2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

MUNICH, GERMANY
DECEMBER 6 - 7, 2022

Achieving system dependability: the role of automation and scalability

Teo Cupaiuolo, Synopsys
Paul Baron, Melexis
Ghani Kanawati, Arm

accellera SYSTEMS INITIATIVE

Agenda

- Requirements, challenges and opportunities for automation (Synopsys)
- Early RTL fault injection for automotive ASIL D sensors (Melexis)
- Mitigating Soft Error Impact on System Dependability (Arm)

SYNOPSYS®

2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

MUNICH, GERMANY
DECEMBER 6 - 7, 2022

Requirements, challenges and opportunities for automation

Teo Cupaiuolo, Functional Safety Solutions Engineer, Synopsys

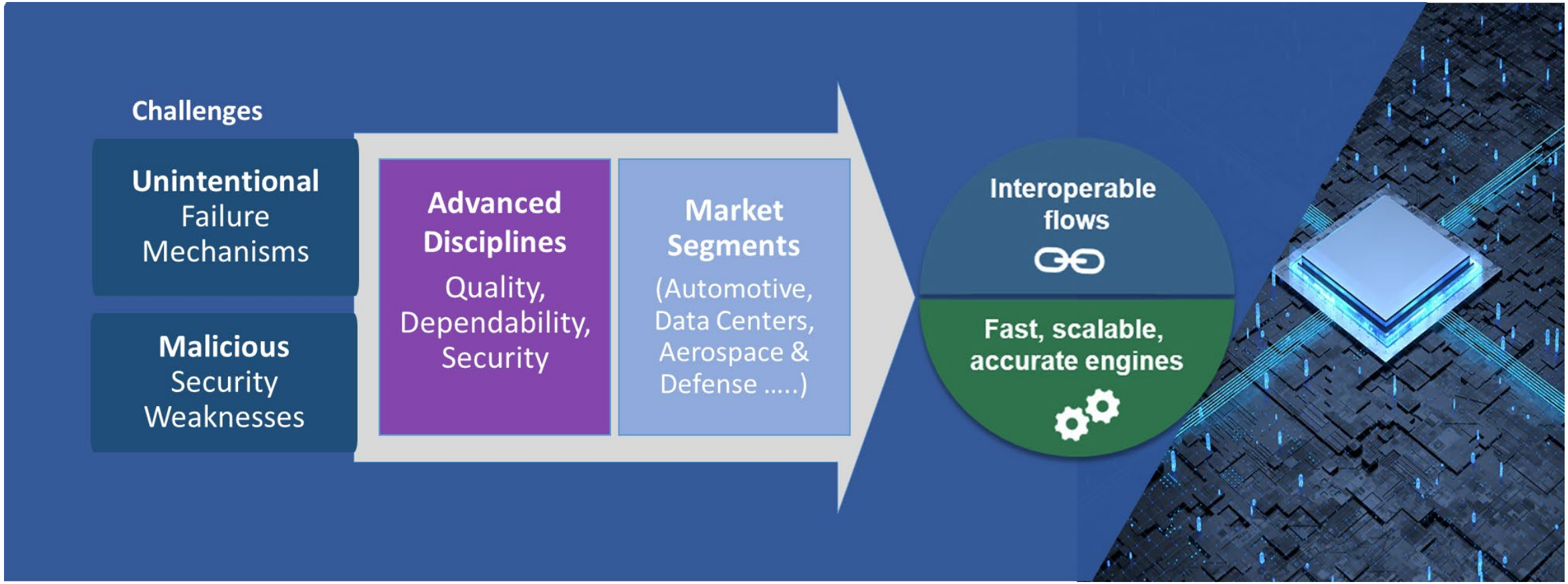
accellera SYSTEMS INITIATIVE

Agenda

- EDA beyond PPA: introducing new metrics
- Synopsys EDA Solution for Functional Safety
  - Random and Systematic Failures
  - Analysis, Implementation and Verification
  - Tool certification
- Conclusion

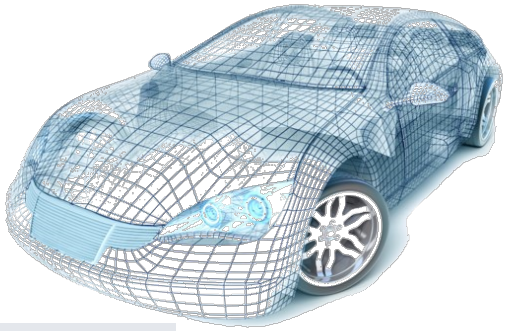
EDA Beyond PPA

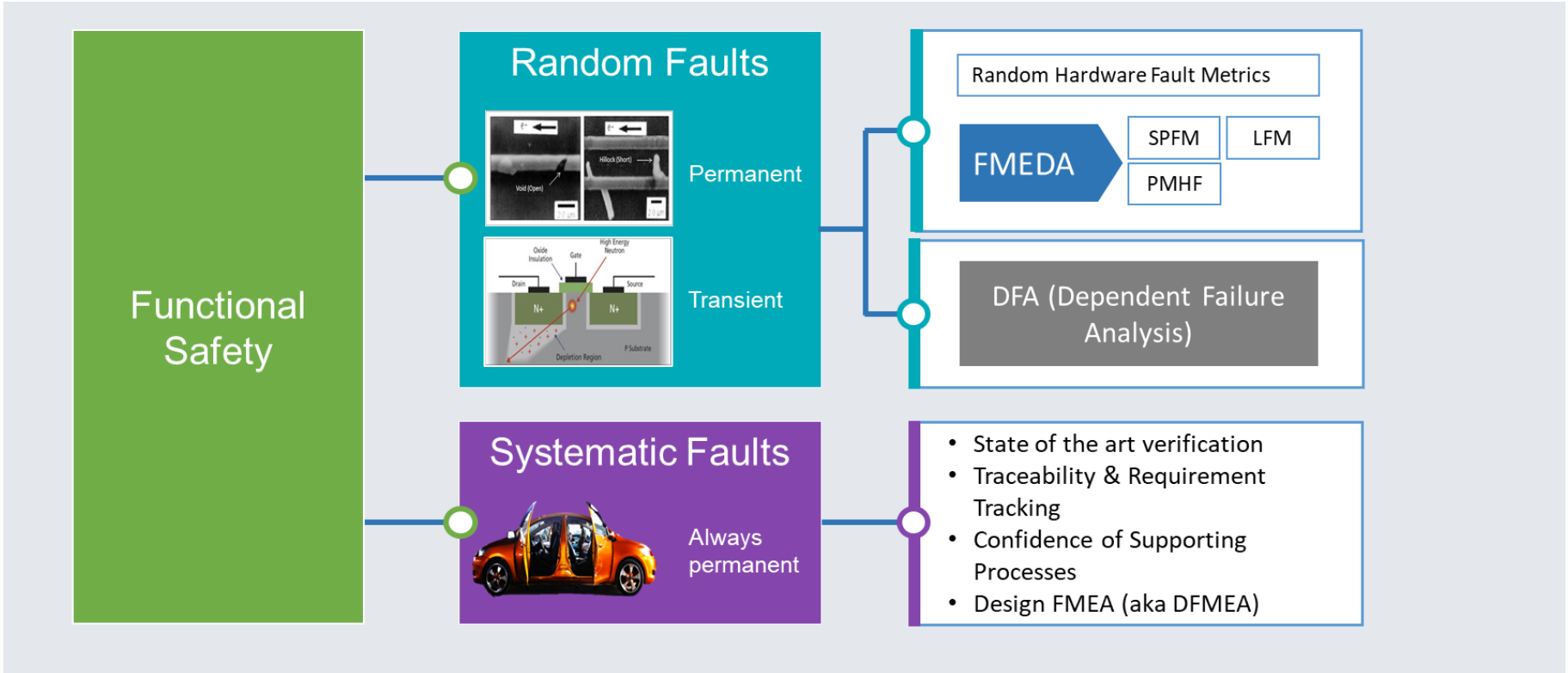
New Metrics and Requirements for Hardware Design



Functional Safety in a Nutshell

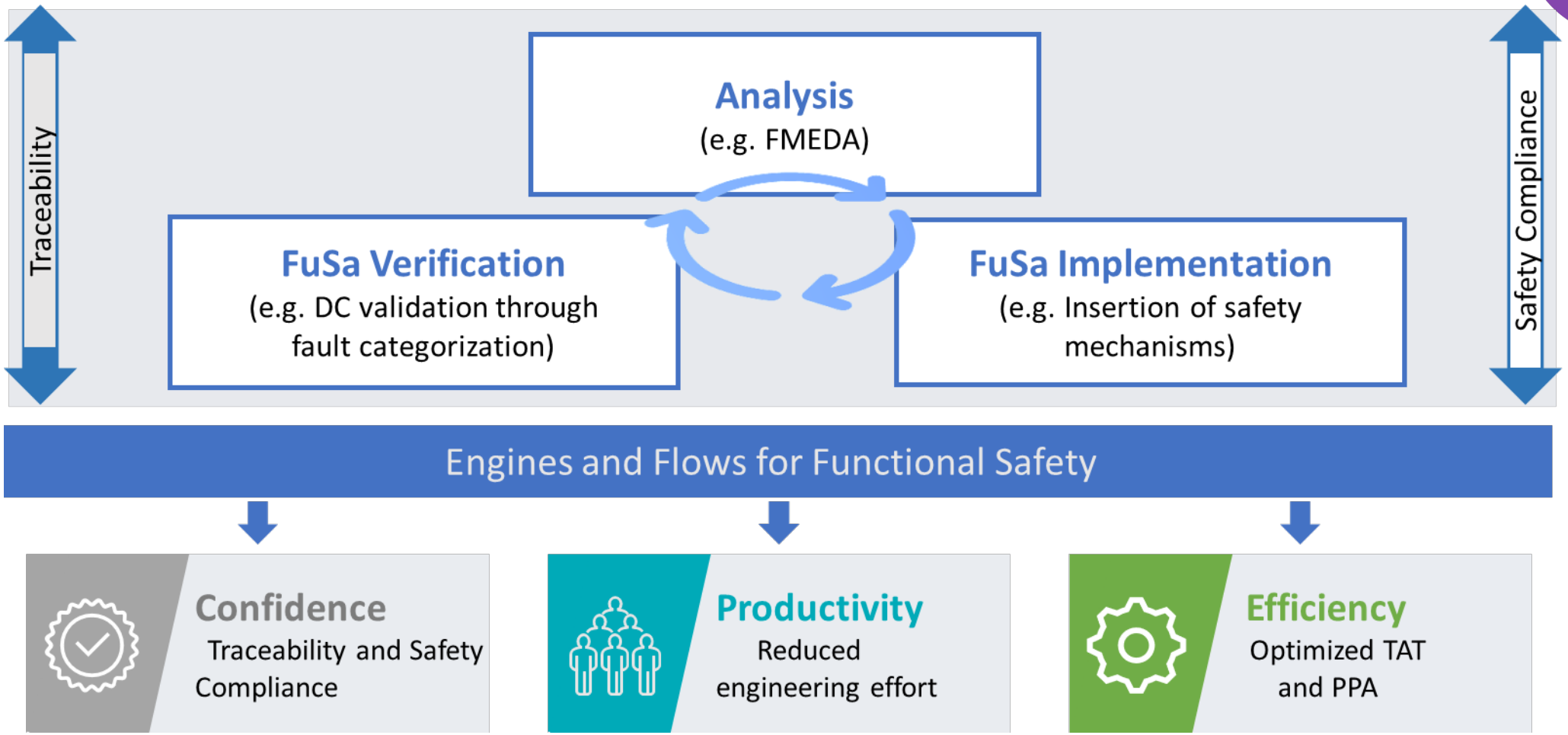
Functional Correctness & Controlled Impact of Defects



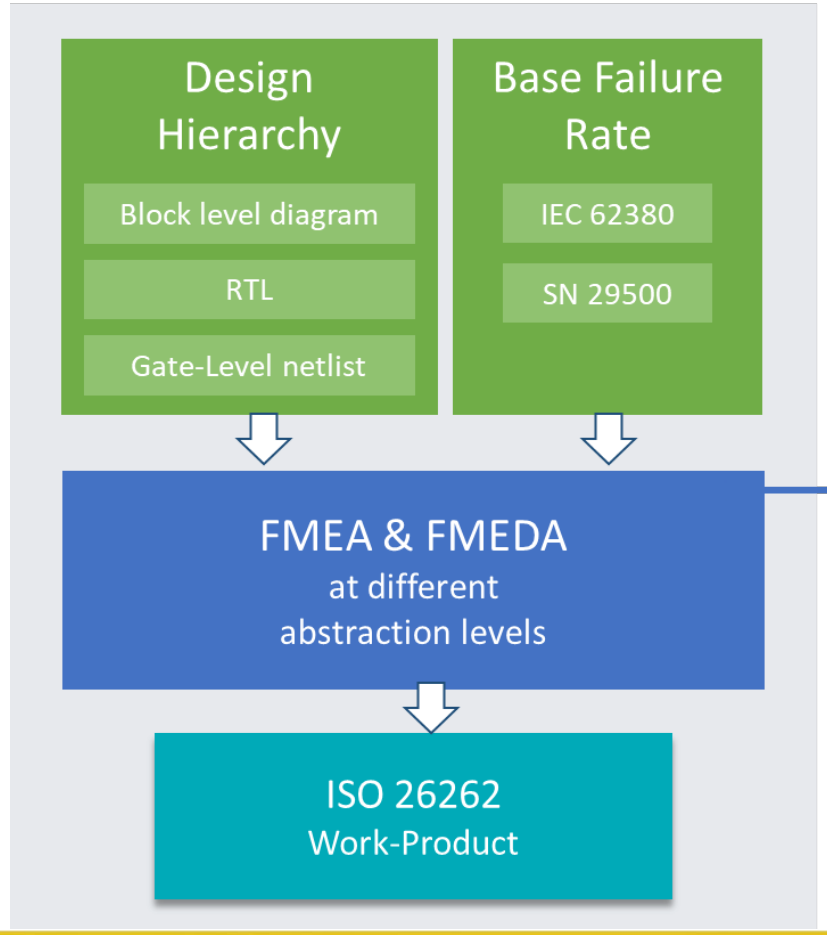


Safety is the Reduction of Risk Caused by Electric and Electronic Systems Malfunctions

Functional Safety as a New Metric in RTL2GDS Flow



FuSa Analysis & Exploration



FMEDA Analysis

- At IP, sub-system and SoC level

Safety Design Exploration

- What-If & Use Case

Automation

- Early design's statistics estimation (RTL) and accurate (netlist)
- Design's safety parameters: AVF, Failure Mode Distribution
- SM insertion guidance
- Fault campaign definition and results back-annotation

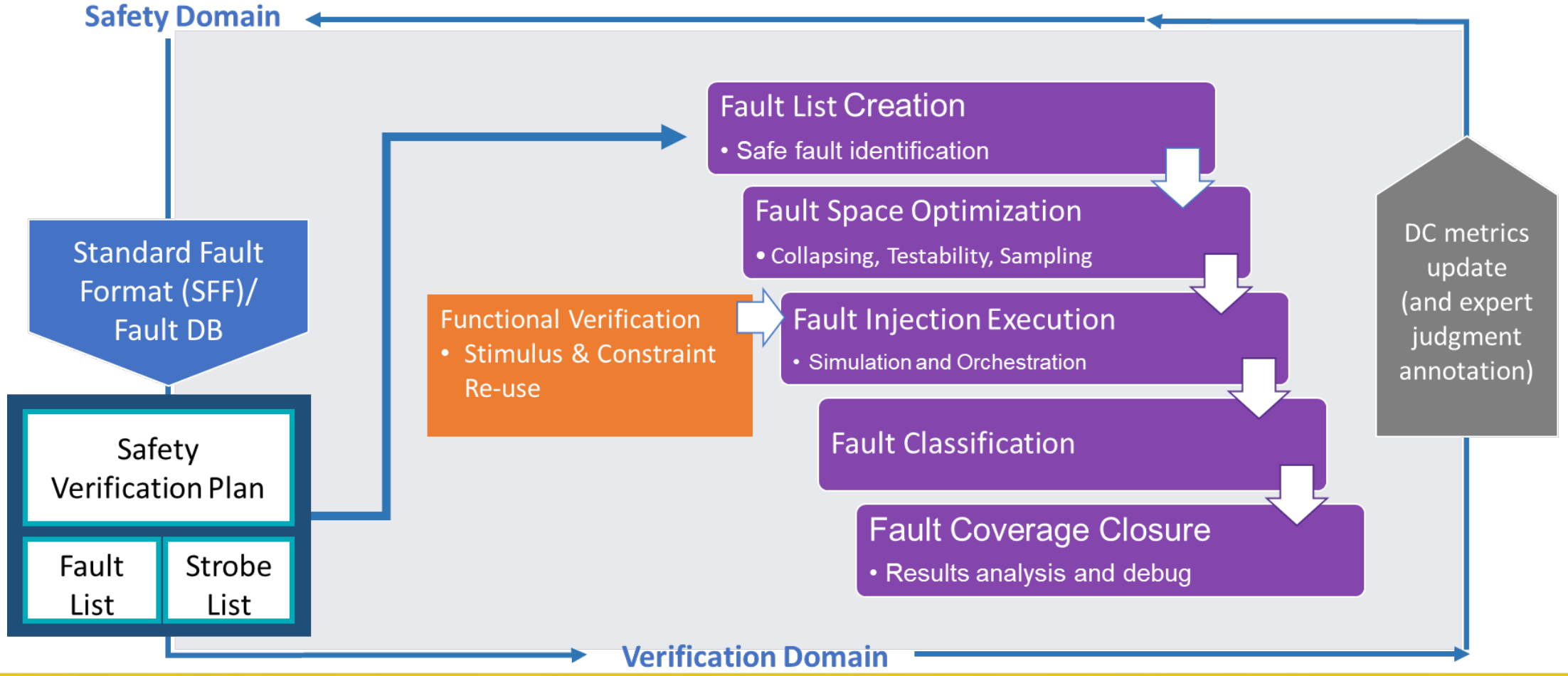
ISO 26262 compliancy

- Time and requirement-based Traceability
- Base Failure Rate calculation

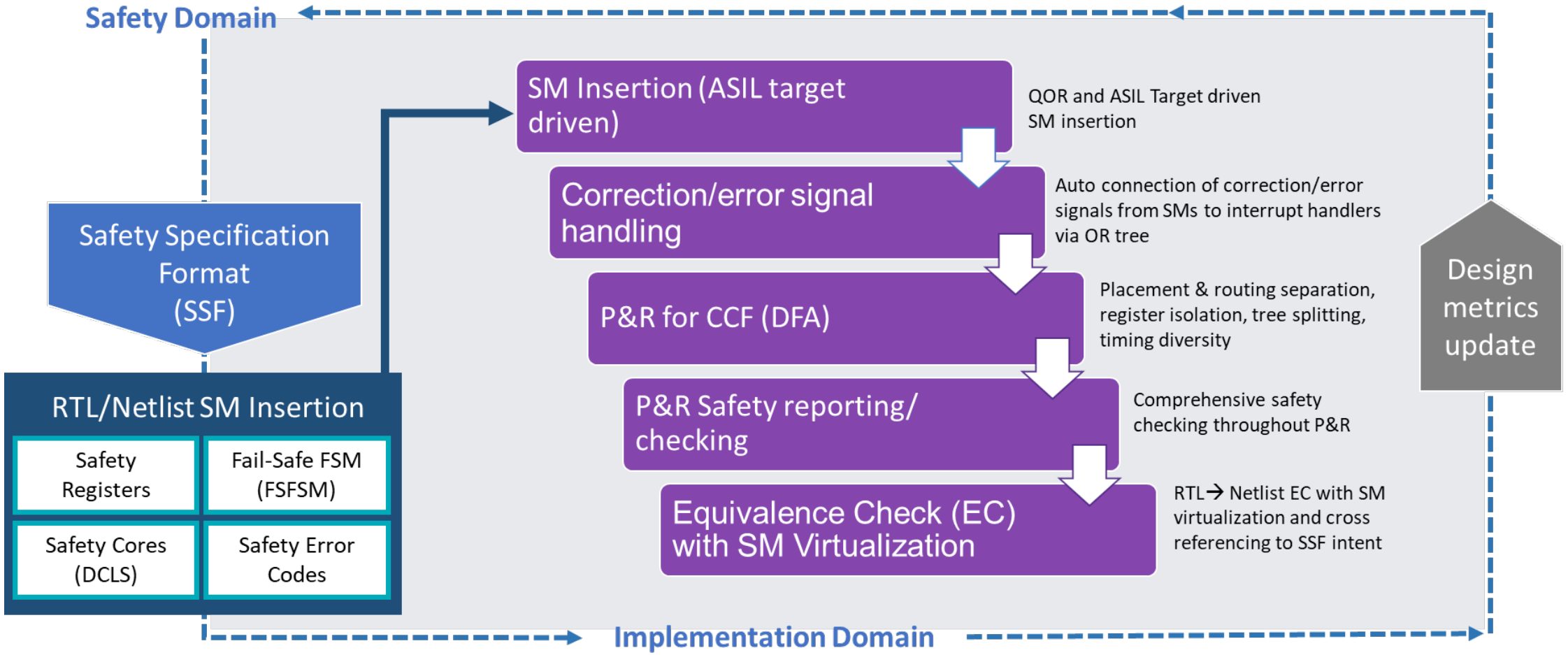
Robustness

- Access control
- Data confidentiality, prevent tampering

Efficient and Faster FuSa Verification

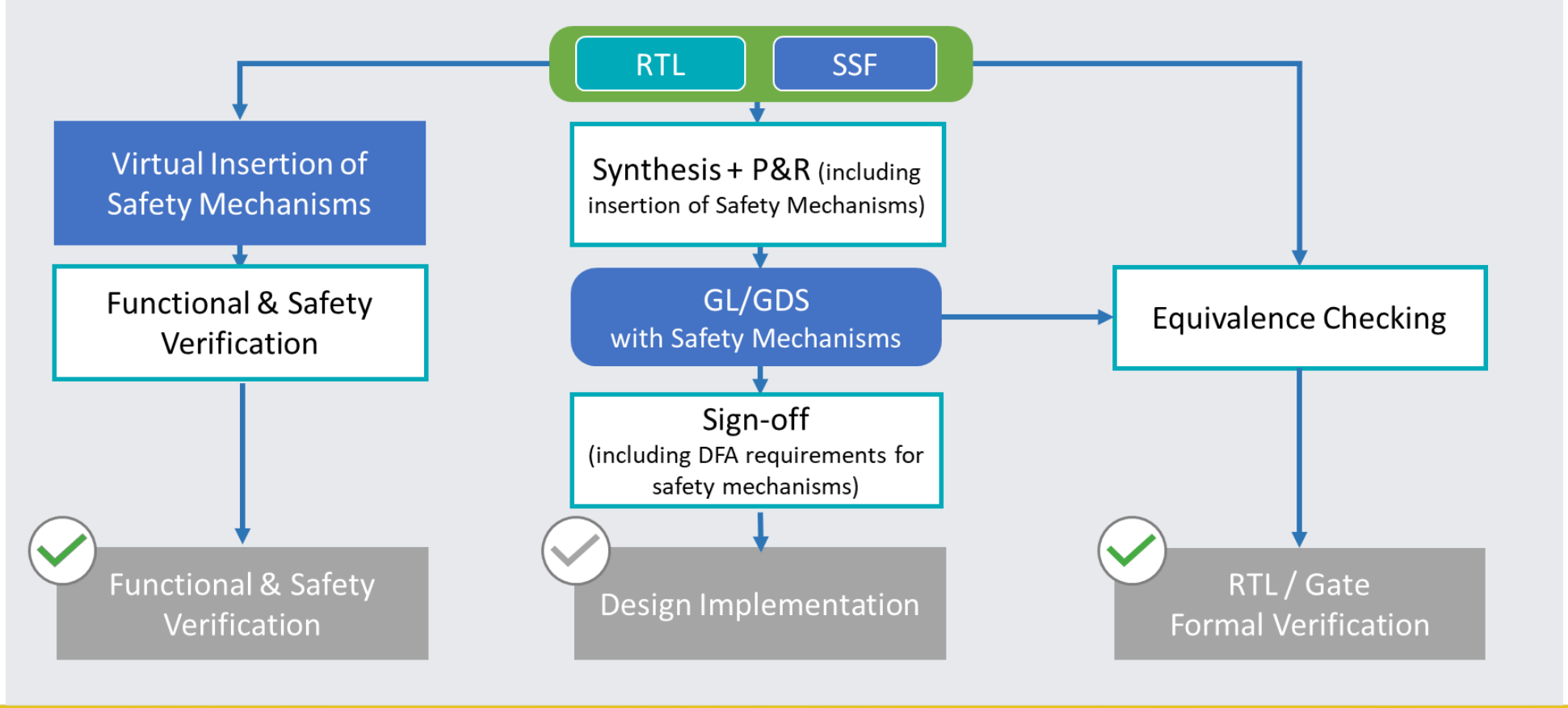


FuSa Implementation



Holistic Support of Safety Mechanisms

Analysis, Verification and Implementation



FuSa Analysis Challenges

Challenge 1:
Scalability

- Design Data Extraction
- SoC FMEDA
- Self-consistency design mapping (rule checks)
- Multiple concurrent user
- Maintenance

Challenge 2:
Automation

- Architectural Vulnerability Factor (AVF)
- Failure Mode Distribution
- Safeness for permanent faults
- Integration with Verification and Implementation tool

Challenge 3:
Supporting evidence for ISO 26262 (work products)

- Fully traceability and connection with Requirement Management System (RMS)
- Failure Mode and Safety Mechanisms library compliant to ISO-26262
- Versioning & review workflow

Productivity
Reduced engineering effort

Efficiency
Optimized TAT and PPA

Confidence
Traceability and Safety Compliance

FuSa Verification Challenges

Challenge 1:
Complexity & Scalability

- FuSa adds significant overhead to verification
- Scalability of solution for large SoCs
- Tool guided analysis for diagnostic coverage closure

Challenge 2:
Throughput

- Runtime and Compute Power
- Deployment of the correct tools at the appropriate stage of verification
- Automation of multiple technologies

Challenge 3:
Supporting evidence for ISO 26262 (work products)

- Traceability of activities
- Connection to analysis tools for FMEDA calculation
- Deployment of certified tools and toolchains

Productivity
Reduced engineering effort

Efficiency
Optimized TAT and PPA

Confidence
Traceability and Safety Compliance

FuSa Implementation Challenges

Synopsys Solutions Based on Safety Specification Format (SSF) & EDA

Challenge 1:
Scripting / Flow Overhead

SSF

- Flow overhead removed with SSF adoption
- SSF drives native SM insertion throughout Digital Tool Chain

Productivity
Reduced engineering effort

Challenge 2:
Maintain QoR with Safety Mechanism Insertion

SSF

- Native SM insertion through all implementation stages
- Native SM support for all implementation engines (placer, legalizer, etc) enabling optimal QOR

Efficiency
Optimized TAT and PPA

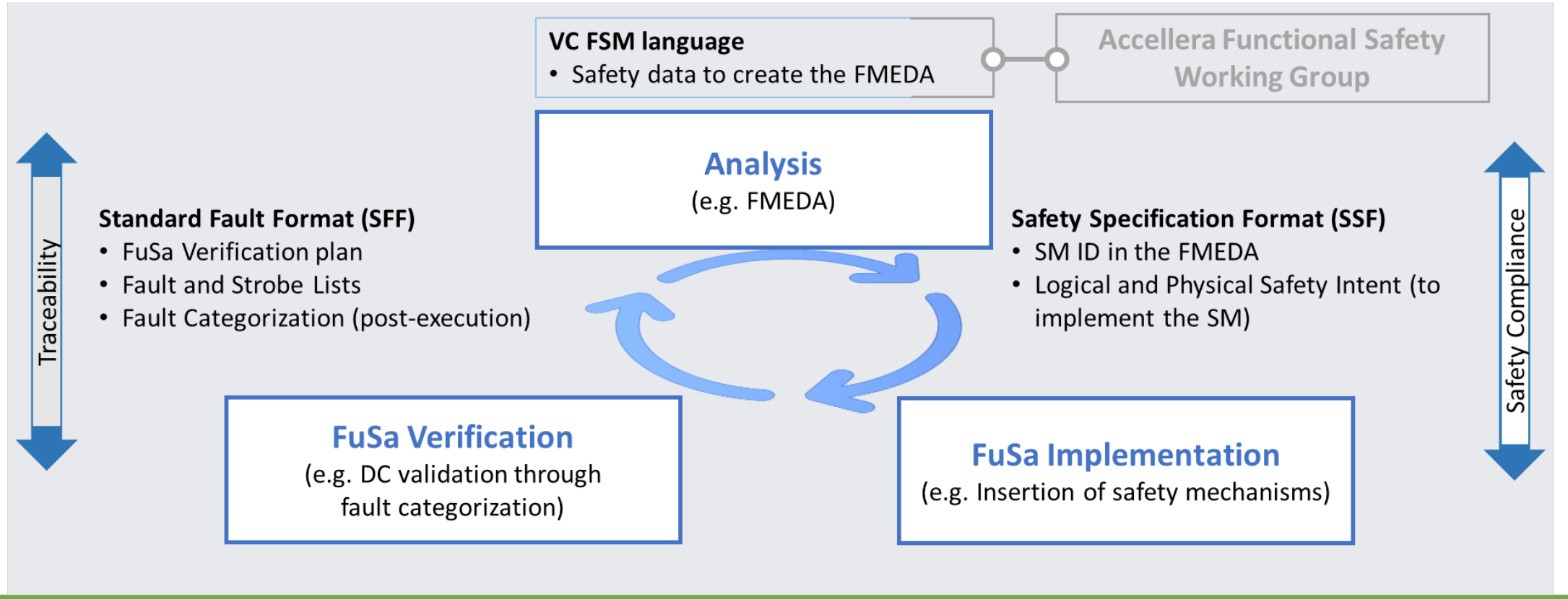
Challenge 3:
Supporting evidence for ISO 26262 (work products)

SSF

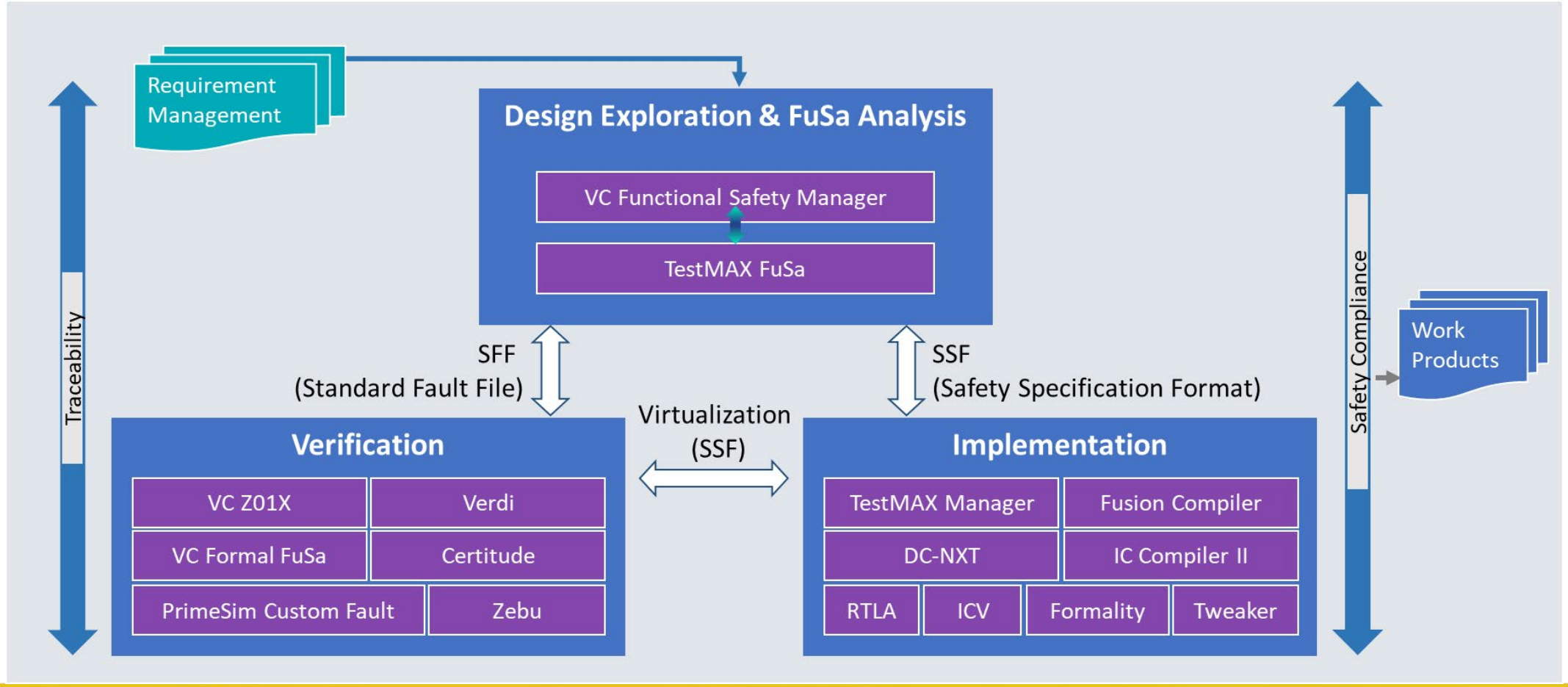
- Native safety reporting through all implementation stages
- Provides ISO 26262 supporting evidence and replaces custom checking

Confidence
Traceability and Safety Compliance

A Holistic Solution: Flows and Interoperability

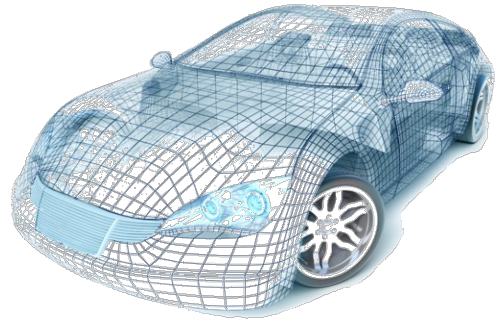


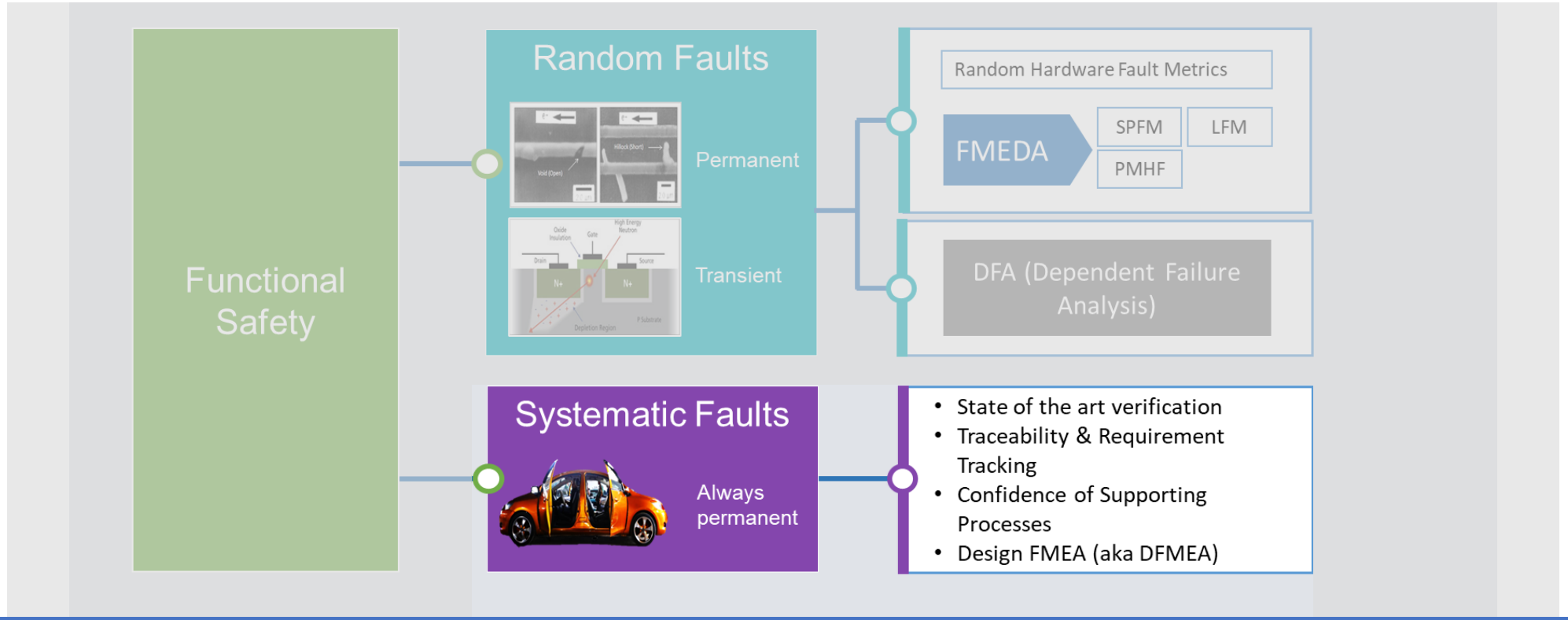
Synopsys Engines and Flows for Design-For-Safety



Functional Safety in a Nutshell

Functional Correctness & Controlled Impact of Defects





Safety is the Reduction of Risk Caused by Electric and Electronic Systems Malfunctions

Risk of Systematic Faults Is Minimized

Processes, Knowledge, Certified Tool Chains

A function can't be considered safe if it is buggy in an unknown way

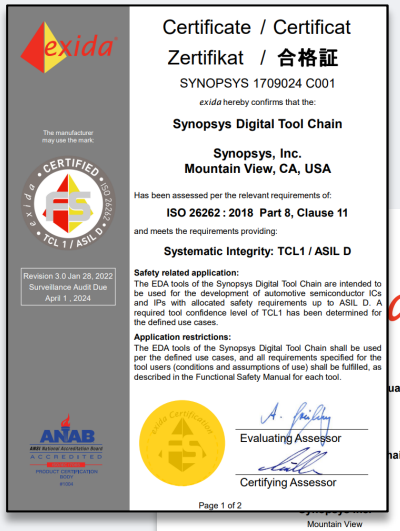
Use **state of the art tools** and **techniques for verification**

- Design architecture, modeling and implementation must follow best practices
- **D**esign **F**ailure **M**ode and **E**ffect **A**nalysis (DFMEA) must be used to recognize and evaluate potential systems, products, or process systematic failures and to define the corresponding mitigation measures
- Functional verification must be more thorough and rigorous
  - Mandatory to know the limitations of the verification techniques and to combine them
- Functional and safety requirements must be tracked and linked to the actual design specification and functional verification
- Tools and processes must be evaluated
  - Both dynamic and formal functional verification must be qualified

Synopsys: Design with the Highest Confidence Level!


Certificate / Certificat
Zertifikat / 合格証
SYNOPSYS 1904101 C007
exida hereby confirms that the:
Synopsys Verification Tool Chain
Synopsys, Inc.
Mountain View, CA, USA
Has been assessed per the relevant requirements of:
ISO 26262 : 2018 Parts 8, Clause 11
and meets the requirements providing:
Systematic Integrity: Tool Qualification ASIL D
Safety related application:
The EDA tools of the Synopsys Verification Tool Chain are intended to be used for the development and verification of automotive semiconductor ICs and IPs with allocated safety requirements up to ASIL D.
Application restrictions:
The EDA tools of the Synopsys Verification Tool Chain shall be used per the defined use cases, and all requirements specified for the tool users (conditions and assumptions of use) shall be fulfilled, as described in the Functional Safety Manual for each tool.
Revision 2.0 Jun 25, 2021
Surveillance Audit Due May 1 , 2023
Evaluating Assessor
Certifying Assessor
Page 1 of 2


Certificate / Certificat
Zertifikat / 合格証
SYNOPSYS 1812004 C005
exida hereby confirms that the:
Synopsys TestMAX Tool Chain
Synopsys, Inc.
Mountain View, CA, USA
Has been assessed per the relevant requirements of:
ISO 26262 : 2018 Parts 8, Clause 11
and meets the requirements providing:
Systematic Integrity: Tool Qualification ASIL D
Safety related application:
The EDA tools of the Synopsys TestMAX Tool Chain are intended to support the testing and diagnosis of automotive ICs and IPs with allocated safety requirements up to ASIL D.
Application restrictions:
The EDA tools of the Synopsys TestMAX Tool Chain shall be used per the defined use cases, and all requirements specified for the tool users (conditions and assumptions of use) shall be fulfilled, as described in the Functional Safety Manual for each tool.
Revision 1.0 Aug 27, 2020
Surveillance Audit Due Sep 1 , 2023
Evaluating Assessor
Certifying Assessor
Page 1 of 2


Certificate / Certificat
Zertifikat / 合格証
SYNOPSYS 1709024 C001
exida hereby confirms that the:
Synopsys Digital Tool Chain
Synopsys, Inc.
Mountain View, CA, USA
Has been assessed per the relevant requirements of:
ISO 26262 : 2018 Part 8, Clause 11
and meets the requirements providing:
Systematic Integrity: TCL1 / ASIL D
Safety related application:
The EDA tools of the Synopsys Digital Tool Chain are intended to be used for the development of automotive semiconductor ICs and IPs with allocated safety requirements up to ASIL D. A required tool confidence level of TCL1 has been determined for the defined use cases.
Application restrictions:
The EDA tools of the Synopsys Digital Tool Chain shall be used per the defined use cases, and all requirements specified for the tool users (conditions and assumptions of use) shall be fulfilled, as described in the Functional Safety Manual for each tool.
Revision 3.0 Jan 28, 2022
Surveillance Audit Due April 1 , 2024
Evaluating Assessor
Certifying Assessor
Page 1 of 2


Certificate / Certificat
Zertifikat / 合格証
SYNOPSYS 1709024 C003
exida hereby confirms that the:
Synopsys Library Tool Chain
Synopsys, Inc.
Mountain View, CA, USA
Has been assessed per the relevant requirements of:
ISO 26262 : 2011 Parts 8, Clause 11
and meets the requirements providing:
Systematic integrity: TCL1 / ASIL D
Safety related application:
The EDA tools of the Synopsys Library Tool Chain are intended to be used for the preparation of silicon cell libraries, which are used for the development of automotive ICs and IPs with allocated safety requirements up to ASIL D. A required tool confidence level of TCL1 has been determined for the defined use cases.
Application restrictions:
The EDA tools of the Synopsys Library Tool Chain shall be used per the defined use cases, and all requirements specified for the tool users (conditions and assumptions of use) shall be fulfilled, as described in the Functional Safety Manual for each tool.
Revision 1.0 July 2, 2018
Surveillance Audit Due August 1 , 2021
Evaluating Assessor
Certifying Assessor
Page 1 of 2

Verification Tool Chain Systematic Certification

https://www.exida.com/SAEL-Safety/synopsys-verification-tool-chain

TestMAX Tool Chain Certification

https://www.exida.com/SAEL-Safety/Synopsys-TestMAX-Tool-Chain

Digital Tool Chain Certification

https://www.exida.com/SAEL-Safety/Synopsys-Digital-Tool-Chain

Library Tool Chain Certification

https://www.exida.com/SAEL-Safety/synopsys-library-tool-chain

Summary

- EDA beyond PPA…introducing new metrics
- EDA Solution for Functional Safety
  - Random and Systematic Failures
  - Analysis, Implementation and Verification
  - Tool certification
- Functional Safety methodologies are still evolving
  - Synopsys is focused on innovation
  - And interested in collaborating with partners

Melexis
INSPIRED ENGINEERING

2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE
MUNICH, GERMANY
DECEMBER 6 - 7, 2022

Early RTL fault injection for automotive ASIL D sensors

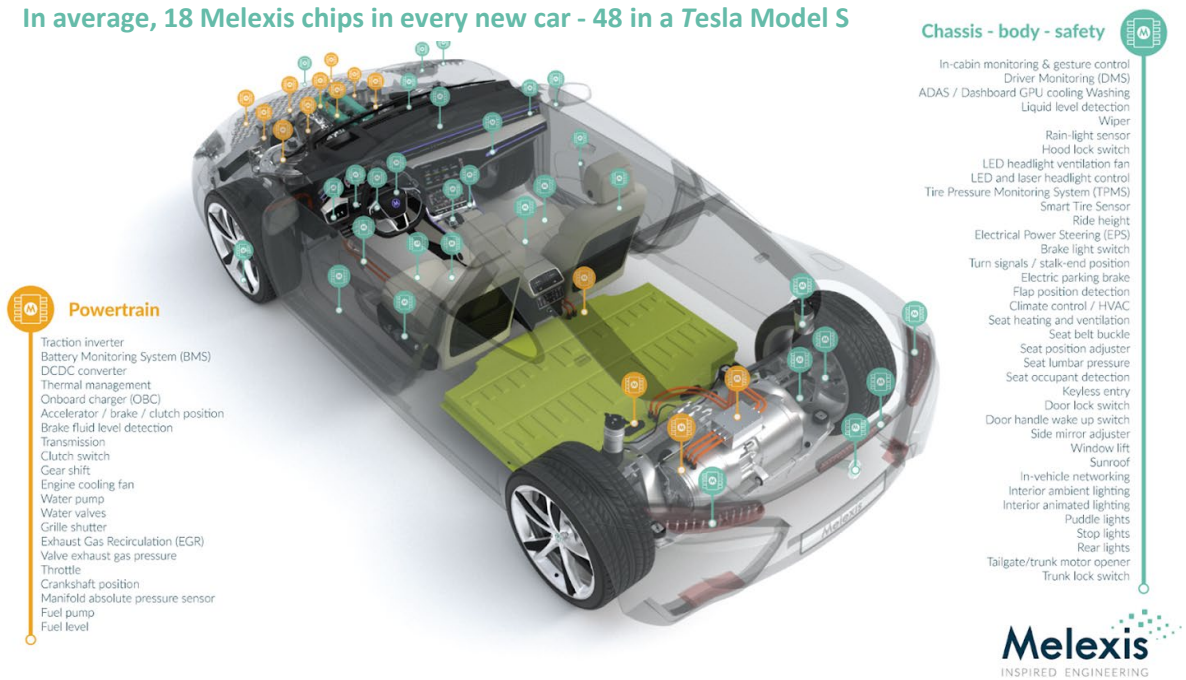
Paul Baron, Digital Design Engineer

Philippe Laugier, Digital Competence Center manager

accellera SYSTEMS INITIATIVE

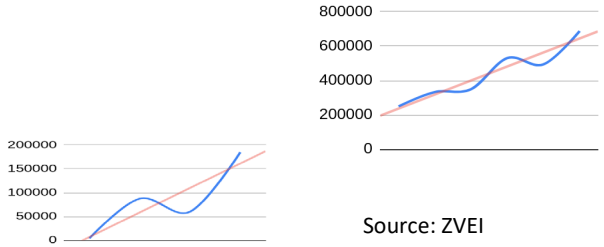
Our activities have high safety needs

- Magnetic position **sensors**
- Inductive position **sensors**
- Current **sensors**
- Pressure **sensors**
- Tire monitoring **sensors**
- Temperature **sensors**
- Optical **sensors**
- **Sensor** interfaces



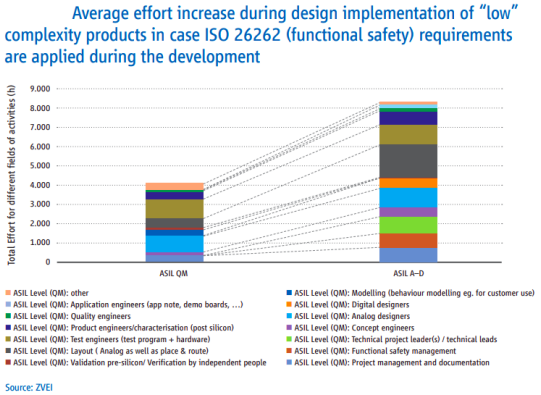
Automotive context

- IC complexity increase



Source: ZVEI

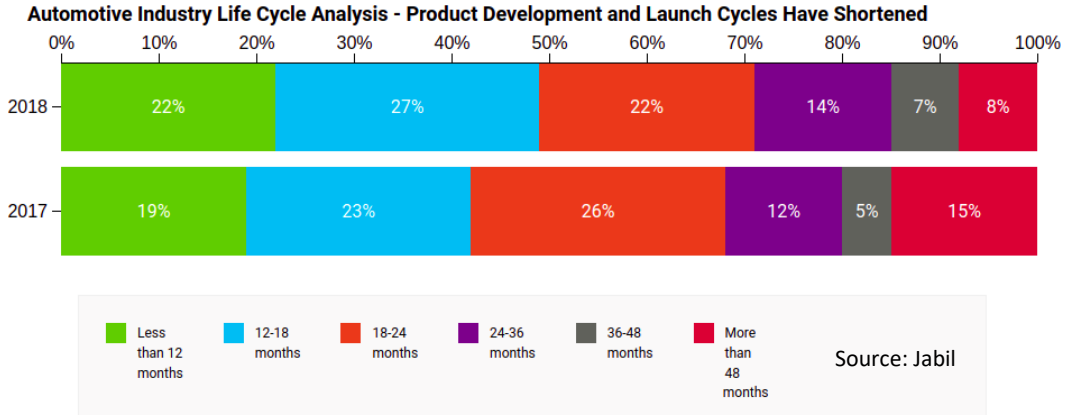
- More safety critical applications (ADAS)



Less time available

More time needed

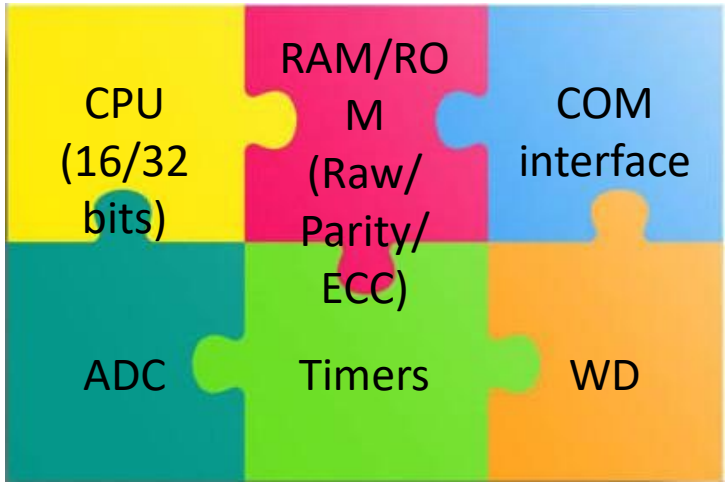
- Time to market decrease (～10 to 20% per year)

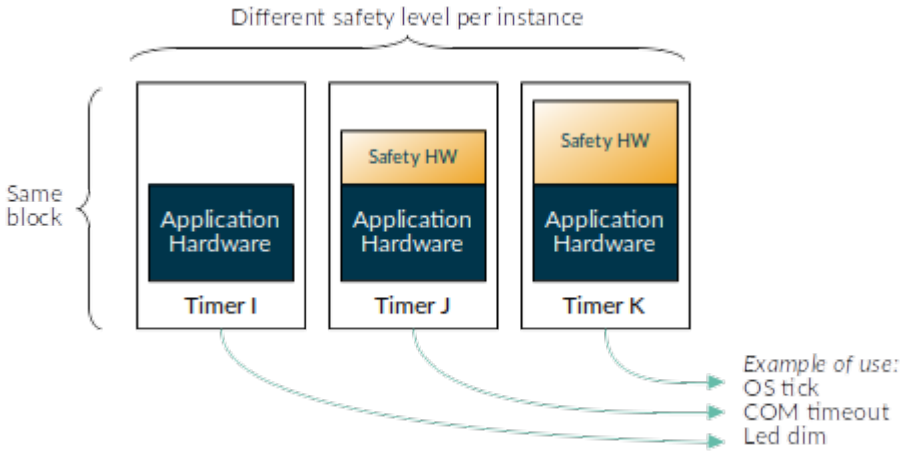


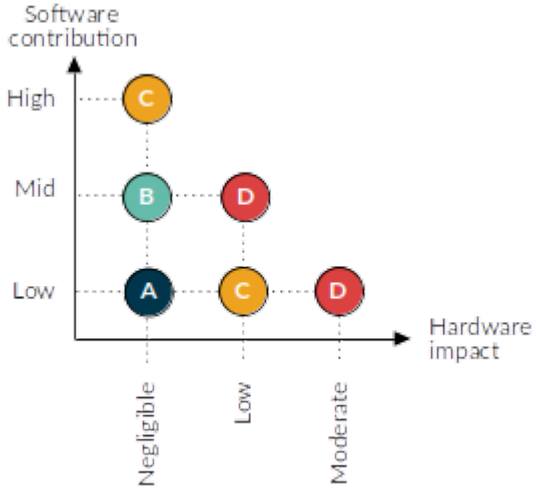
Houston, we have a problem!

Our solution: A Platform approach

- A ready to use set of highly configurable components
- Components are assembled to realize the specified functionality
- Each component is "*ASIL-D ready*"





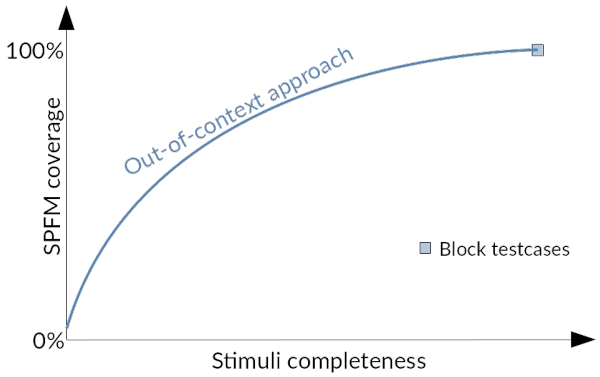


Platform Pros and Cons

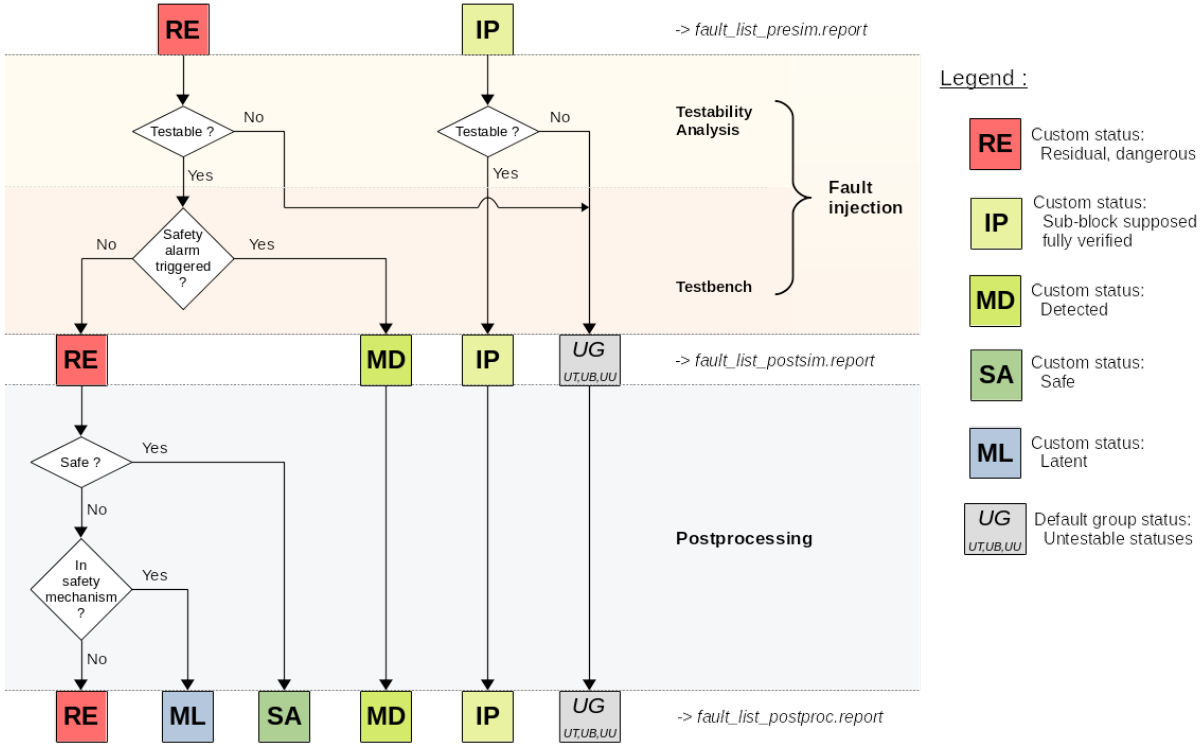
Constraints	Consequences	[mitigable] Threats
"A kind of" Bottom-up approach	Not very fashionable	
	Components must exist **upfront**	Marketing good enough?
	Components must be **highly configurable**	Development time overhead?
Components **not** designed for a specific project	Impose a **SEooC** methodology	Area overhead?
	Cannot rely on another component	
Technology and layout agnostic	Impose a **RTL fault injection**	Results representivity vs. gates?
RTL coding attention	From coding constraints up to coding obfuscation	

Method	Strengths	Notes/Threats
Fault injection per component	Results available [almost] at project start	Avoid late “bad surprises”
	Much faster than on the full circuit	seconds/minutes vs. hours/days
	Ease development of complex components	"Russian dolls"
RTL fault injection per component	Validate safety mechanism(s)	Gate coverage may be ≤ RTL coverage
SEooC	Project specific requirements give better results	e.g residual faults can become safe
Final gate level fault injection campaign	Only to confirm estimated values	Minor differences

Block verification



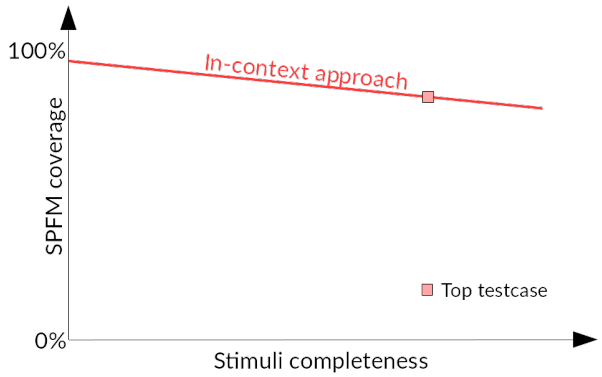
- Safety Element out of Context (SEooC)
- Configurable independant safety level
- Fault injection on each block:
  - On RTL
  - Documented safety hypotheses
  - SPFM target: 100%, LFM target: 90%
  - All faults injected
  - Out-of-context approach
- Reuse of shared verified sub-blocks
  (registers with parity, safe counters, etc...)

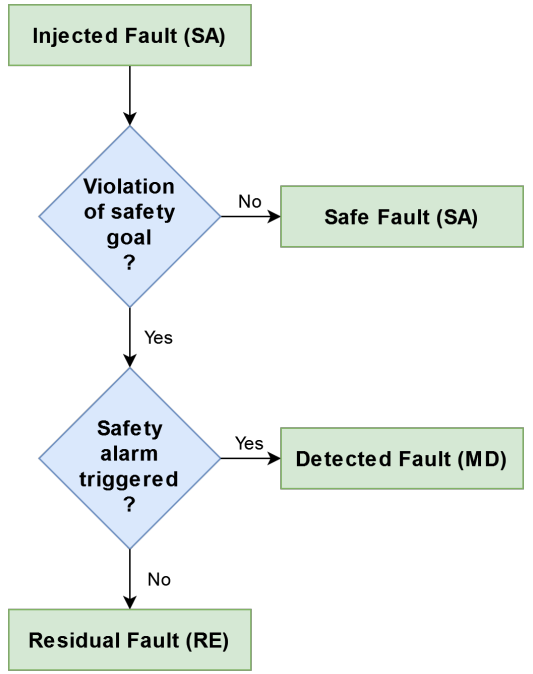


Out-of-context fault injection flow

Top verification

- Known context (technology, safety goal, safety margin, etc…)
- Configured safety level of blocks according to needs
- Fault injection on top:
  - On RTL -> to get early estimation
  - On Gates -> to confirm results
  - SPFM and LFM targets according to ASIL level
  - Statistical approach for injected fault sample
  - In-context approach of a typical application
- Reuse of platform blocks already checked
- Custom blocks development respects platform rules

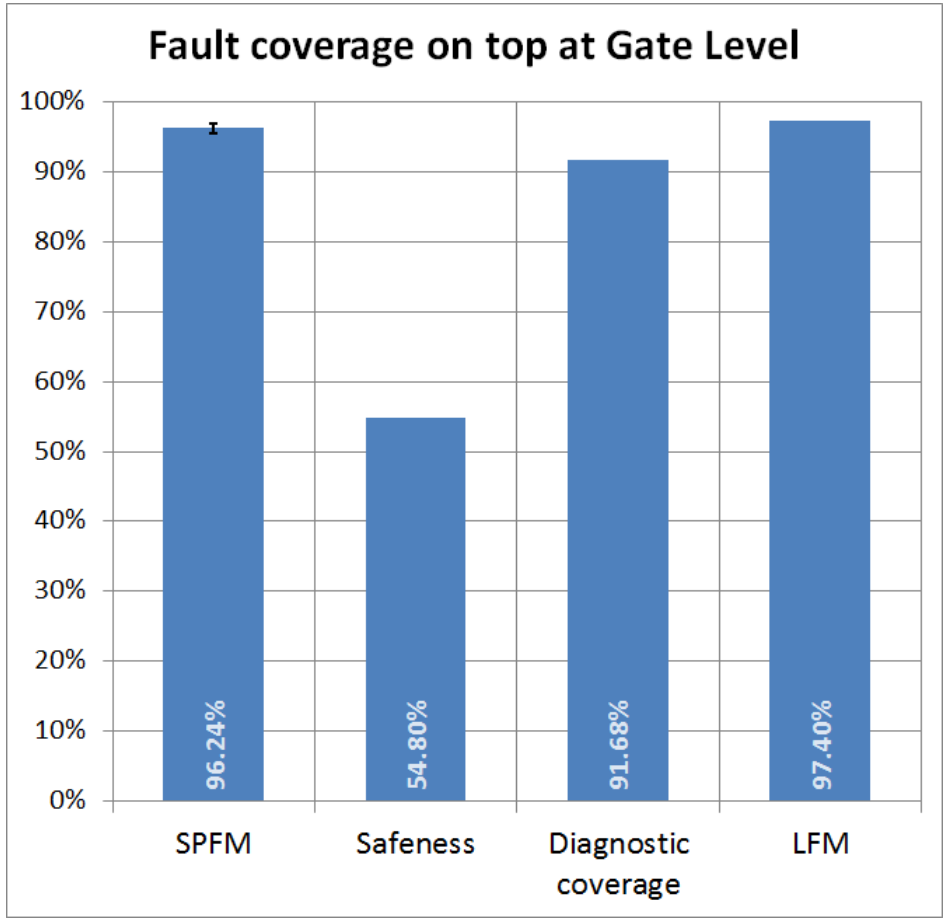




In-context fault injection flow

Metric customization

- SPFM can be split into:
  - **Safeness**:
    Proportion of safe faults (≈50%)
    -> Check that top testcase gives enough stimulus
  - **Diagnostic coverage**:
    Proportion of detected faults among unsafe faults



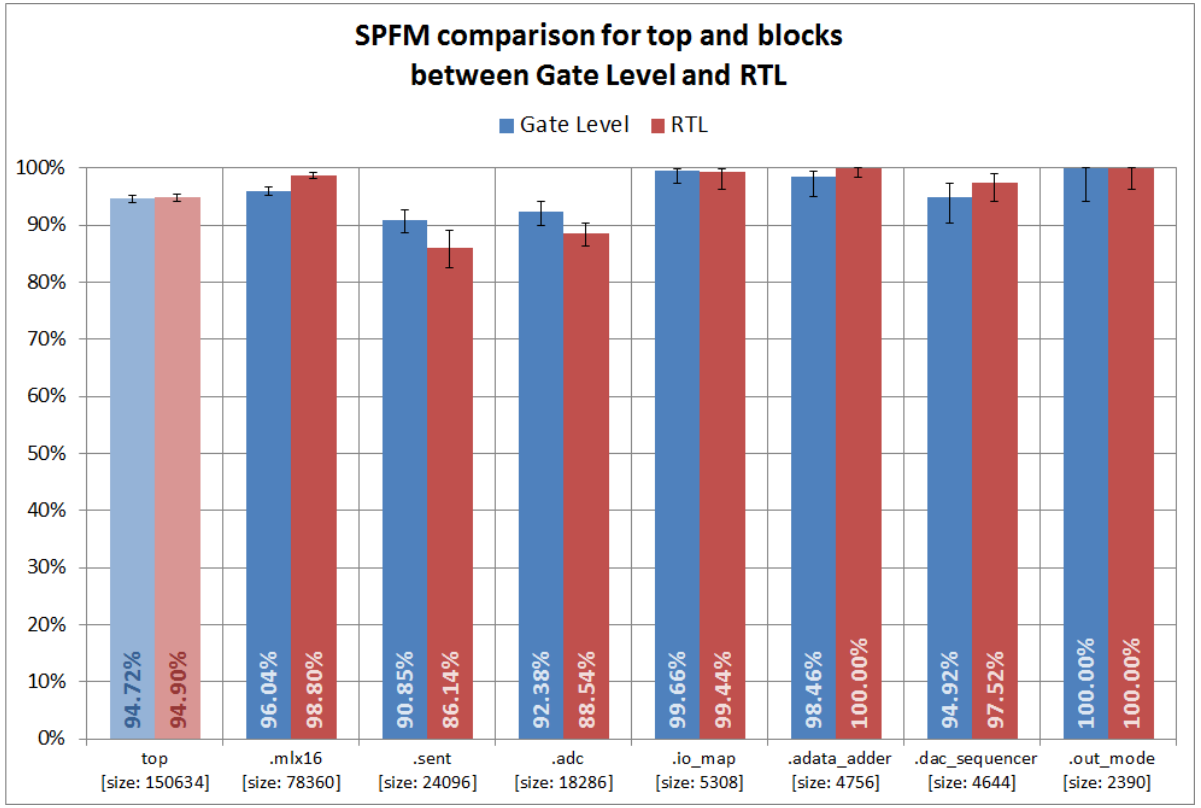
Fault injection on RTL and Gate Level

- For blocks:
  - RTL coverage ≈ Gate coverage ± 5%
- For top:
  - RTL coverage ≈ Gate coverage ± 1%

- Accurate RTL coverage thanks to:
  - Multiplicity and diversity of blocks

    -> Averaging tendency
  - Z01X optimizations (pruning, testability, etc…)

    -> Mimic synthesizer optimization



Summary

	Platform's block	Project's top
Fault population	RTL, all faults	RTL and Gate Level, sampled faults
Strategy	Improve safety mechanisms and add hypotheses until all faults are detected	Check metric targets in typical usecases
Flow	***Out-of-context check for full coverage*** (injected as Residual)	***In-context check for project target*** (injected as Safe)
Typical fault amount	100 - 5,000	10,000 - 30,000 for RTL 50,000 - 200,000 for Gate Level
Typical runtime	≈ 10 min	≈ few hours

2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

MUNICH, GERMANY
DECEMBER 6 - 7, 2022

Questions?

accellera SYSTEMS INITIATIVE

2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

MUNICH, GERMANY
DECEMBER 6 - 7, 2022

Mitigating Soft Error Impact on System Dependability

Ghani Kanawati, Technical Director of Functional Safety, Arm

accellera SYSTEMS INITIATIVE

Agenda

- Introduction/Problem Statement
- Identification of Critical Registers
- Automatic Insertion of Error Detection Codes with SSF
- Proposed Customer Flow with SSF
- Parity Insertion on Arm Design
- Summary

2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE
MUNICH, GERMANY
DECEMBER 6 - 7, 2022

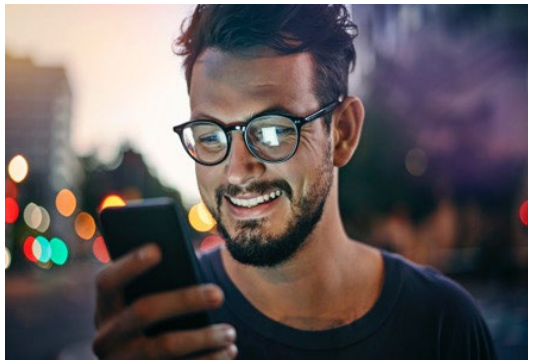
Introduction/Problem Statement

accellera SYSTEMS INITIATIVE

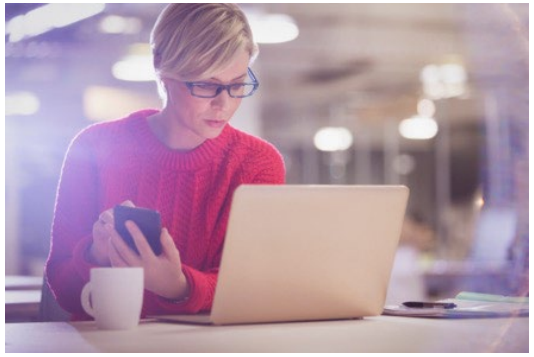
Arm IP is Ubiquitous Throughout Industry

Infrastructure

Mobile



Client



Wearables

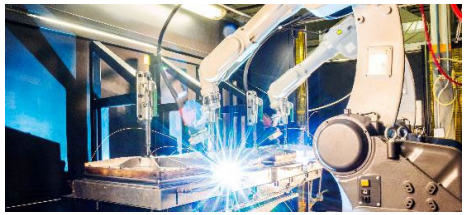
Market Segment Dependability Requirements

Different Segments have different Dependability Requirements

- Arm targets IP for many different markets
  - Each market can have its own standard for compliance
- Arm IP is developed to be configurable.
  - Same core, different use case
  - Heavy customisation and configuration would be necessary to deliver a compliant IP for use in automotive as well as Industrial application
- Not feasible for Arm to model all of these different use cases

Automotive
Autonomous driving

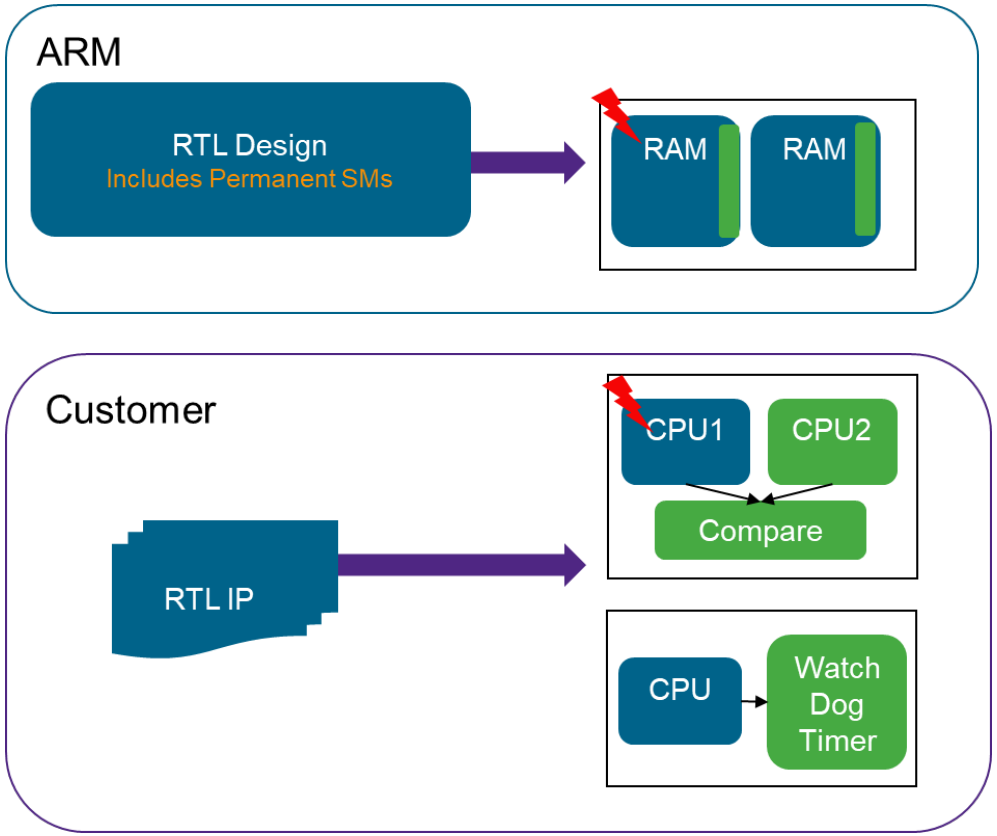
Industrial
Factory Automation



Consumer
Domestic Robots

Current Arm IP Release Flow

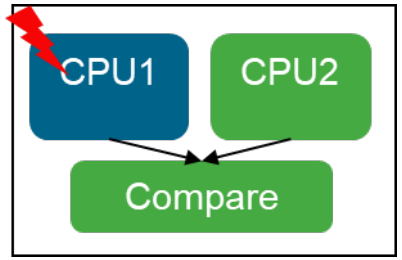
IP Protection for Dependability



- Arm will insert SMs at the IP RTL level
- Examples
  - ECC
  - Parity on large Register Files
- Arm Software Test Library (STL)
  - SW safety mechanisms targeting permanent faults
- Arm define additional SM requirements for customer insertion at SoC level (Assumptions of Use)
- Examples
  - DCLS
  - Watchdog timer/monitoring around IP
  - IP register hardening (SEU tolerant)/Parity
- Arm also highlight potential areas of weakness
  - Recommend further analysis by customer depending on use case

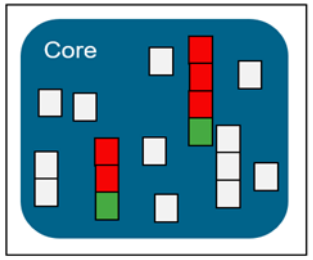
Assumption of Use (AoU) - Limitations

- AoUs are costly to implement at the customer SoC level
  - High Area Overhead
  - SMs not efficient



DCLS has High Area Overhead

- Register hardening/Parity insertion throughout IP
  - Customer may not accept 100% hardening due to PPA limitations
  - Customer will need to do further analysis to find critical registers



Register hardening throughout the core has limitations for PPA

2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

MUNICH, GERMANY
DECEMBER 6 - 7, 2022

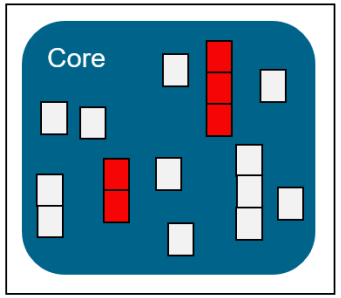
Identification of Critical Registers

accellera SYSTEMS INITIATIVE

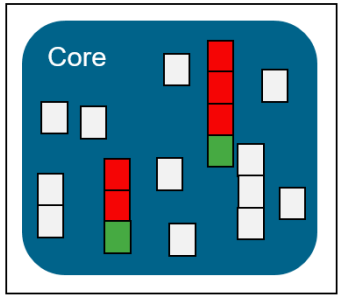
Identification of Critical Registers

Shift Left Solution

- Need a shift left to enable customers to identify critical registers in IP based upon unique use case



Use TestMAX FuSa to identify critical registers in Arm IP



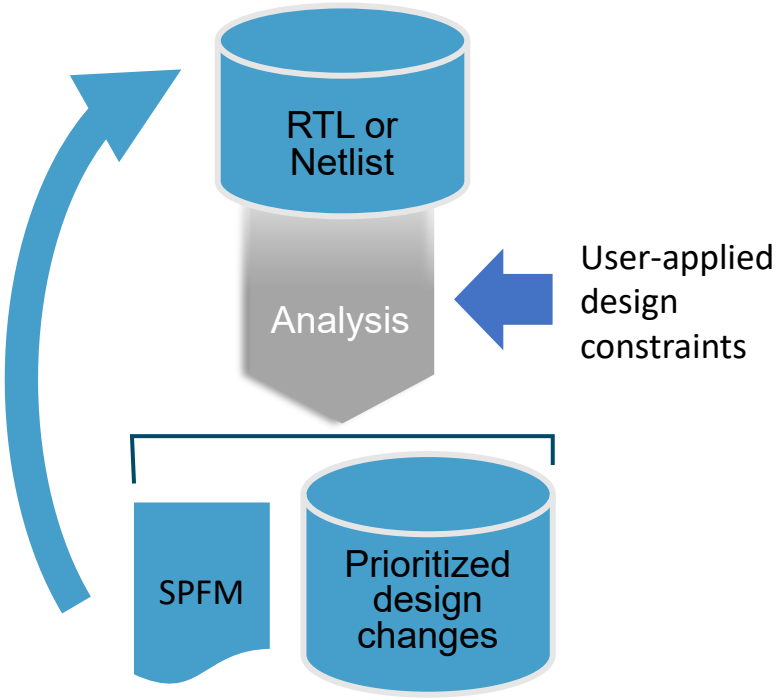
Use Fusion Compiler to insert parity into design

- Advantages
  - No costly SOC level solutions
  - Better PPA tailored to customer use case
  - Efficient SM insertion

Solutions Methodologies: TestMAX FuSa

Fast static analysis to drive design changes for FuSa & Reliability improvement

ISO 26262 Certified



- Fast early Dependability metrics calculation
  - Diagnostic Coverage (DC)
  - Single Point Fault Metric (SPFM)
  - Failure Mode Distribution (FMD)
- Shift-left – analysis performed at RTL and netlist
  - Report a priority list of registers with higher vulnerability to soft errors
- Vector-less – does not require testbench
  - Option available to run with vectors through FSDB input
- Scales to very large designs
  - Runs in hours on hundreds of millions of gates
  - Can be run at hierarchical level and not limited to block level

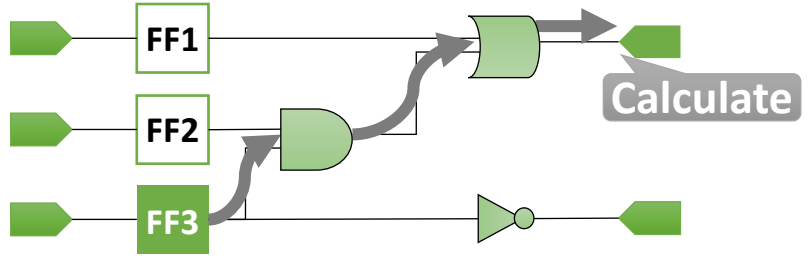
TestMAX FuSa static analysis

Fault Propagation based on probabilities

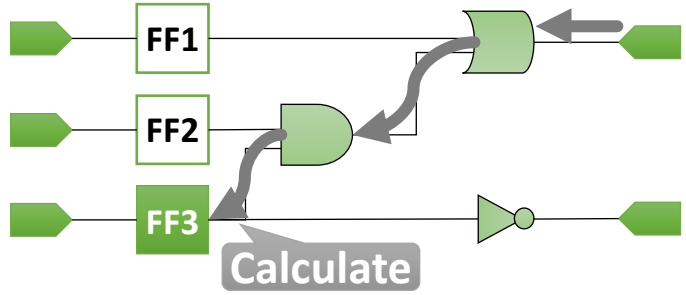
- TestMAX FuSa calculates controllability and observability probabilities of logic nodes in a design
- Does not require testbench stimuli
- Observation points can be specified at top-level port or hierarchical pin or net
  - Ability to black box modules protected via ECC
- **Ability to identify fault sensitive aspects of the hardware**
- **Report a priority list of registers with higher vulnerability to soft errors**

Calculation of Soft Error Failure

Static analysis: controllability calculation



Static analysis: observability calculation



TestMAX FuSa Calculates SPFM & Reports Register Contributions

```
################################################################################
# Format :
#     SPFMContribution of a register 'i' is calculated by using below formula.
#          SPFMContribution[i] = lambda_reg[i] x (1 - DC[i]) x (1 - probability_safe[i])
#                                SPFMContribution[i]
#          Contribution % = -------------------- x 100
#                            Sum(lambda_reg[i])
#
#     Note: Top N percent contributors are reported in this report if N percent
#           count is more than 500. At max 500 contributors are reported if N percent
#           count is less than 500
################################################################################

################################################################################
# Parameter values :
#     'ser_control_sequential_depth'       :    100
#     'ser_observe_sequential_depth'       :    100
#     'ser_observe_nsr_initial_value'      :    0.5
#     'ser_propagation_difference_threshold:    5
#     'ser_register_report_top_contributors:    1e+02
################################################################################


################################################################################
# Design information :
#     Top module : 'core_cascade'
################################
```

```
Index Safety SPFMContribution lambd


1     N      0.010000          0.010
2     N      0.010000          0.010
3     N      0.010000          0.010
4     N      0.010000          0.010
5     N      0.010000          0.010
6     N      0.010000          0.010
7     N      0.010000          0.010
8     N      0.010000          0.010
9     N      0.010000          0.010
10    N      0.010000          0.010
```

```
################################################################################
# Format :
#     SPFMContribution of a register 'i' is calculated by using below formula.
#          SPFMContribution[i] = lambda_reg[i] x (1 - DC[i]) x (1 - probability_safe[i])
#                                SPFMContribution[i]
#          Contribution % = -------------------- x 100
#                            Sum(lambda_reg[i])
#
#     Note: Top N percent contributors are reported in this report if N percent
#           count is more than 500.
#           count is less than 500
####################################
```

Prioritized List of Registers With Largest Contribution of Single Point Soft Error Failures

	Index	Safety	SPFMContribution	lambda_reg	DC	Probability	Contribution	% Cumulative-Contribution	Register
53	1	N	0.010000	0.010000	0.000000	1.000000	0.085034	0.085034	core_cascade.agu.XAB_DFF.\q_reg[0]
54	2	N	0.010000	0.010000	0.000000	1.000000	0.085034	0.170068	core_cascade.agu.XAB_DFF.\q_reg[1]
55	3	N	0.010000	0.010000	0.000000	1.000000	0.085034	0.255102	core_cascade.agu.XAB_DFF.\q_reg[2]
56	4	N	0.010000	0.010000	0.000000	1.000000	0.085034	0.340136	core_cascade.agu.XAB_DFF.\q_reg[3]
57	5	N	0.010000	0.010000	0.000000	1.000000	0.085034	0.425170	core_cascade.agu.XAB_DFF.\q_reg[4]
58	6	N	0.010000	0.010000	0.000000	1.000000	0.085034	0.510204	core_cascade.agu.XAB_DFF.\q_reg[5]
59	7	N	0.010000	0.010000	0.000000	1.000000	0.085034	0.595238	core_cascade.agu.XAB_DFF.\q_reg[6]
60	8	N	0.010000	0.010000	0.000000	1.000000	0.085034	0.680272	core_cascade.agu.XAB_DFF.\q_reg[7]

0.5% addition to SPFM

2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

MUNICH, GERMANY

DECEMBER 6 - 7, 2022

Automatic Insertion of Error Detection Codes with SSF

accellera SYSTEMS INITIATIVE

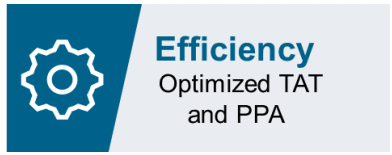
Synopsys Safety Specification Format (SSF)

- What is SSF?
  - Common commands supported by all relevant Dependability tools in digital implementation flow

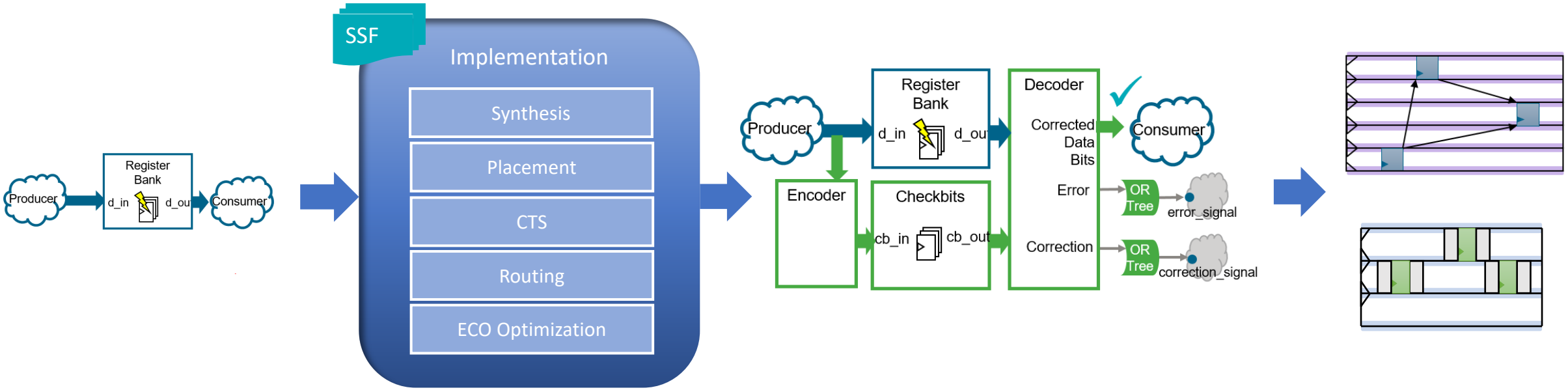
- Why was SSF created?
  - Capture Dependability intent and implementation at various stages of the design

- What does it look like?
  - Intent: define type of SM (*create_*_rule*) to protect certain element (*set_**)
  - Implementation: track elements of inserted SM (*mark_**)

- What value does SSF add?



Error Code (EC) Handling with SSF



SSF Support through all implementation stages/engines
- Maintains QOR

SSF Driven EC Synthesis

Type: ECC/EDC/Parity
Correction/Error Signal Synthesis

SSF Driven CMI Minimization

EC Register Separation
EC Isolation

Error Code Handling with SSF

create_safety_error_code_rule (SSF)

- A safety error code rule is an abstract object that captures information about how to handle ECs
  - ECs can be encoded with even/odd parity, EDC (Hamming2) or ECC (Hamming3)

```
create_safety_error_code_rule
    -name rule_name
    [-type <even_parity | odd_parity | ecc | edc >]
    [-slice_size <num_bits>]
    [-distance <dist>]
    [-isolation]
```

Physical handling

Error Code Handling with SSF

set_ safety_error_code_rule and mark_ safety_error_code (SSF)

- The set_safety_error_code_rule applies the error code rule to a register bank or bus fabric in the design
  - It sets intent for simulation and insertion
- The mark_safety_error_code command identifies existing ECs in the design (RTL or netlist)
  - The command will be auto created during physical implementation from the applied intent

```
set_safety_error_code_rule

[-requirement_id <string>]
-rule <rule_name>
-data <signal to be encoded>

[-error_signal <pin_port>]
[-correction_signal <pin_or_port>]
```

```
mark_safety_error_code
    -name group_name
    [-requirement_id <string>]
    -rule <rule_name>
    -data <encoded signals>
    [-checkbits <generated checkbits>]
    [-error_signal <pin_port>]
    [-correction_signal <pin_or_port>]
```

2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

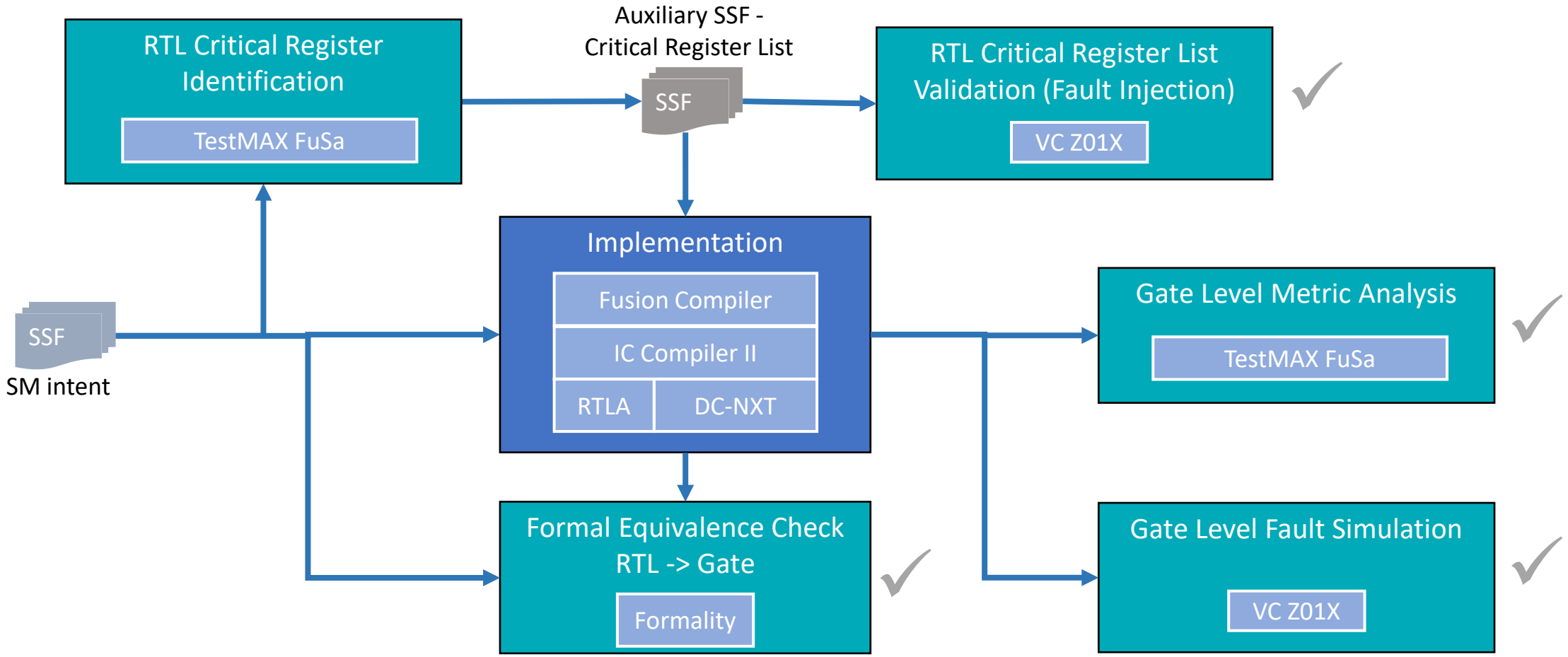
MUNICH, GERMANY

DECEMBER 6 - 7, 2022

Proposed Customer Flow with SSF

accellera SYSTEMS INITIATIVE

Proposed Customer Flow with SSF



2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

MUNICH, GERMANY

DECEMBER 6 - 7, 2022

Parity Insertion on Arm Design

accellera SYSTEMS INITIATIVE

Parity insertion on Arm design using SSF

hntelp_vector Block

- hntelp_vector, large block from hunter_elp design
  - 2.5M Instances
  - tsmc cln05

- Requirement - to insert parity on all registers in u_ct hierarchical block
  - 150,757 standard cells
  - 14,507 registers

Parity insertion on Arm design using SSF

Safety Specification Format (SSF)

- SSF File
  - Single input file to Fusion Compiler to drive FuSa Intent
- Parity Insertion Format

Odd Parity

Slice size of 8 bits

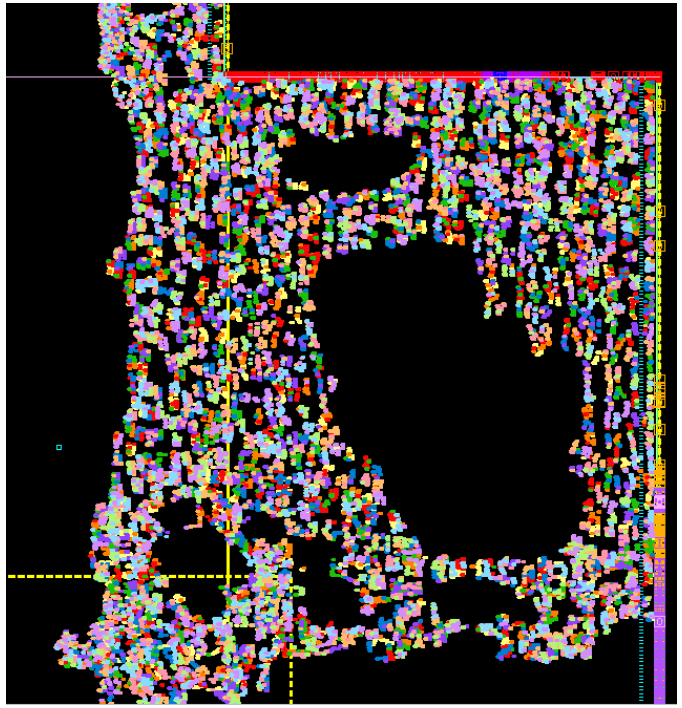
Applied to registers in hierarchical block u_ct

Error signals to top level port

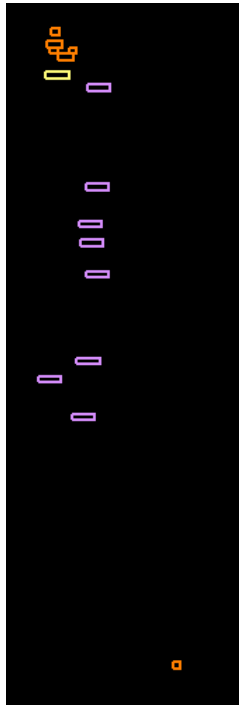
```
ssf_version 1.1

create_safety_error_code_rule -name
error_rule1 \
-type odd_parity \
-sequential \
-slice_size 8

set pin [get_objects_for_safety -pattern
u_ct/*reg*/Q -object_type pin]

set_safety_error_code_rule \
  -rule error_rule1 \
  -data $pin \
  -error_signal sec_err
```

Parity insertion on Arm design using SSF

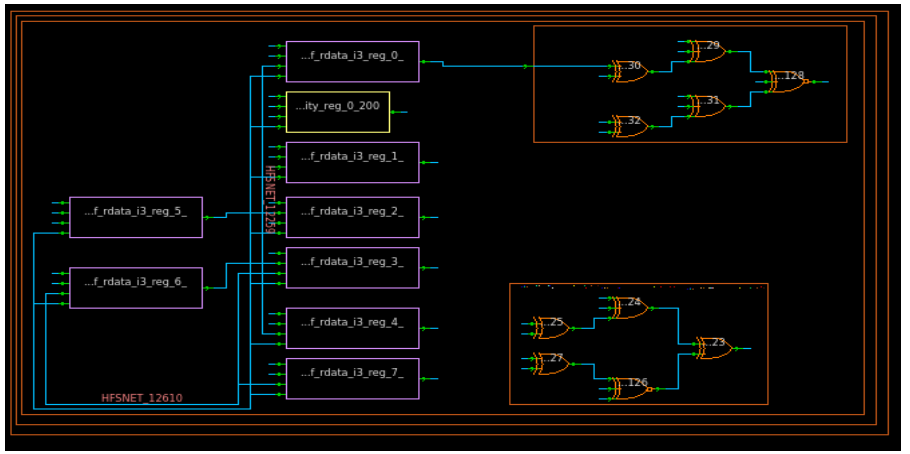
Error Code Placement and Logical Representation



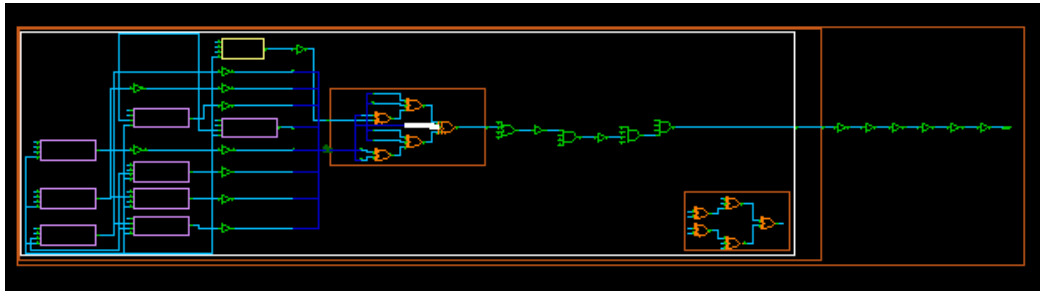
EC placement (~3000 groups)



Single group placement



Single group schematic



Including OR tree

Parity insertion on Arm design using SSF

Dependability Report

- Dependability report performs checks on all inserted safety mechanisms
- > report_safety_status

Statistics - passed/failed

type	category	total	passed	failed
SR	safety register rules	0	0	0
SR	safety register groups	0	0	0
SC	safety core rules	0	0	0
SC	safety core groups	0	0	0
SC	safety cores	0	0	0
FSM	failsafe FSM rules	0	0	0
FSM	failsafe FSM groups	0	0	0
SEC	safety error code rules	1	1	0
SEC	safety error code groups	2966	2966	0

Parity insertion on Arm design using SSF

Flow Results

- Fusion Compiler baseline and parity flow QOR results

	WNS	TNS	NVE	Util	StdCelllArea	StdCells	Total, uW	Leakage, uW	DRCs	Shorts
Baseline	-0.0083	-0.0271	8	0.6184	278062	2262906	5.01E+04	4.90E+04	356	1
Parity	-0.0111	-0.0726	37	0.6228	280037	2299896	5.55E+04	5.45E+04	419	1

- EC Group metrics
  - Registers in u_ct hierarchy (pre-insertion): 14,507
  - EC Groups Inserted: 2,966
  - Data + parity registers in EC Groups: 17,473

- EC Group bit slice distribution

  Group Count with 1 bits: 704

  Group Count with 2 bits: 164

  Group Count with 3 bits: 19

  Group Count with 4 bits: 546

  Group Count with 5 bits: 71

  Group Count with 6 bits: 401

  Group Count with 7 bits: 15

  Group Count with 8 bits: 10,046

2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

MUNICH, GERMANY
DECEMBER 6 - 7, 2022

Summary

accellera SYSTEMS INITIATIVE

Summary

- Arm dependability flows require partnership with customers in handling soft error protection
  - TestMAX FuSa can be used at the RTL phase to identify critical registers and pass them on to Implementation for conversion to Error Codes during Synthesis
- Arm Design
  - Native Error Code insertion during synthesis was demonstrated on Arm design using SSF intent
  - Error Code density within the hierarchy was as expected
  - Neutrality in QOR compared to baseline was demonstrated
- Next Collaboration Steps
  - Identify critical registers using TestMAX FuSa and pass to implementation via SSF
  - RTL to gate Formal Equivalence Check using Error Code virtualization in reference
  - Integration into Arm customer reference flows

2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

MUNICH, GERMANY
DECEMBER 6 - 7, 2022

Questions?

accellera SYSTEMS INITIATIVE