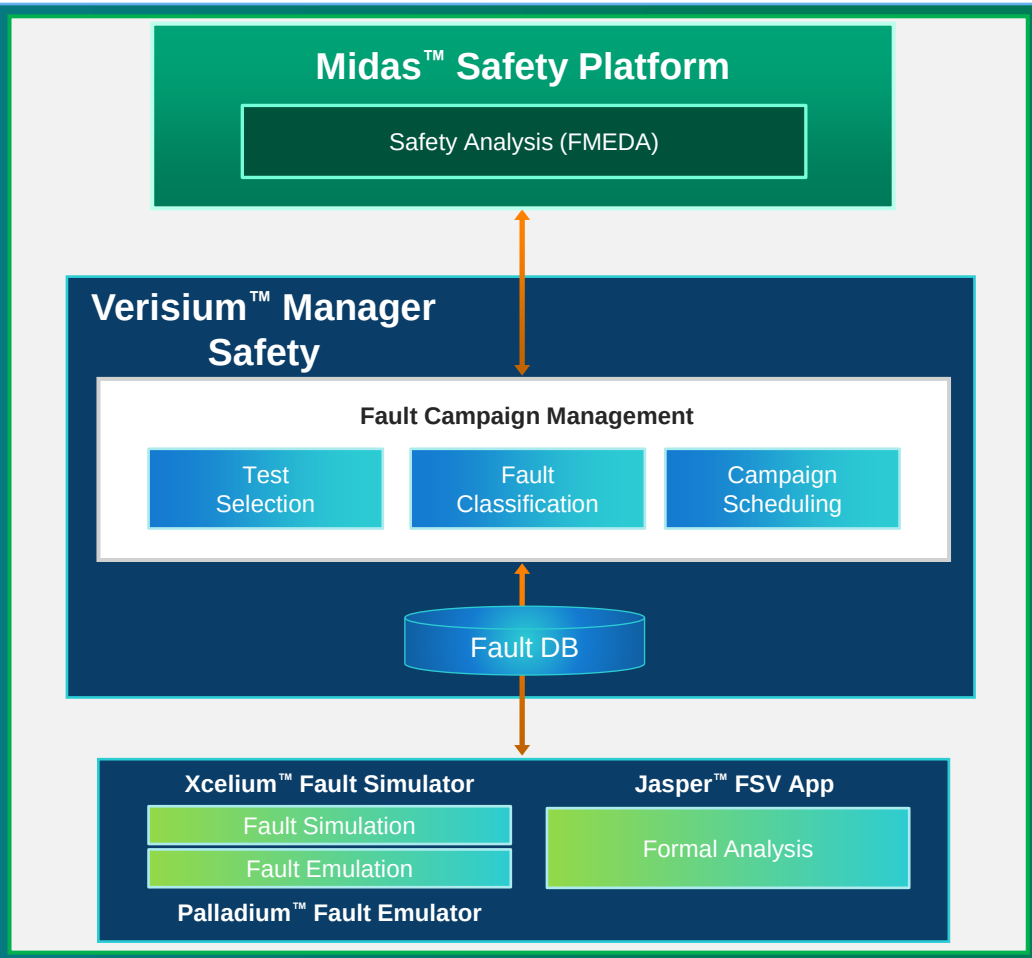
- Applies industry-leading formal techniques to fault analysis
- Increases safety verification performance

Injection Engine – Xcelium™ Fault Simulator

- Native serial and concurrent fault simulation engine

Acceleration – Palladium™ Fault Emulator

- Seamless define faults and strobe as for simulation



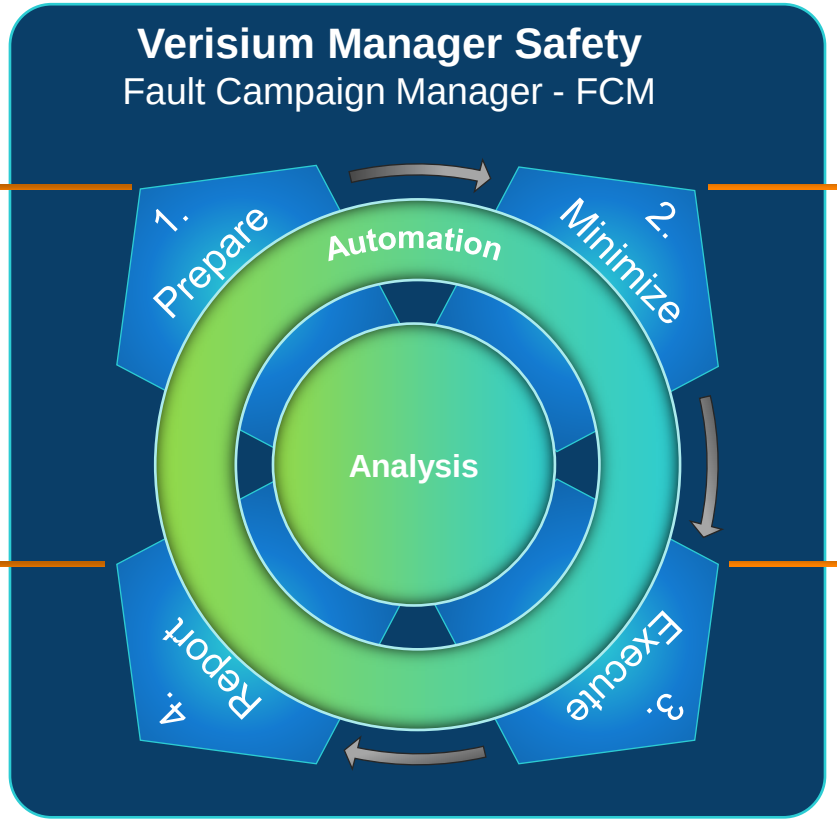
Verisium™ Manager Safety

Fault Campaign Manager – FCM

Fault Campaign Automation and Analysis

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



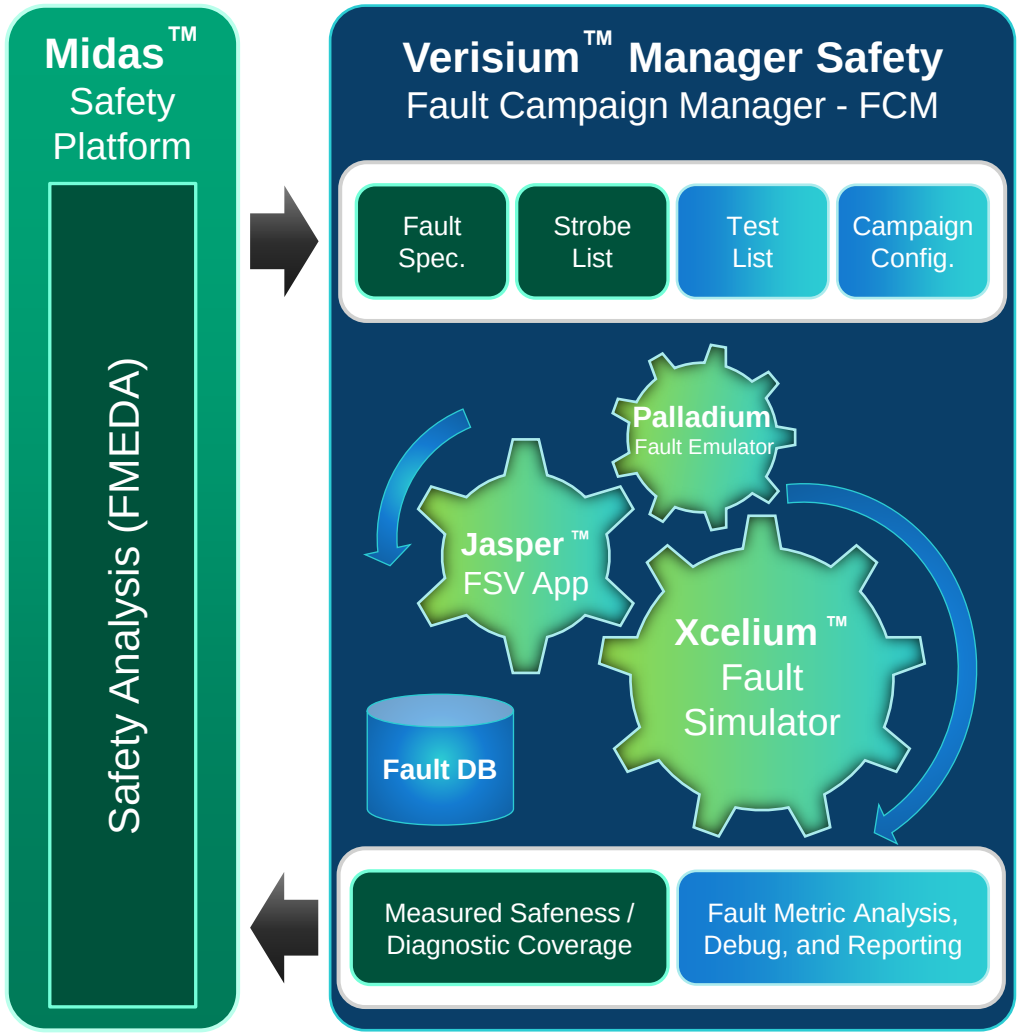
4. Generate Reports
- Campaign summary report
- Diagnostic Coverage / Safeness
- FMEDA validated results

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

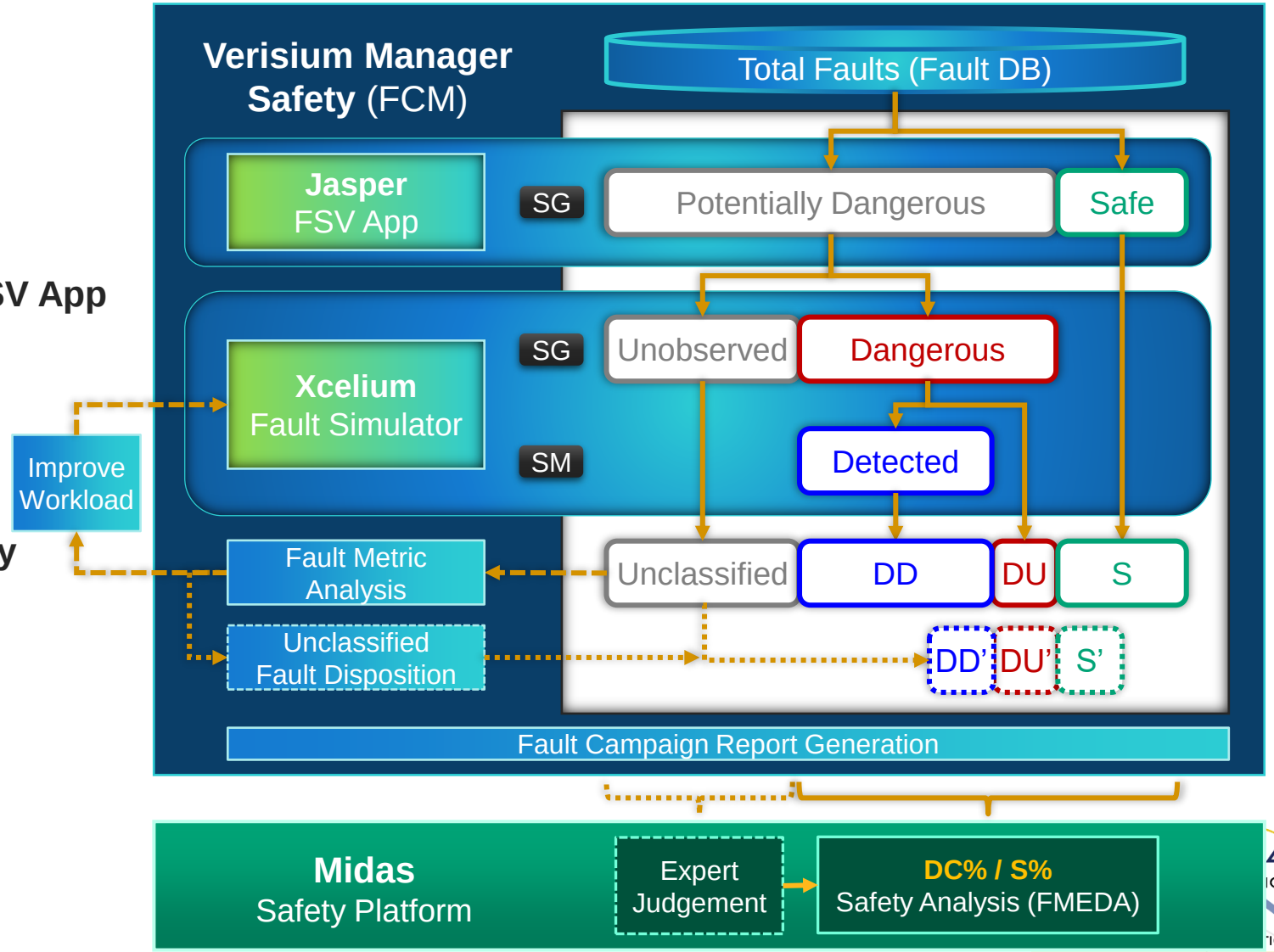
FMEDA-driven Fault Campaign



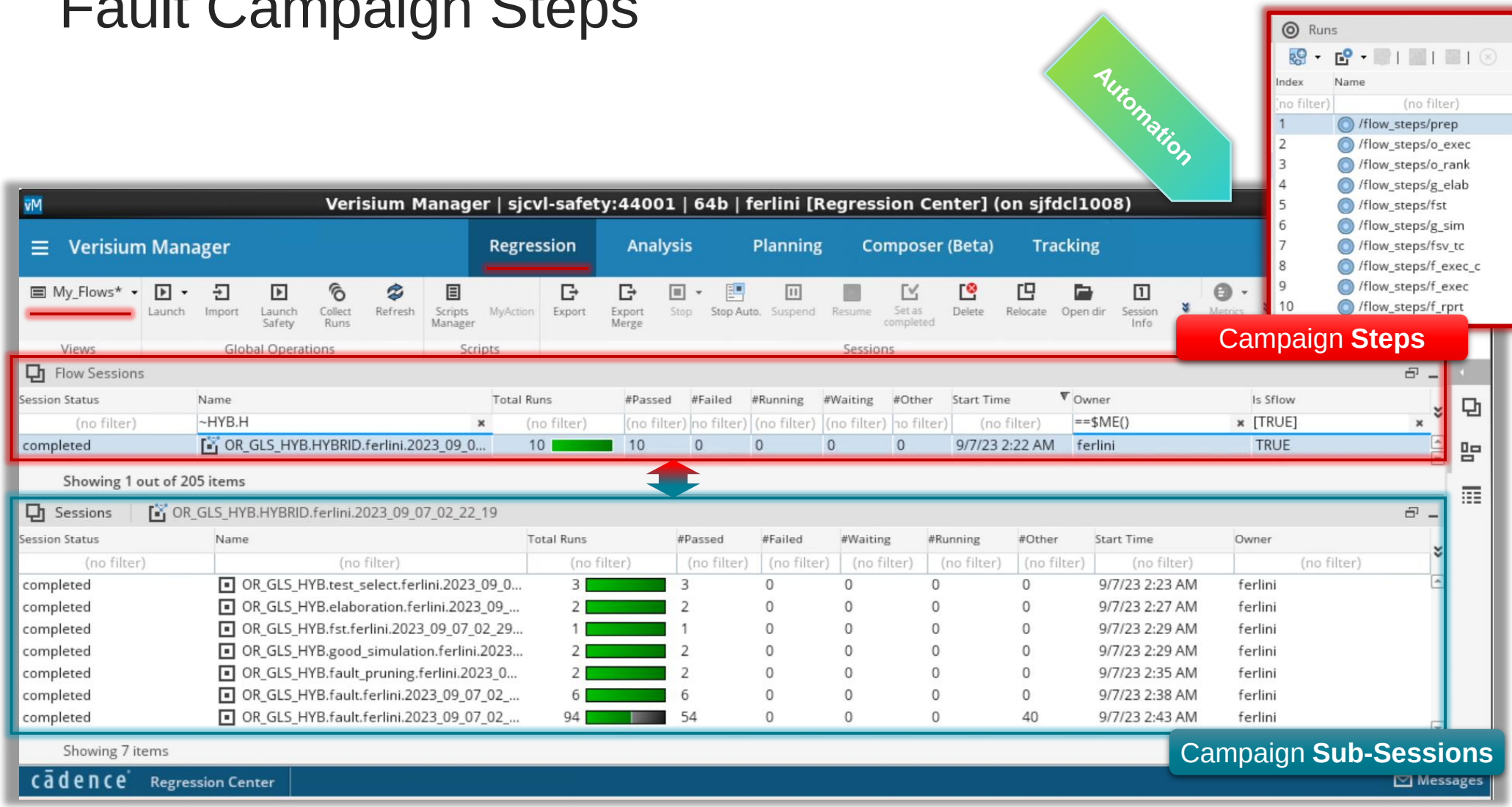
- **Inputs**
  - Safety Engineer
    - Fault Targets (derived from FMEDA ⇔ design mapping)
    - Strobe List (observation and detection points)
  - Verification Engineer
    - Test List (selected for fault analysis)
    - Campaign Configuration
      - Optimizations, runs distribution, customization, etc.
- **Outputs**
  - Summary Report
    - Measured Fault/Diagnostic Coverage, Safeness
  - Fault Annotation
    - Fault Metric Analysis, annotated fault list, …

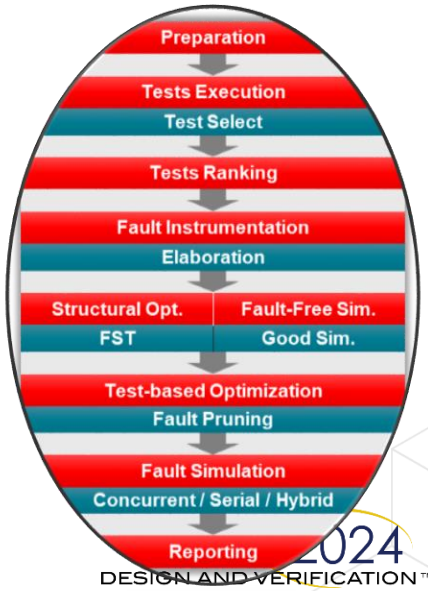
Fault Classification

- **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



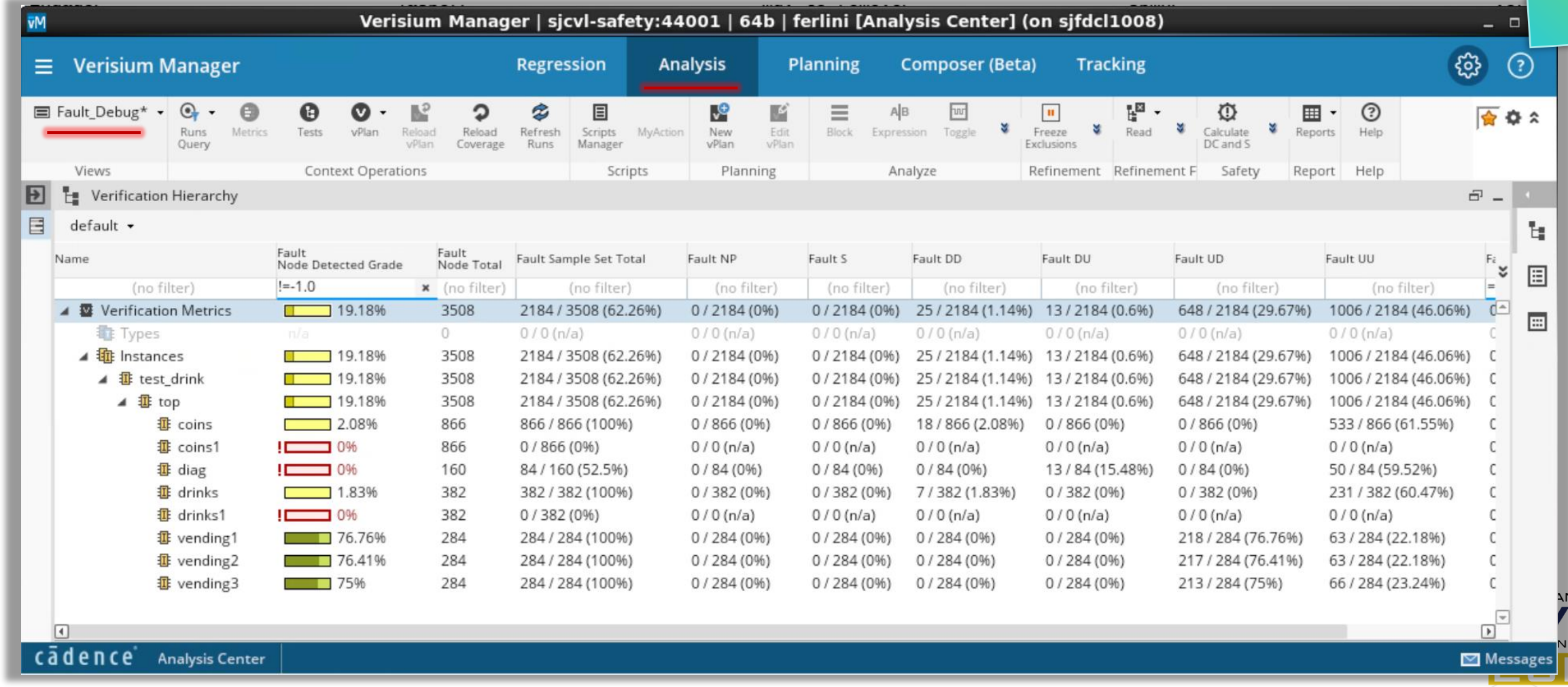


2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE MUNICH, GERMANY OCTOBER 15-16, 2024 cadence

Dedicated Fault Analysis

Hierarchical Data

Analysis



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%)

Campaign Summary Report

Date, tool version, fault types, sampling, …

Static instrumentation fault results

Overall campaign(s) merged results

Fault Disposition (user refinement)

Applicable client configuration

```
---- 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
---- 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
--------- 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
```

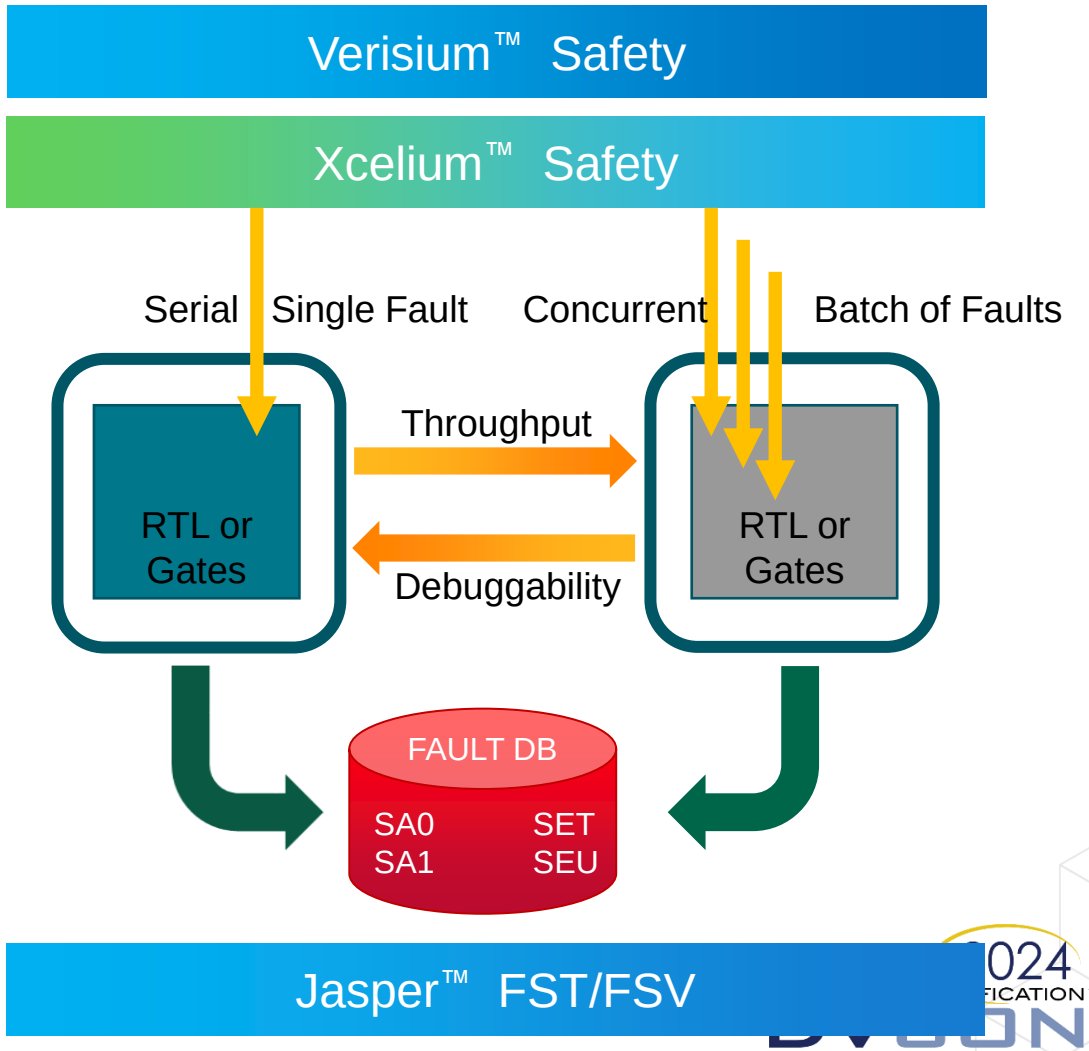
Sampled fault scope

Fault and Test Coverage results and formulas

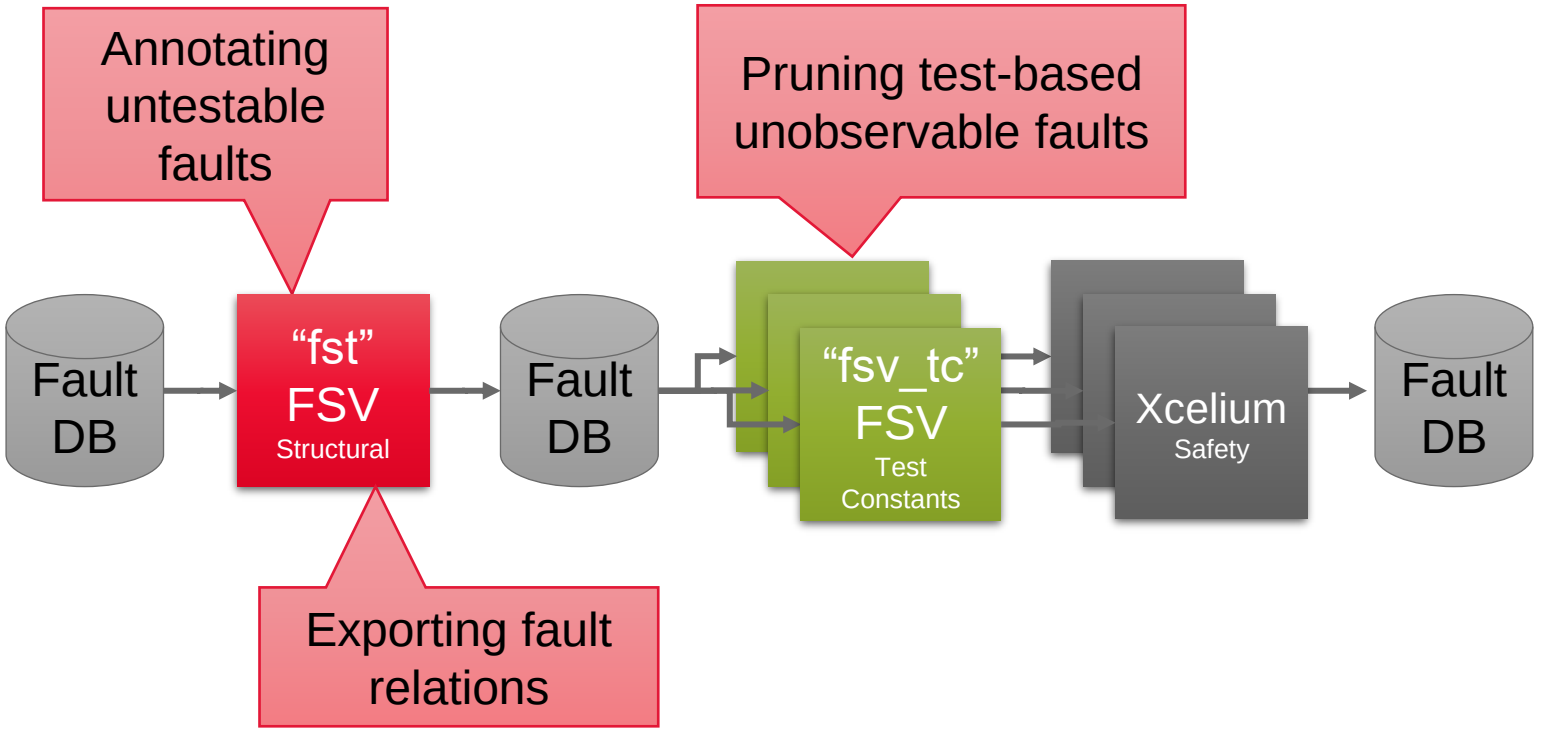
Fault Campaign Management – Safety Engines

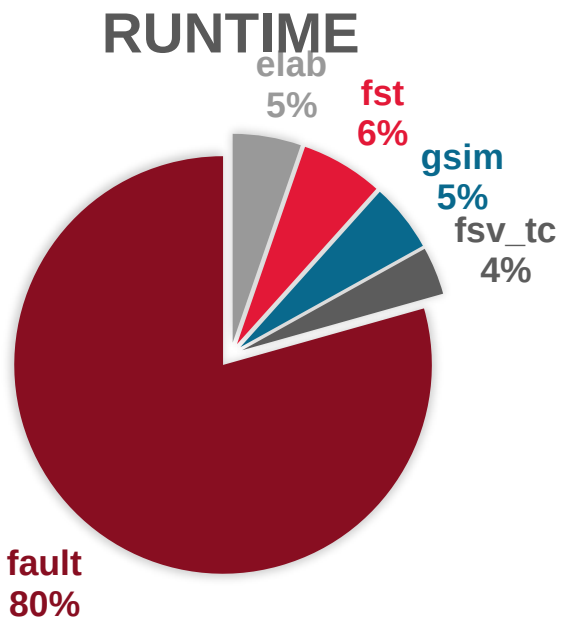
Xcelium Safety App

- 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)



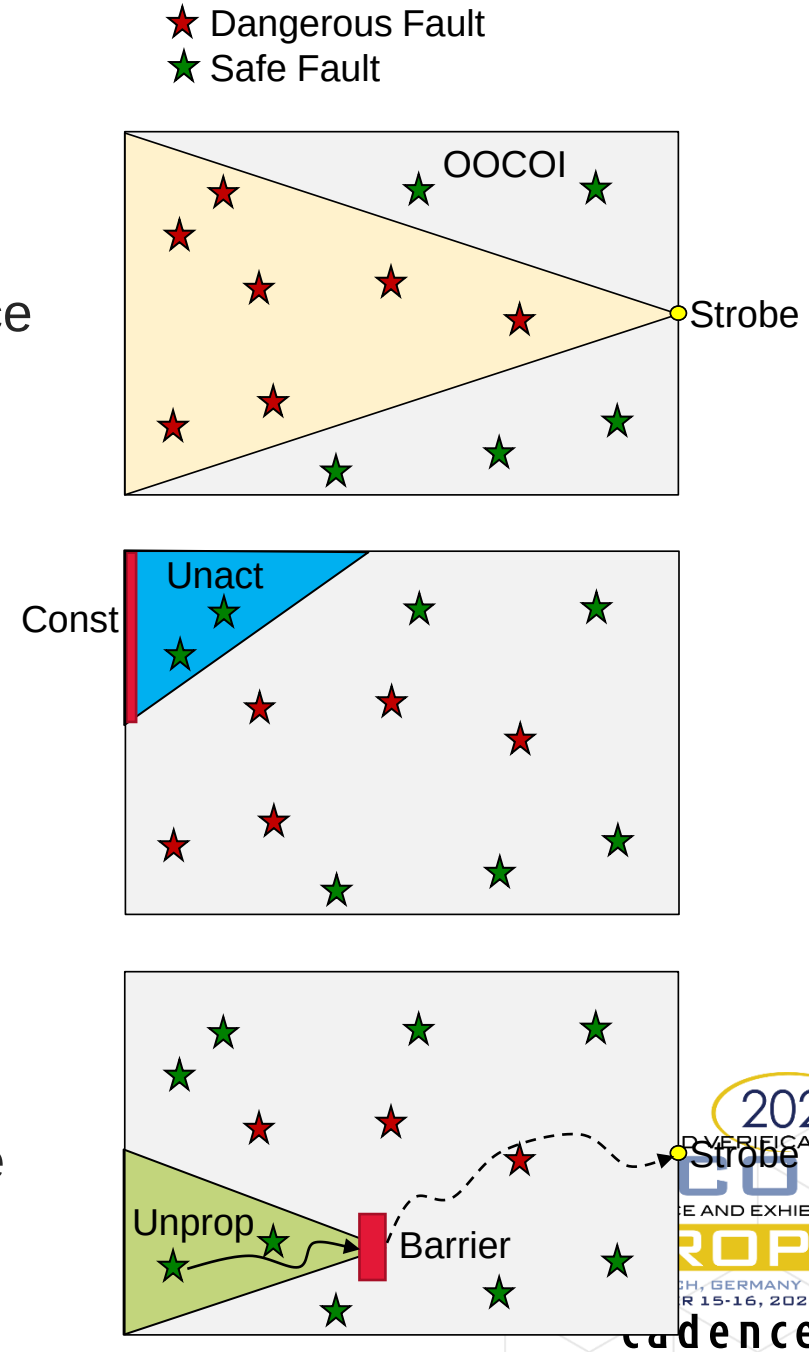


A few minutes of optimization can save hours of simulation

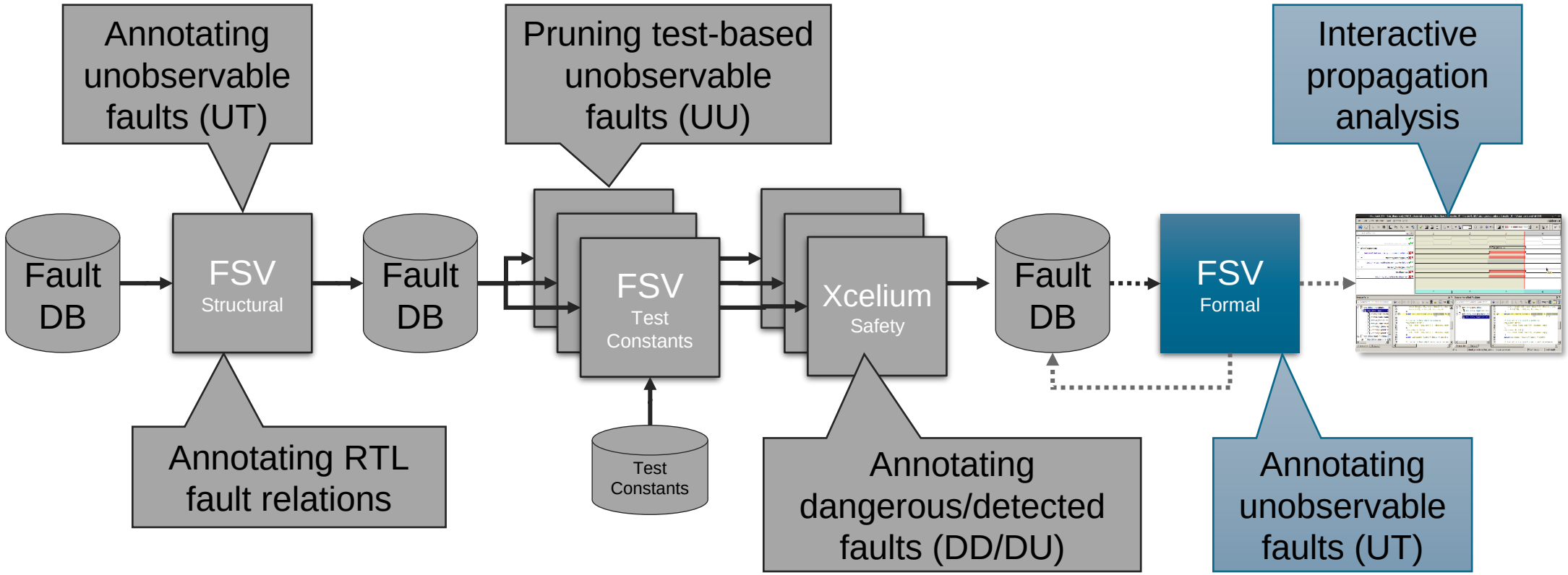
- FSV exports fault relations → equivalent faults will be skipped
- FSV annotates untestable faults → Safe faults will be ignored
- FSV annotates faults as unobservable by test → Pruned faults will be dropped

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



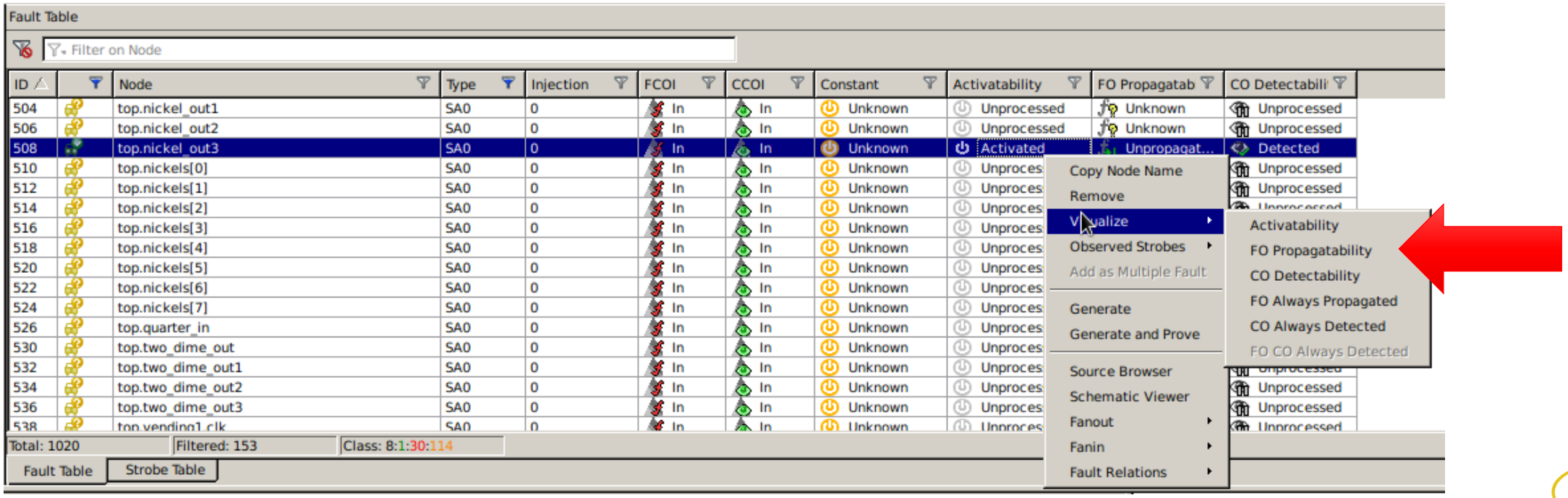
FSV Integration with Xcelium Safety Simulator



- 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

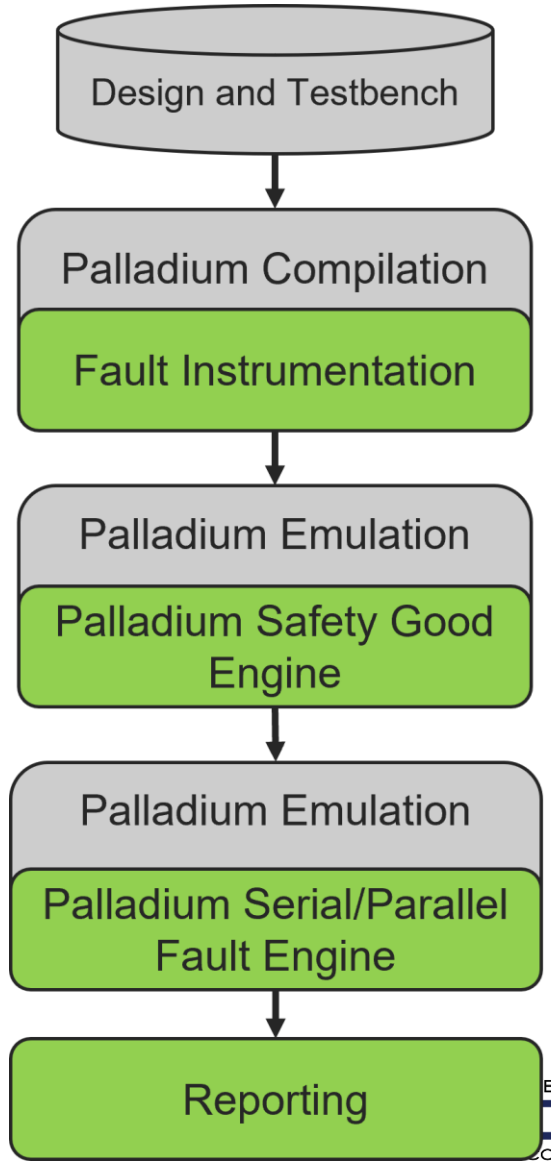
FSV Formal – Debugging Visualize Waveforms

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



Palladium Safety
User Flow

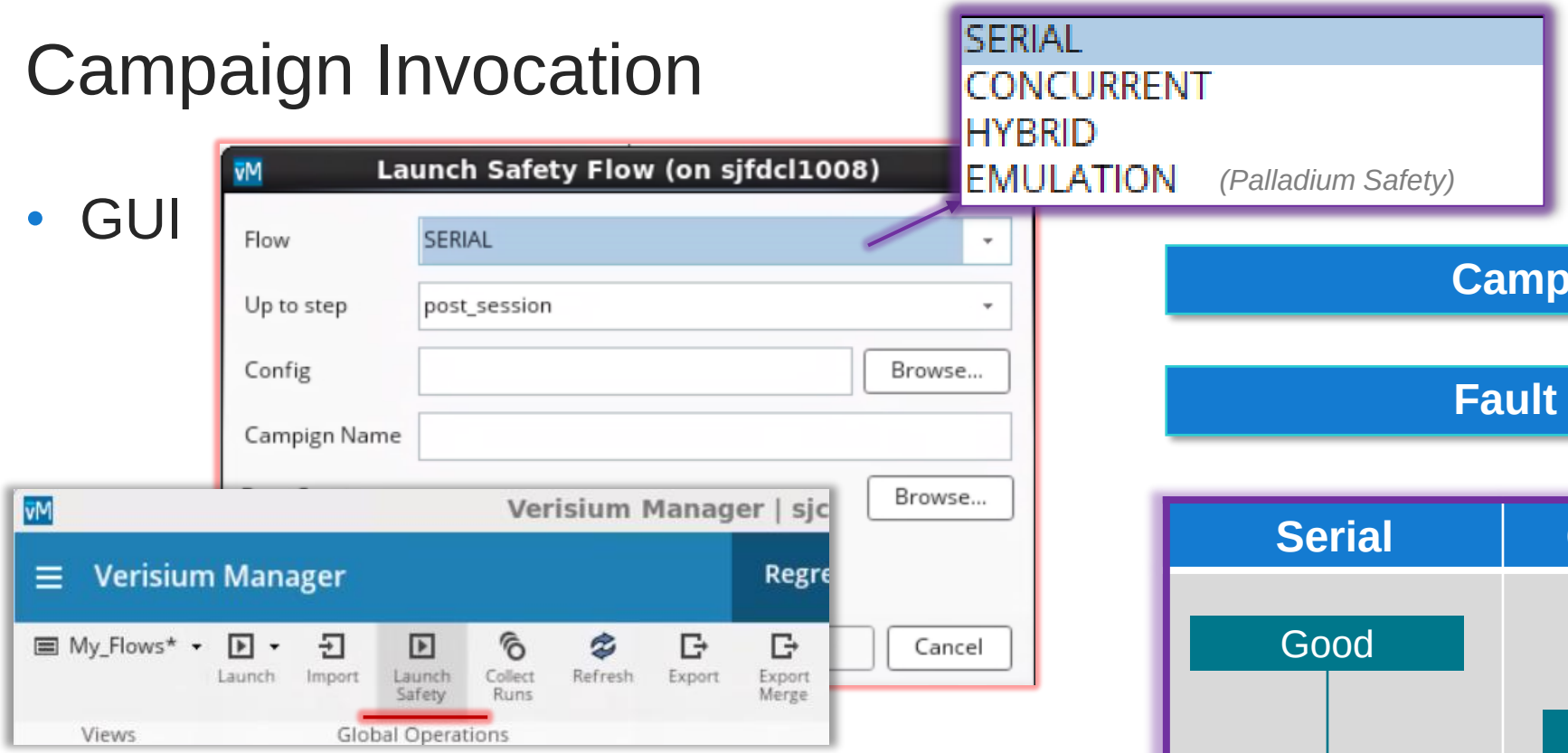
- Easy to migrate from Functional verification flows to Fault Injection
  - Some files and option to be added to Palladium compilation
  - Faults are identified and instrumented during compilation
- Fault-free circuit emulation (Good Emulation)
  - Process strobe points and capture good waveform
- Fault 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)

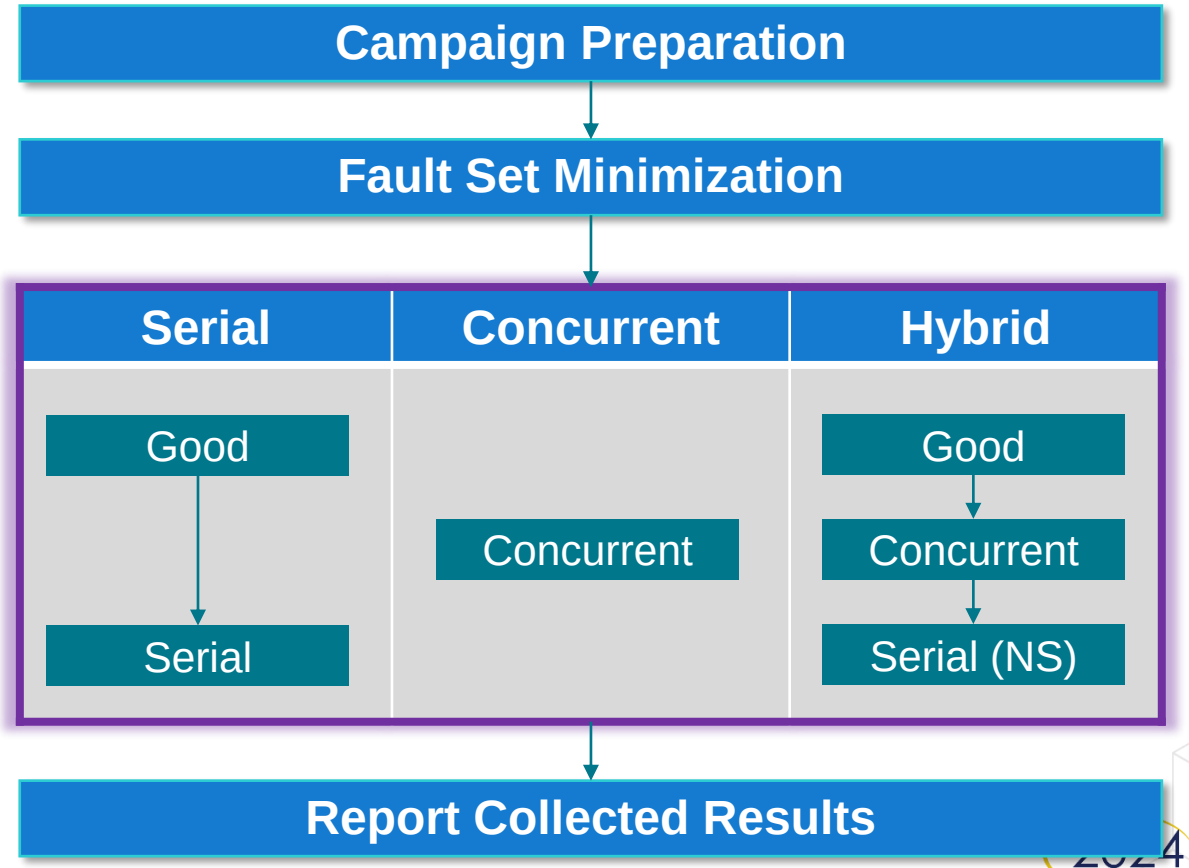


Fault Campaign Automation

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., strobes)

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

- Xcelium syntax

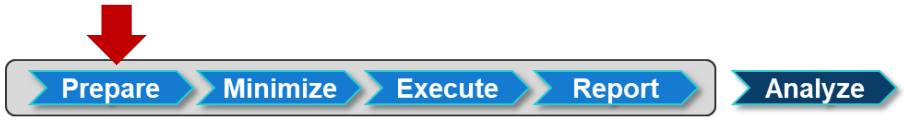
```
fs_strobe -functional top.dut.o
fs_strobe -checker top.sm.alarm
```

- 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



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    : .../strobes.list

FS_SAMPLING_PERCENT         :
FS_SAMPLING_...             :

FS_REGR_TESTS_VSIF          : .../tests.vsif
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

Prepare → Minimize → Execute → Report → Analyze

Legend:
- **Mandatory parameters**
- Midas overridden

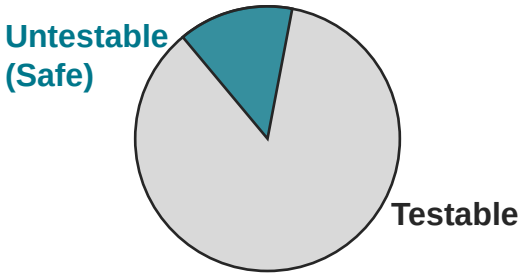
Fault Set Minimization

Design Structure

Testability Analysis

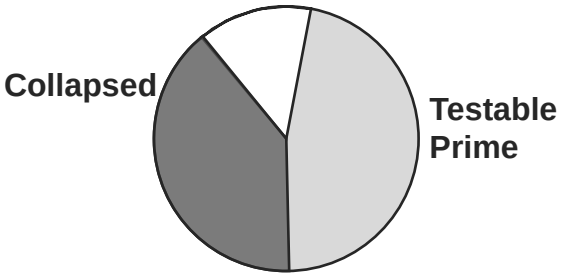
Identify faults:

- Uncontrollable
- Unobservable



Fault Collapsing

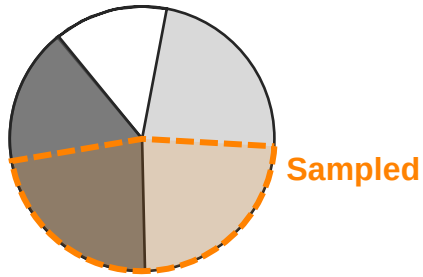
Group equivalent faults and consider only their prime representative



Statistics

Random Sampling

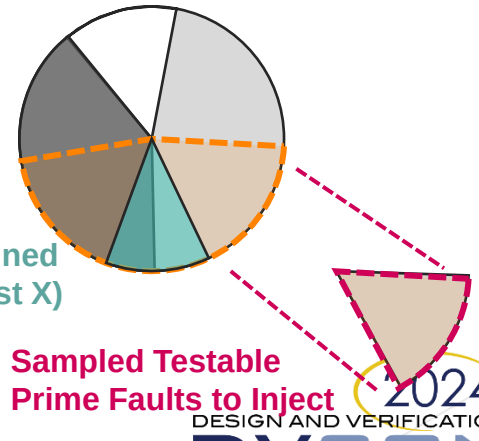
Estimate the overall results based on a representative sample



Test Stimulus

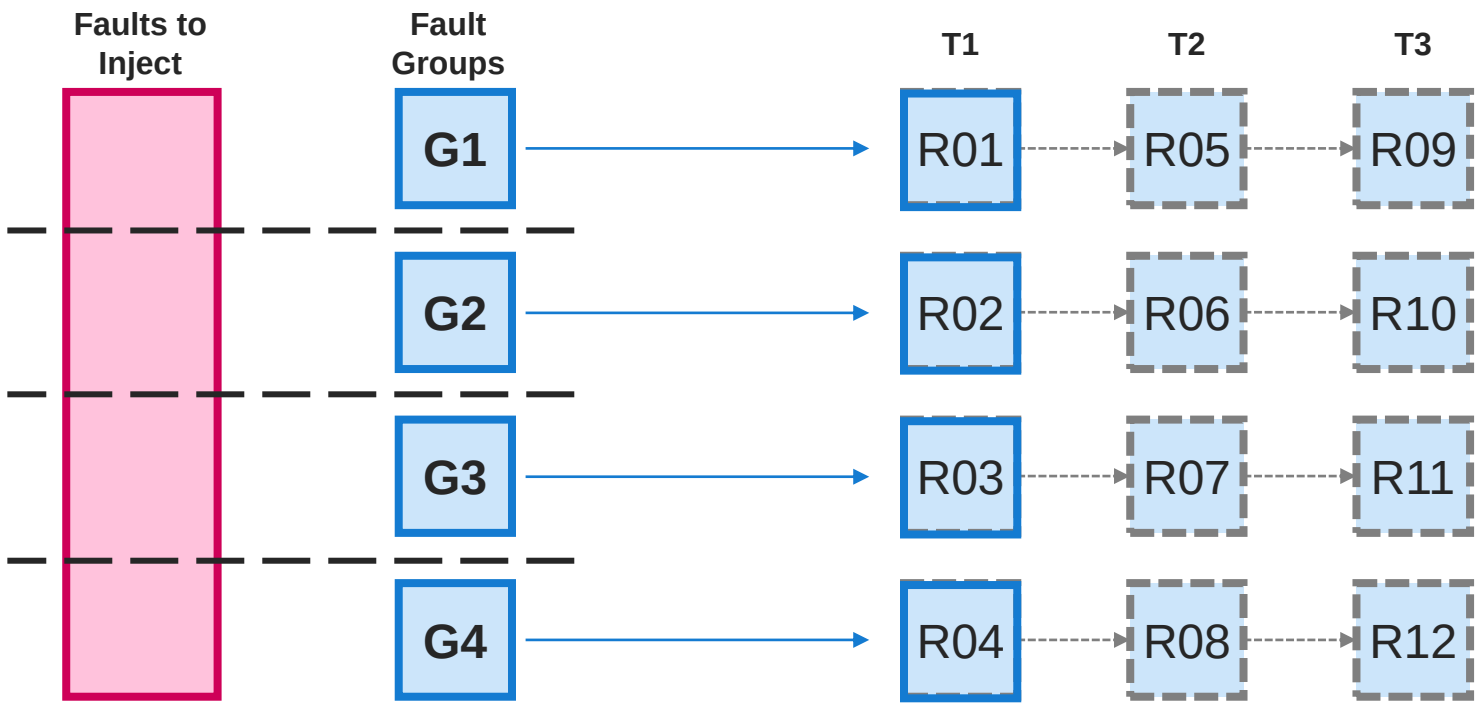
Fault Pruning

Find extra untestable faults by constraining testability based on stimulus patterns



Prepare → Minimize → Execute → Report → Analyze

Fault Injection Execution



RUN XX

1. Filter Pruned Faults
2. Invoke Serial/Concurrent
   1. Inject fault/s
   2. Drop detected
   3. Stop simulation
3. Scan simulator logs
4. Remove dropped faults
   1. Skip next test if no left

Fault Grouping

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

Fault Session Build

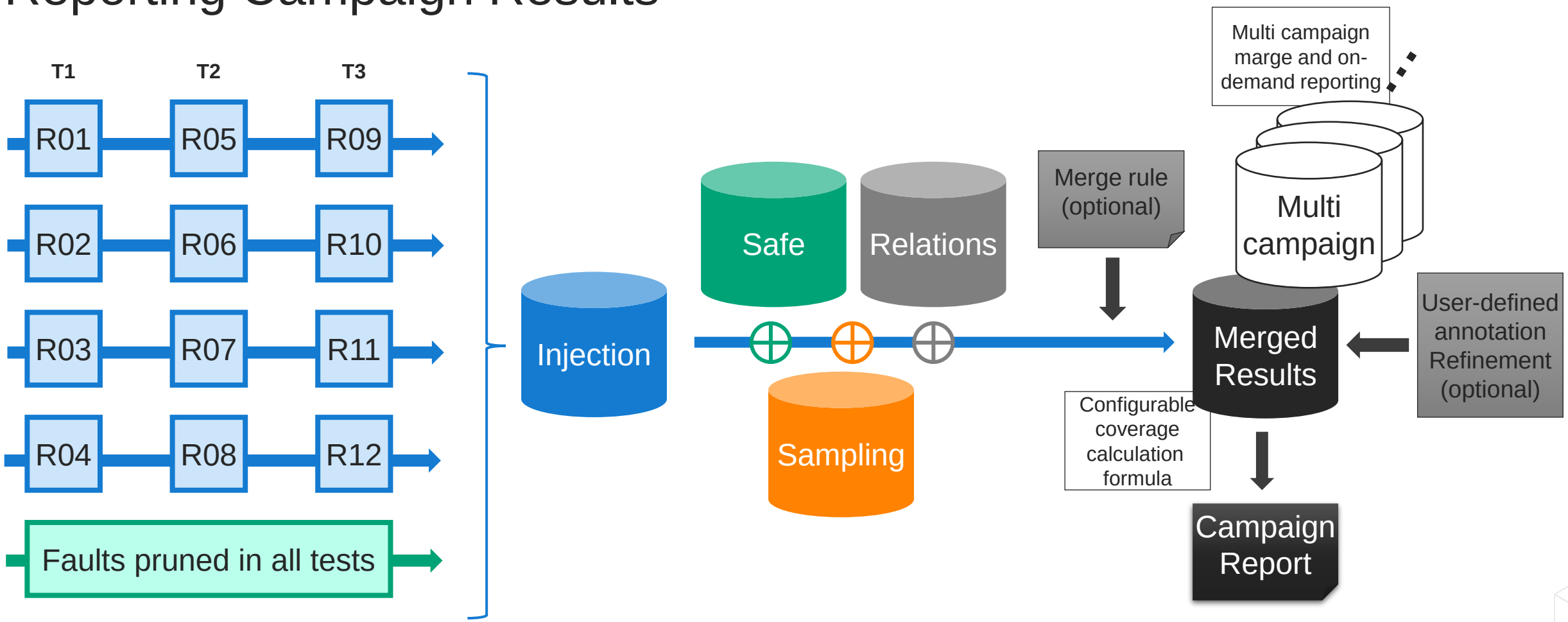
- Test dependency
- Submit runs to the computer farm

Fault Run Execution

- Filter faults
- Check for errors
- Optimize runs

Prepare → Minimize → Execute → Report → Analyze

Reporting Campaign Results

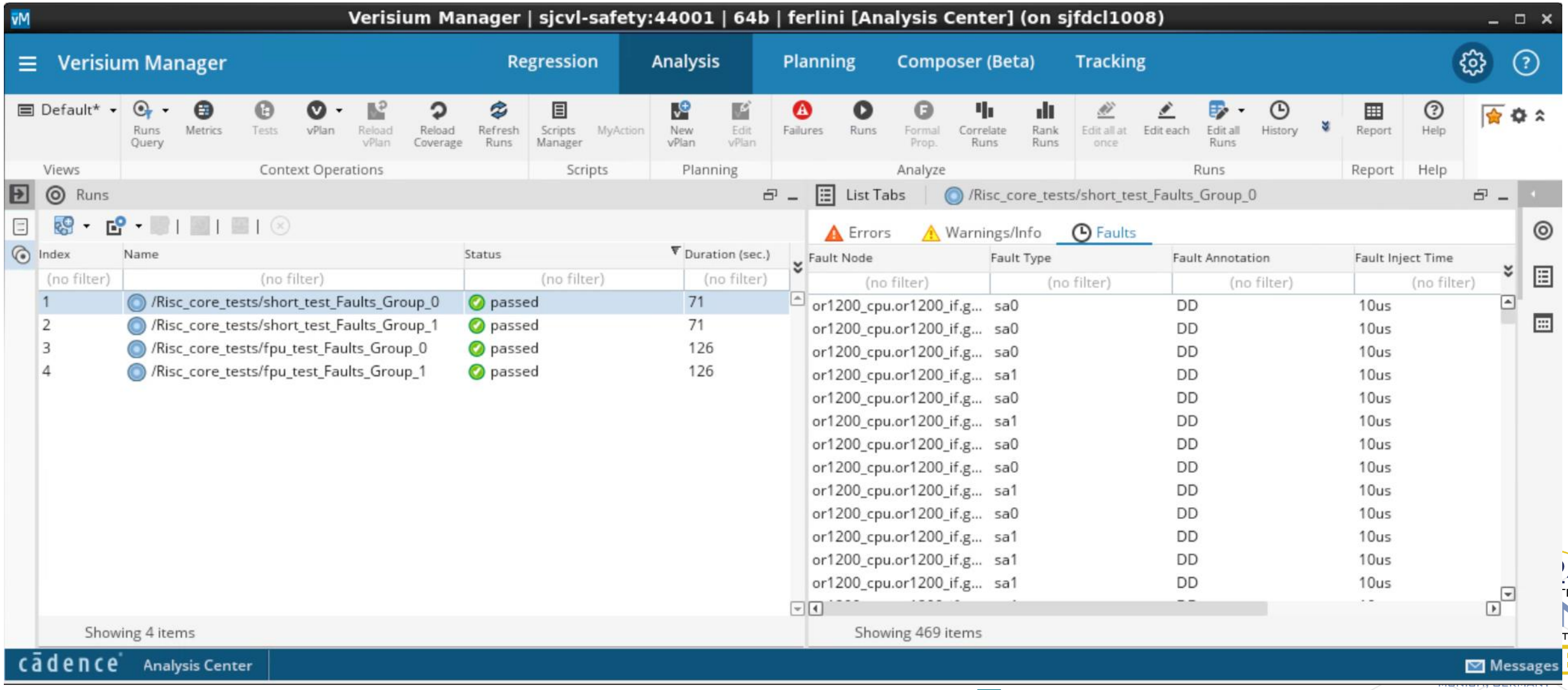


Prepare | Minimize | Execute | Report | Analyze

Fault Campaign Analysis

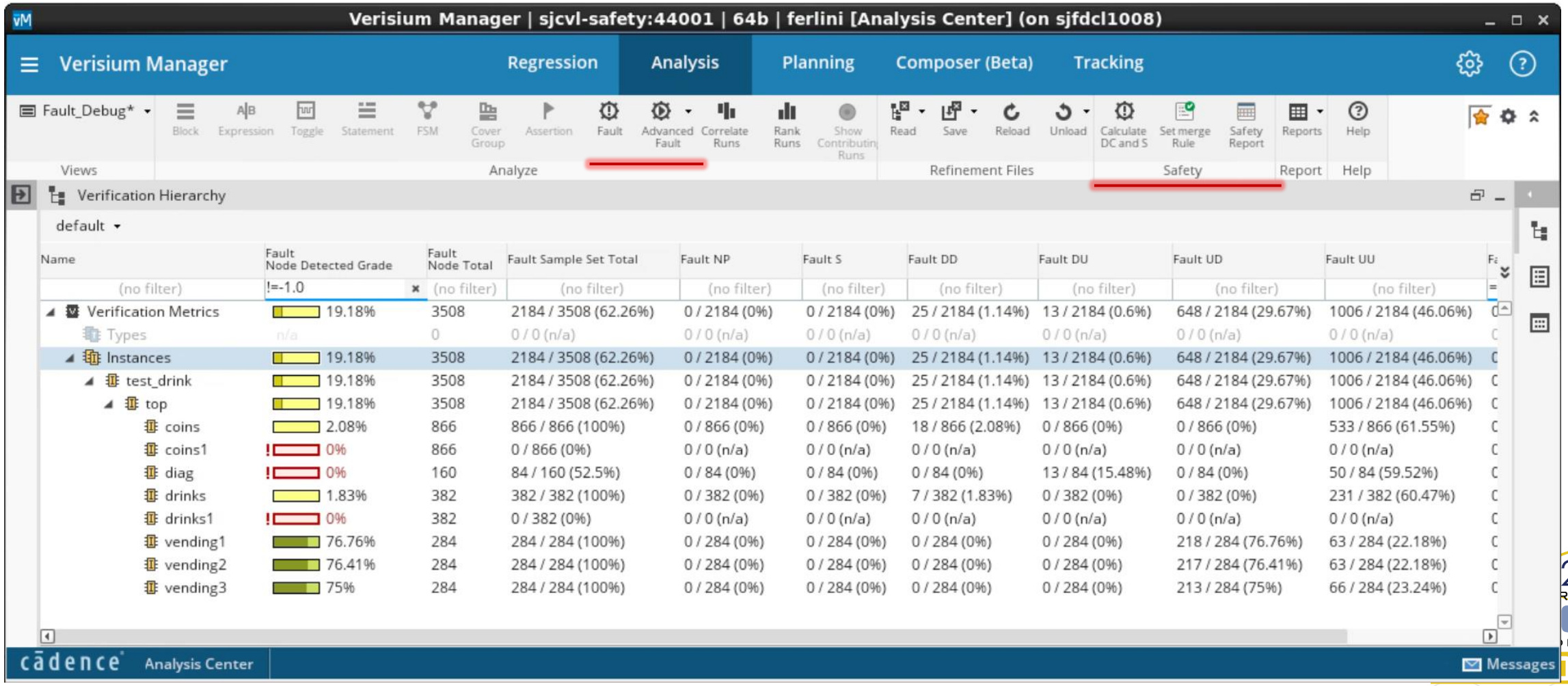
Fault Simulation Results

Run generated fault annotation



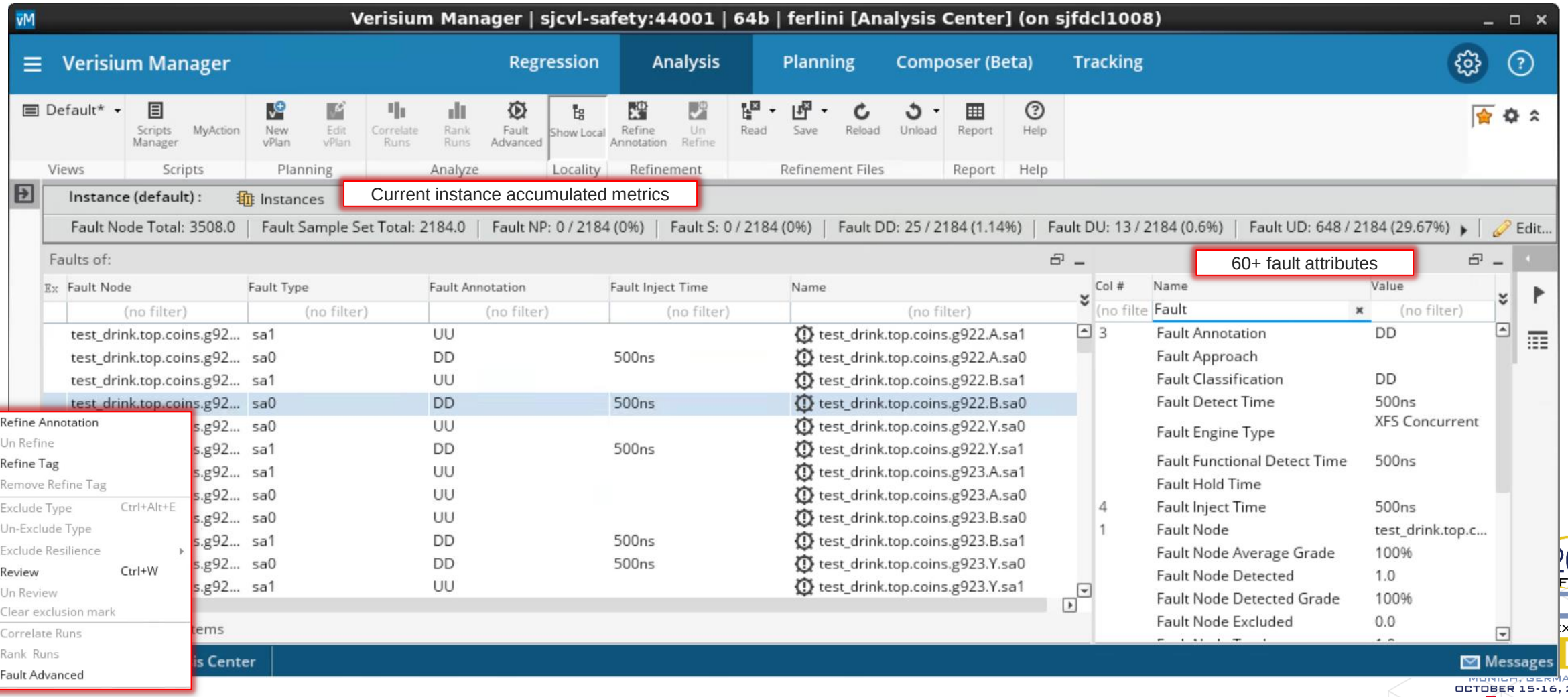
Fault Campaign Results – Hierarchical View

Merged annotation



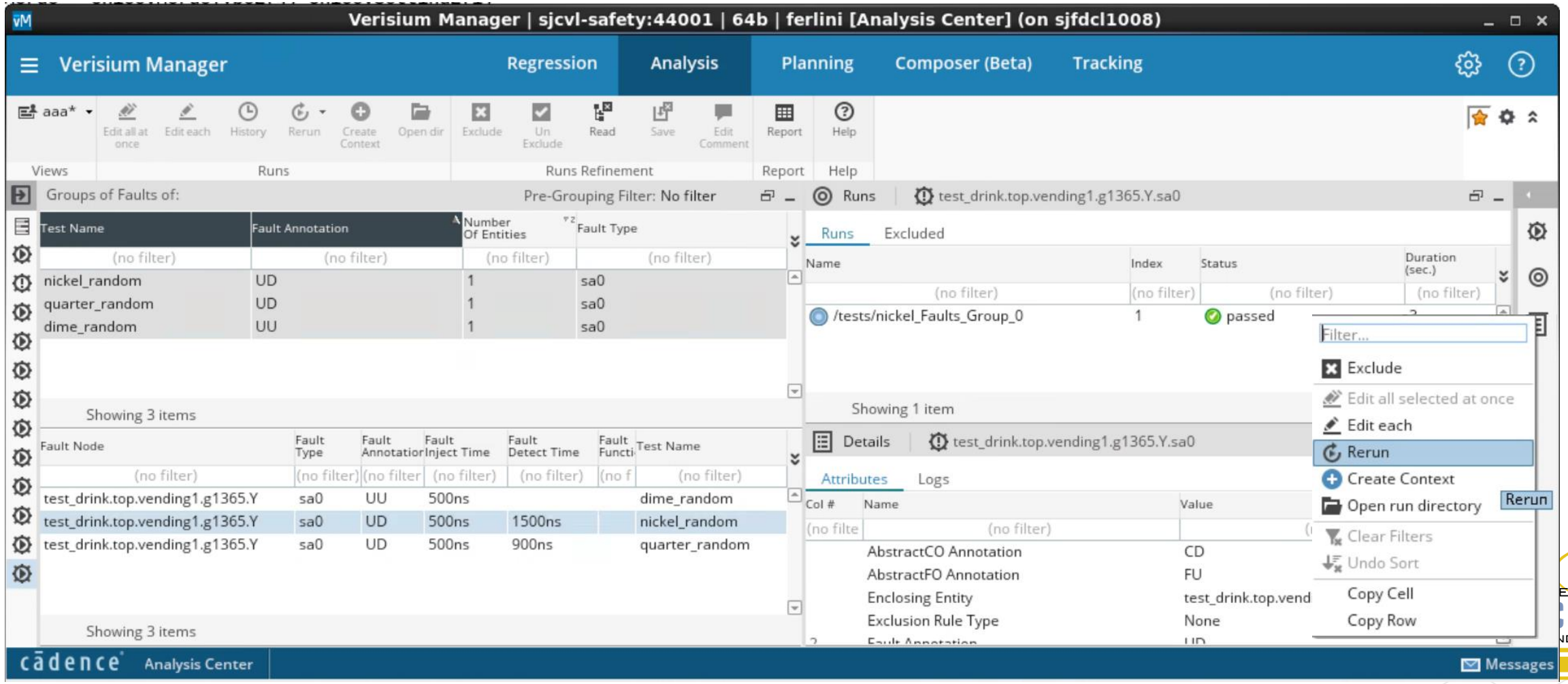
Fault Campaign Results – Annotated Fault List

Merged annotation



Fault Annotation Traceability

Result per each test

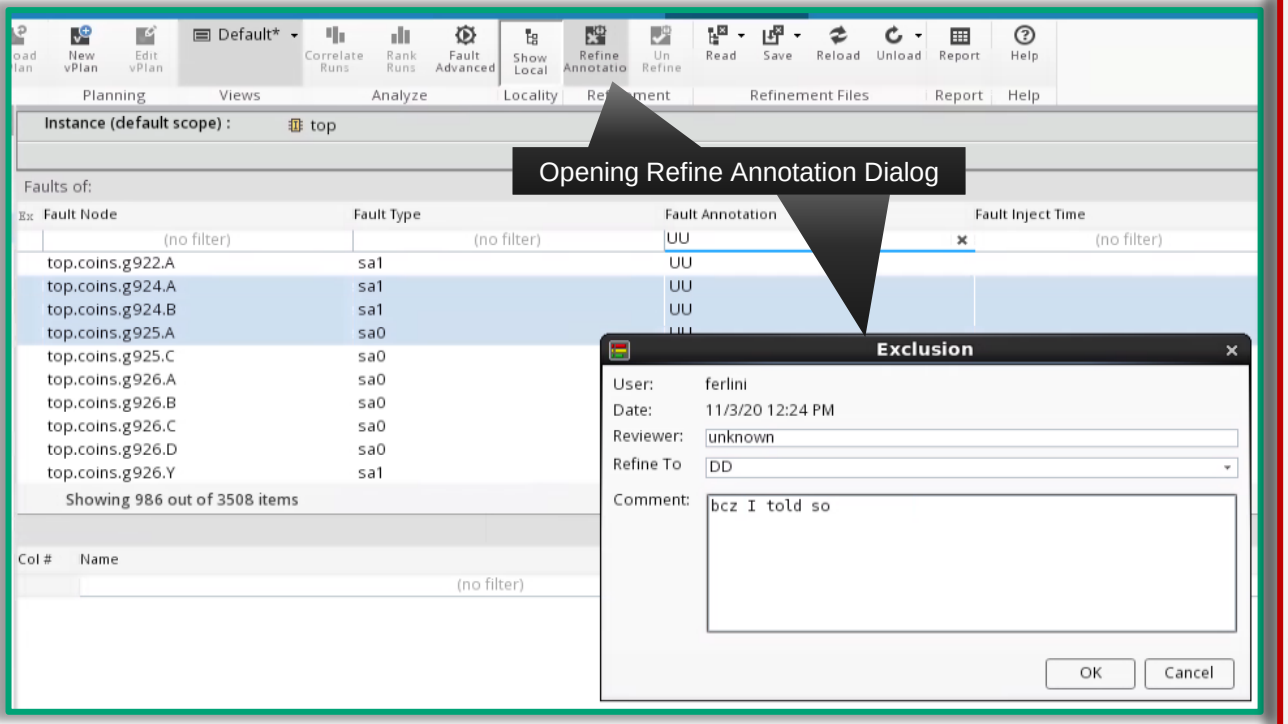


Fault Annotation Refinement

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
```

↓

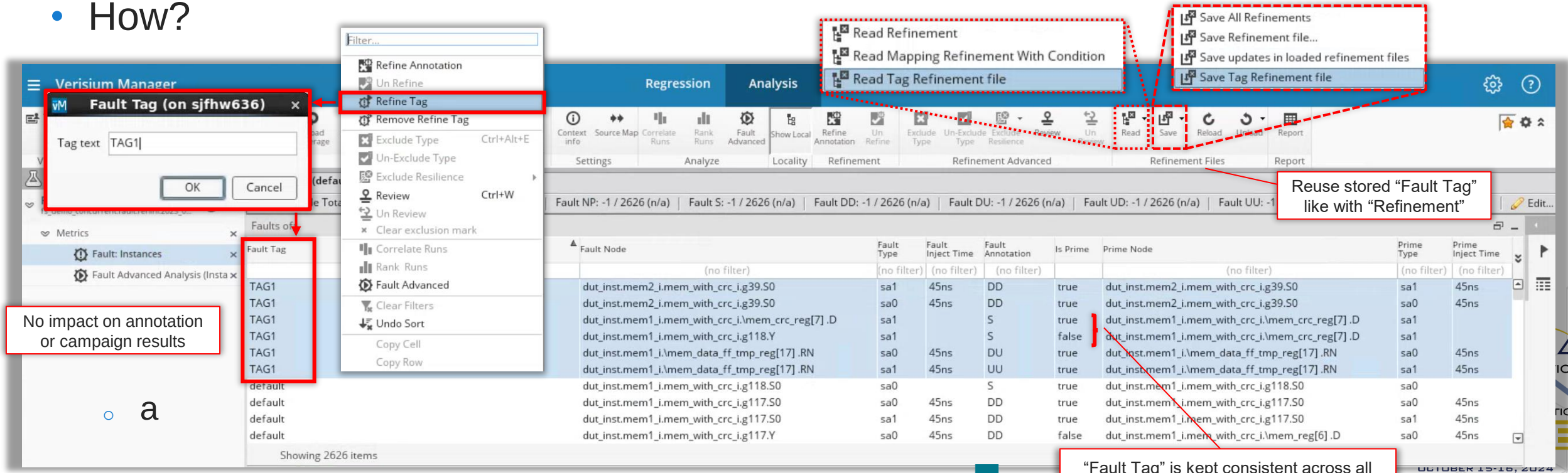
		Total Faults		Total Prime		Sample Faults		Sample Prime	
-------- INSTRUMENTATION ----		------- #	---- %	------- #	---- %	------- #	---- %	------- #	---- %
Faults		2626		2546		484		465	
Safe		1658	63.14	1658	65.12	0	0.00	0	0.00
Not Injected		479	18.24	458	17.99	35	7.23	35	7.53
Injected		489	18.62	430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


Prepare → Minimize → Execute → Report → Analyze

Fault Tagging

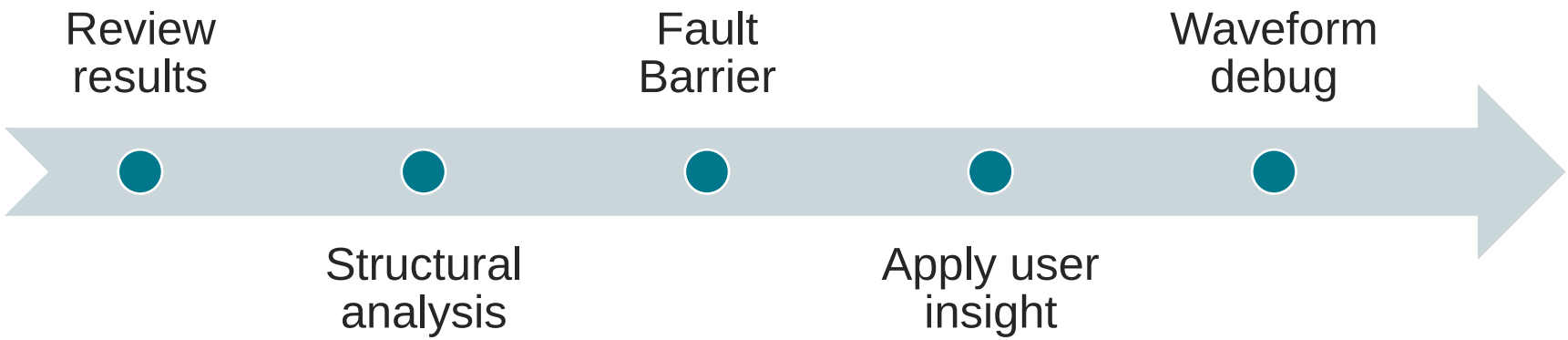
- What?
  - User-editable (string) attribute per fault metric element
- Why?
  - Support post-campaign analysis (debug, refinement, etc.) by tagging relevant faults
    - Logically gather faults even if they do not share a common attribute value (e.g., hierarchy, annotation)
- How?



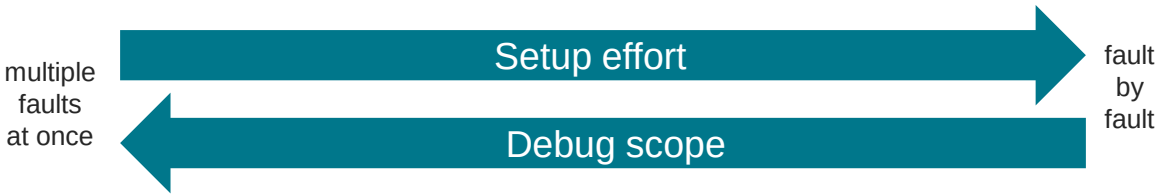
  - a

Fault Campaign Debug

Fault Campaign Closure

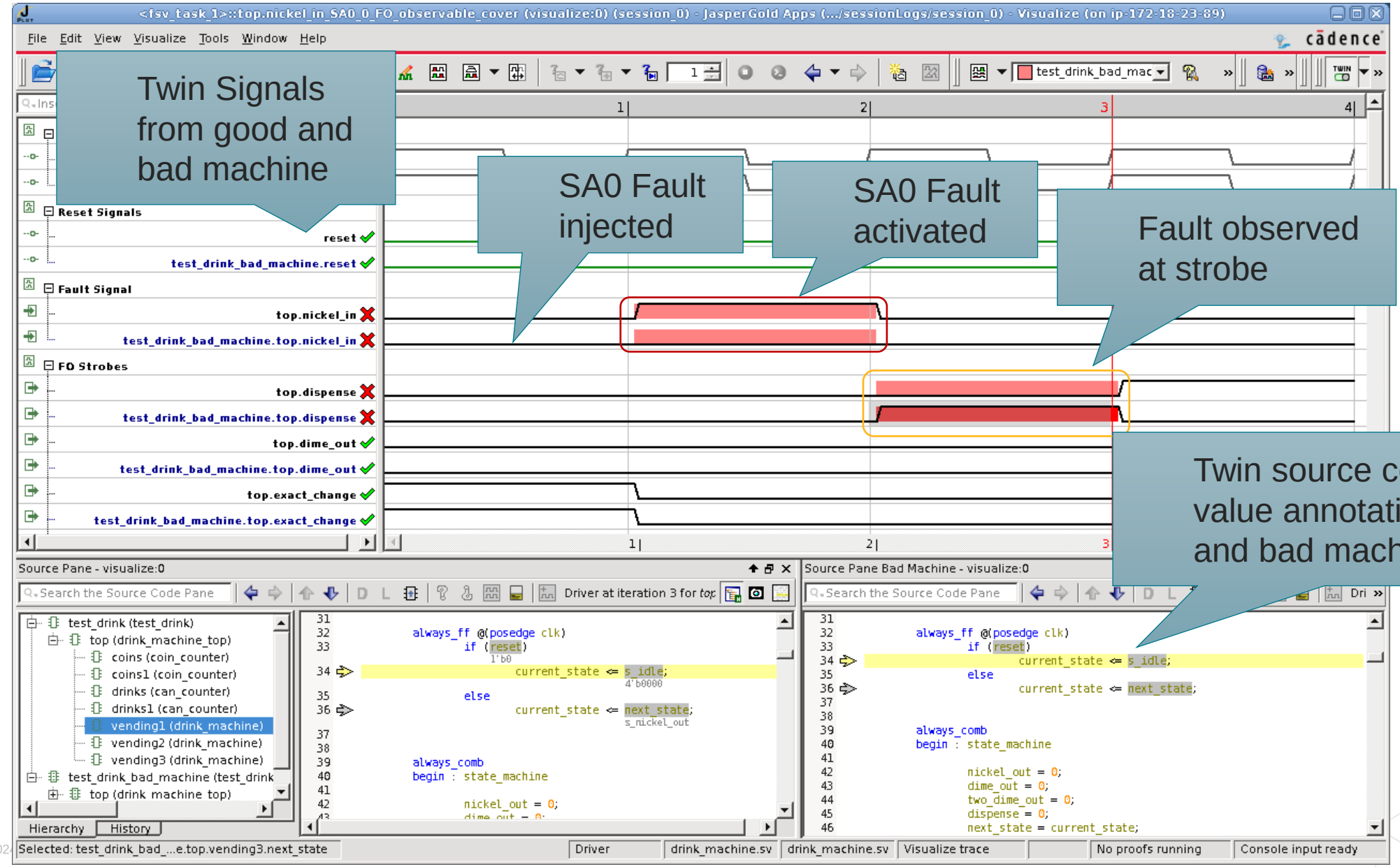


GOAL: minimize and guide the effort towards DC closure



2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE
MUNICH, GERMANY
OCTOBER 15-16, 2024
cadence

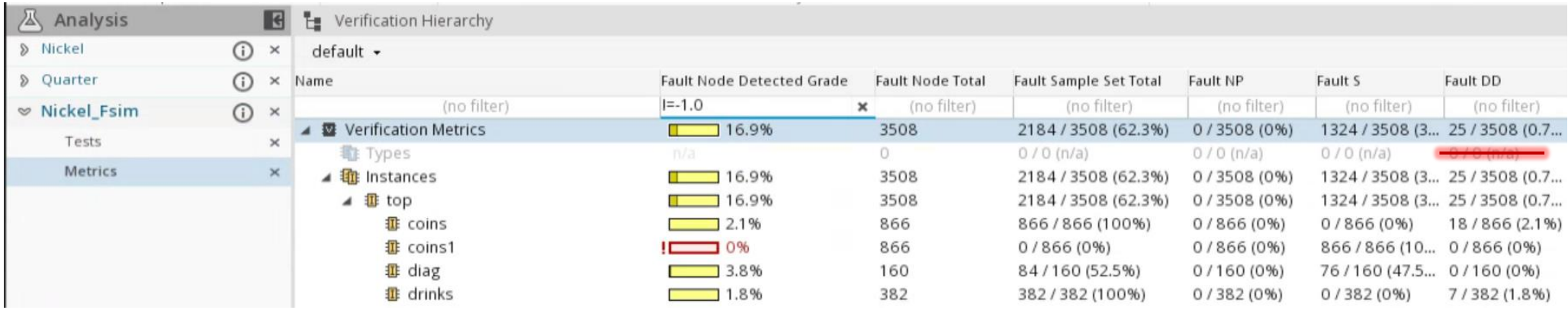
FSV Formal – Visualize Fault Detection Traces



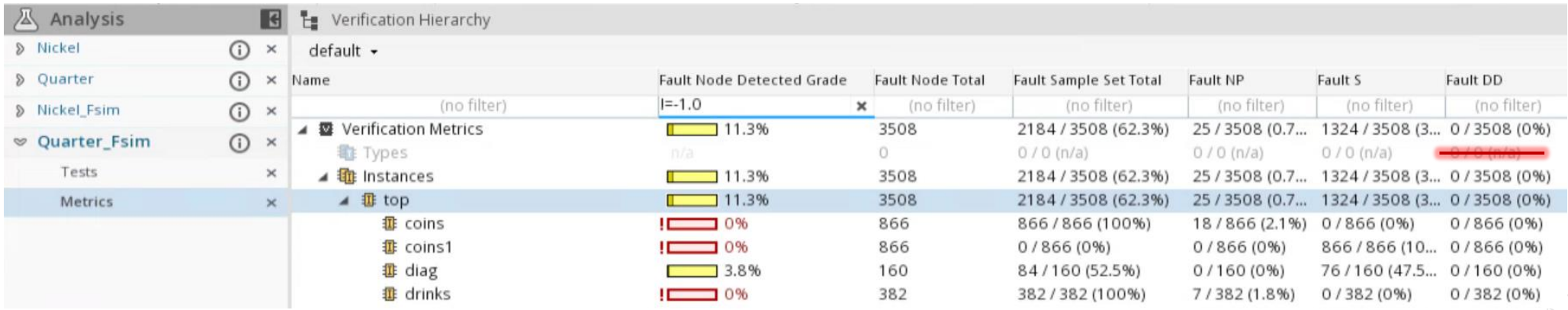
Fault Merged Annotation Per Each Test

Add / Remove tests

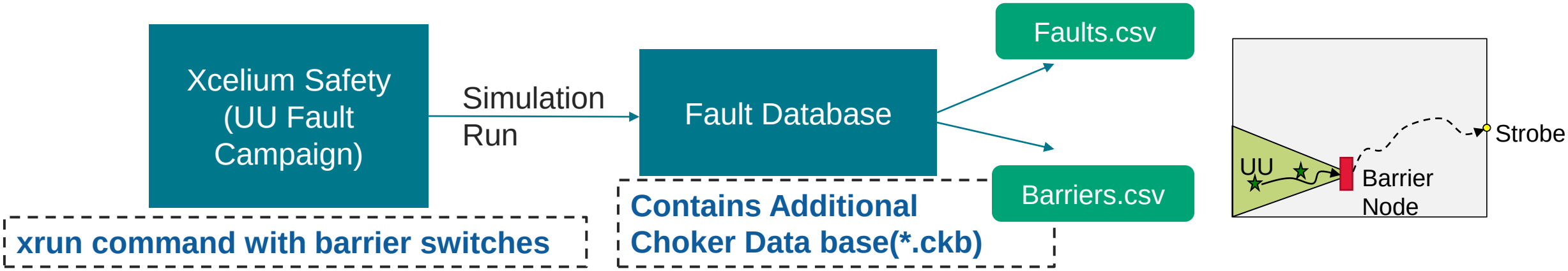
- Test 1 – Nickel



- Test 2 - Quarter



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)**

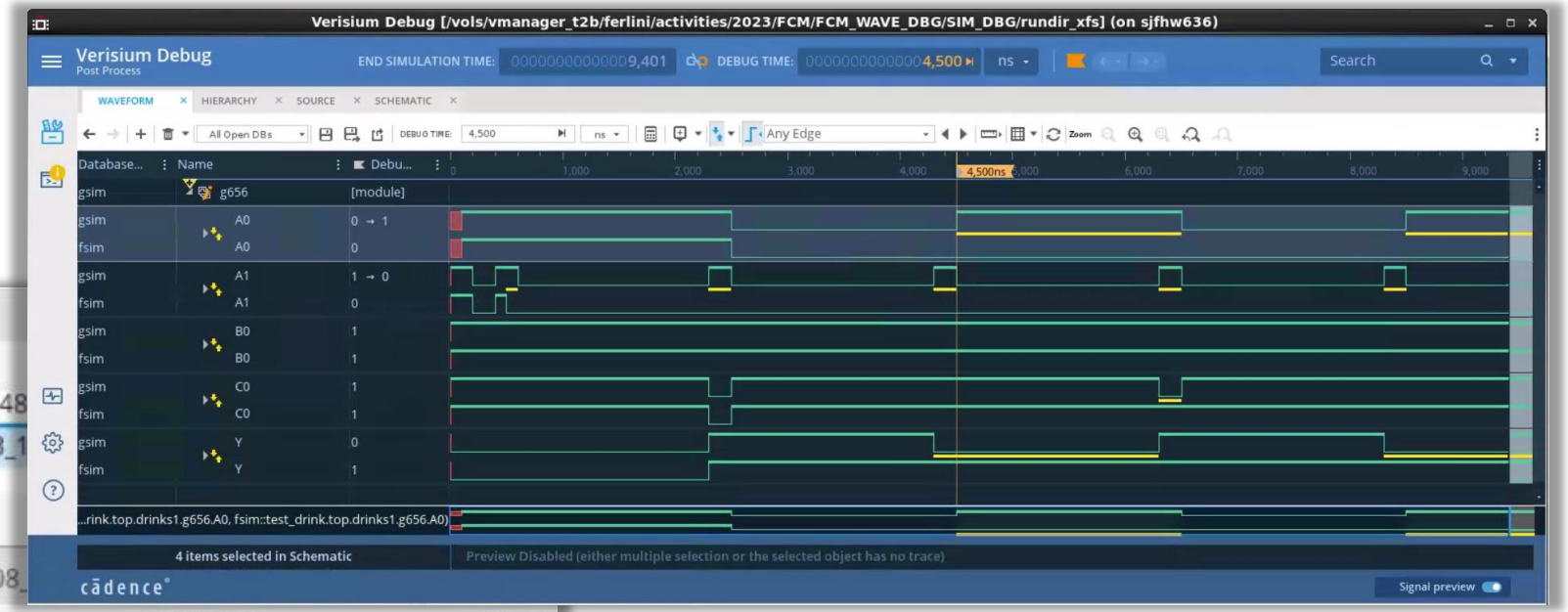
```
Barrier ID,Barrier Node,FanIn Strength,Faults
1,test_drink.top.coins.g1824__4547.Y,2,{1 2}
2,test_drink.top.coins.g1823__1474.Y,2,{1 2}
3,test_drink.top.coins.g1822__3772.Y,2,{1 2}
4,test_drink.top.coins.g2634__7675.CO,1,{2}
5,test_drink.top.coins.g2588__1474.Y,1,{2}
```

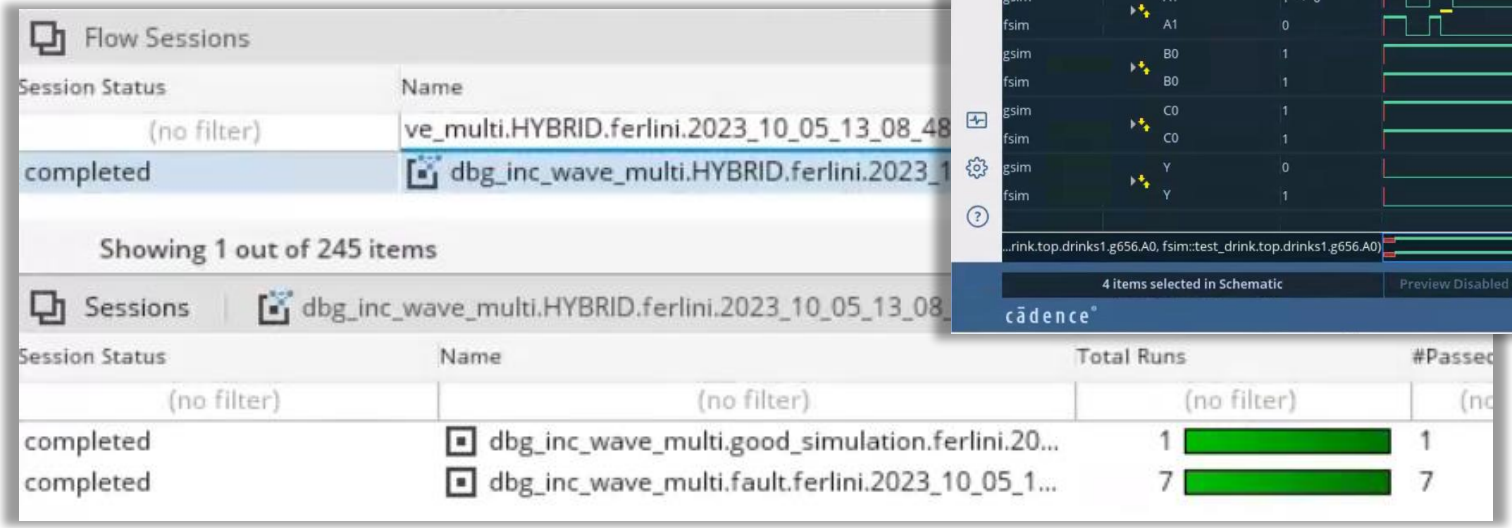
- **Snippet of faults.csv (Faults to Barrier Mapping)**

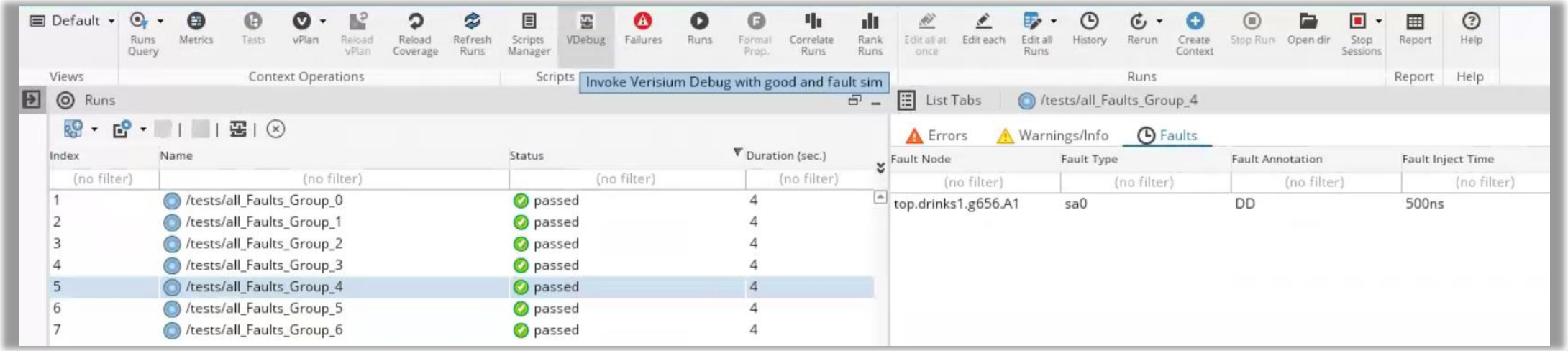
```
Fault ID,Fault Node,Fault Type,Fault Injection Time,FanOut Strength,Barriers
1,test_drink.top.coins.RC_CG_HIER_INST1.RC_CGIC_INST.E,SA1,402NS,3,{1 2 3}
2,test_drink.top.coins.RC_CG_HIER_INST1.RC_CGIC_INST.ECK,SA1,402NS,5,{1 2 3 4 5}
```

Waveform Debug

Incremental Campaign FCM integration

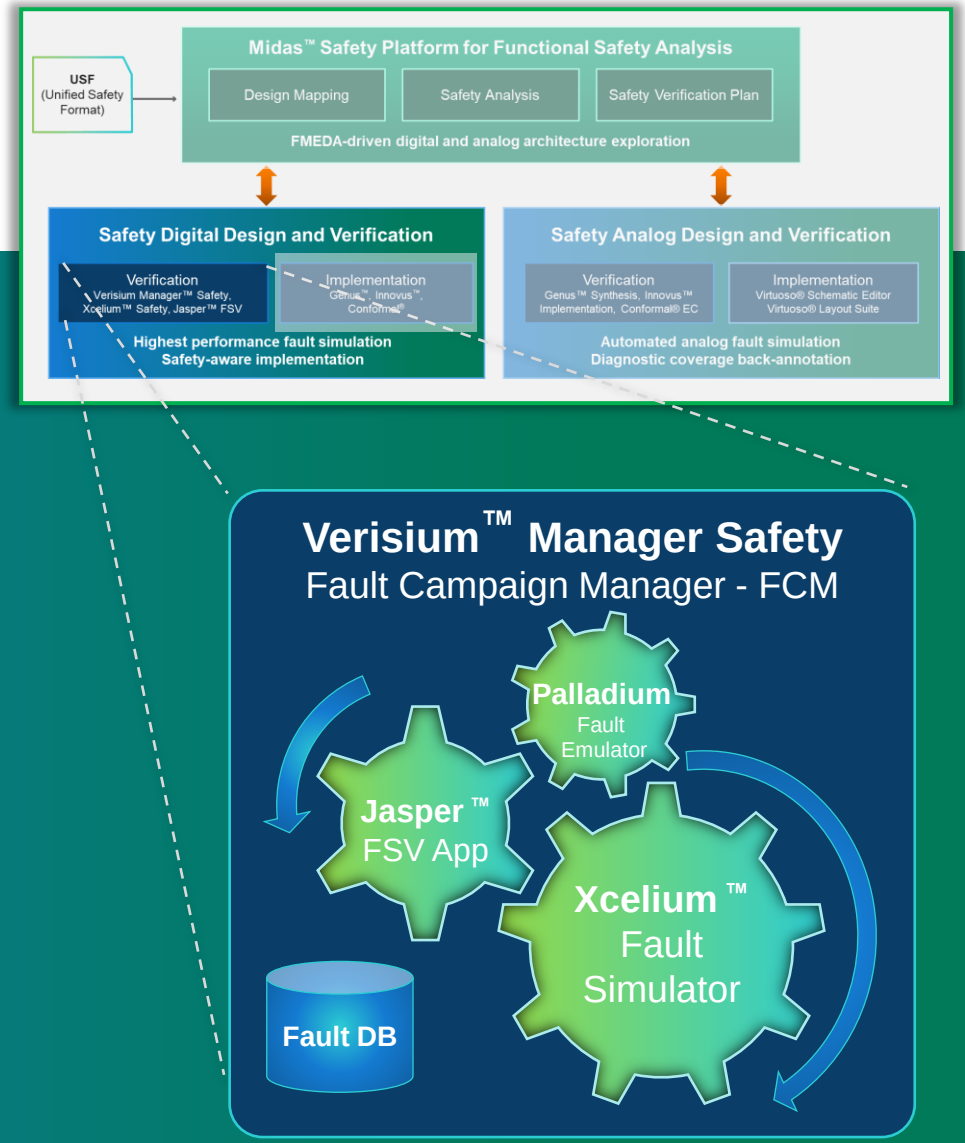






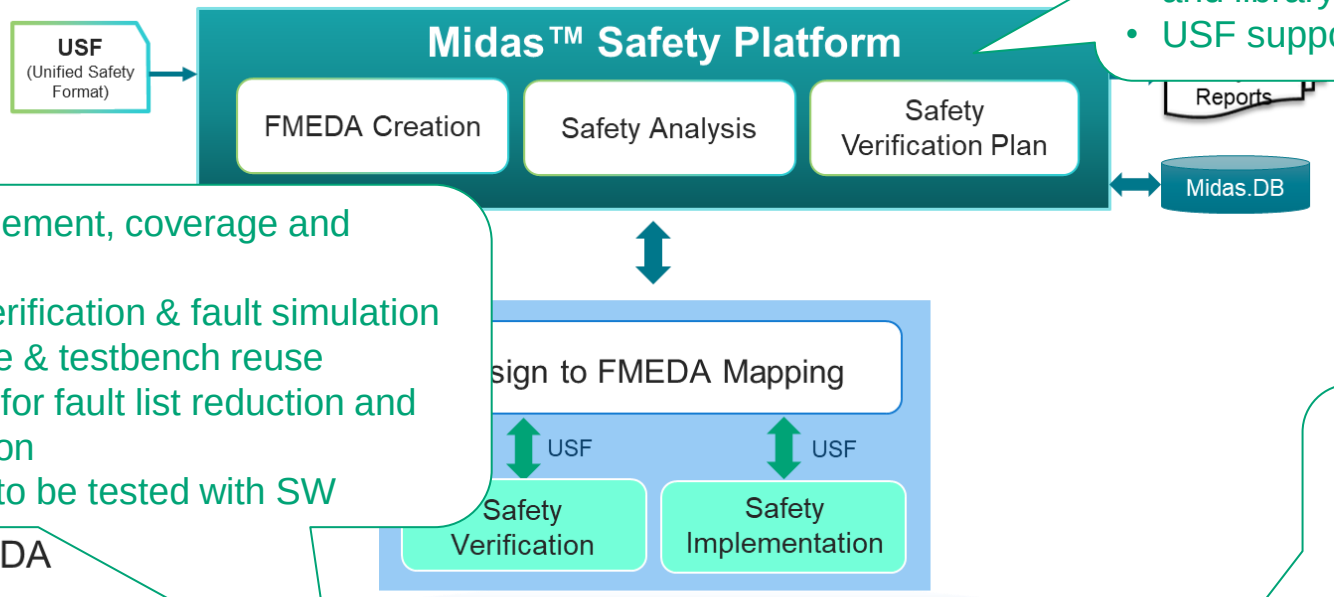
Digital Safety Verification
Summary

- ✓ 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

Verisium™ Safety
Digital Fault Campaign Management

Jasper™
Fault Testability

Xcelium™
Fault Simulation

Palladium®
Fault Emulation

Genus™
Safety Mechanism Insertion

Innovus™
Safety Mechanism Insertion

Virtuoso® ADE Assembler
Analog Fault Campaign Management

Legato™ (Spectre®/AMS Designer)
Fault Simulation

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