Complex Safety Mechanisms Require Interoperability and Automation For Validation And Metric Closure

Daeseo Cha
Principal Engineer
SAMSUNG

Vedant Garg
Principal Architect
SIEMENS

accellera SYSTEMS INITIATIVE
2023 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Agenda

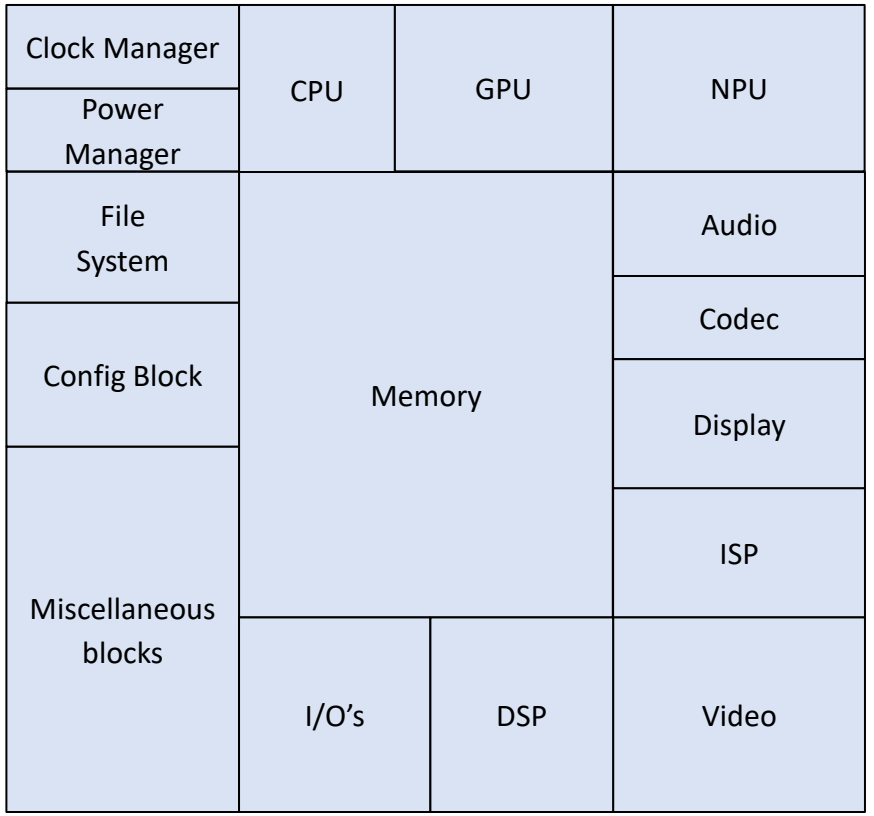
- Problem Statement
- Proposed Solution
  - Safety Analysis
  - Safety Verification
- Experiments of Automotive SOC
- Conclusion

Problem Statement (1/2)

- Increasing sensitivity to random hardware failures as semiconductor technologies are moving towards higher densities and lower operating voltages
- Modern cars deploying ADAS and AV features rely on these digital and analog systems to perform critical real-time applications
- This reliance has led to a concern over validation of these systems, and the question: are they safe?

Problem Statement (2/2)

- Pre RTL, RTL analysis
- SM choice /data
- RTL & GLS fault injection
- Structural proofing & checks
- Safety Metrics



Typical SOC representation

Proposed Solution

Safety Analysis — Fault List

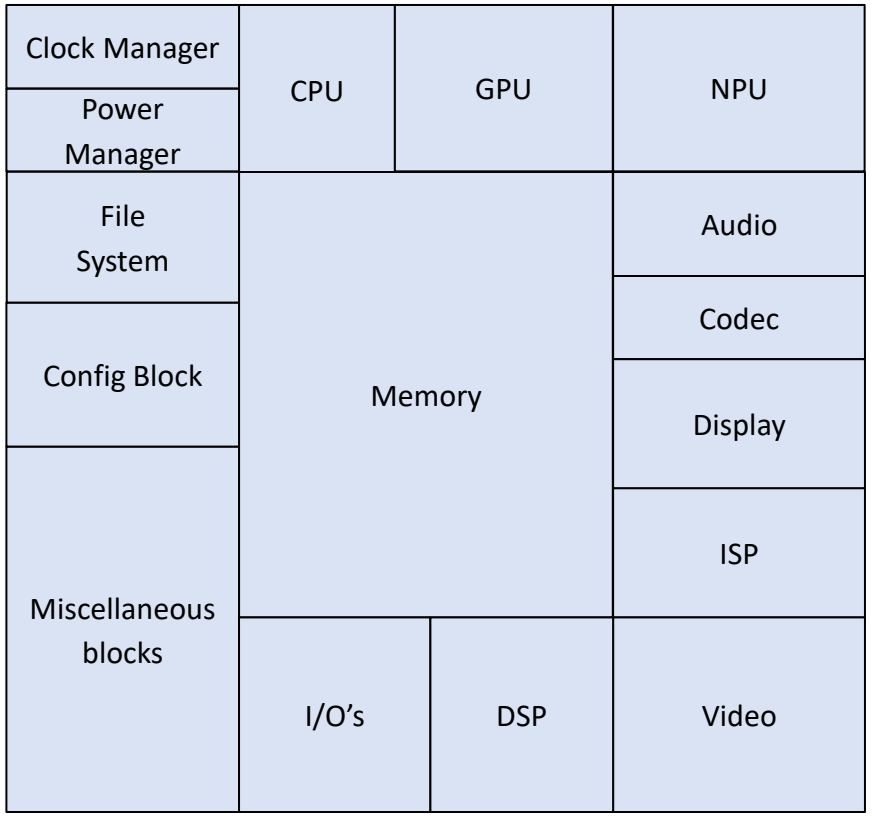
- Pre RTL, RTL analysis
- What-if exploration and Gap analysis
- Assessment of Existing SM
- Accurate Early Metric Estimation

Safety Verification — Fault Injection

- RTL & GLS fault injection
- Structural proofing & checks
- Reduced Iteration

Safety Metrics — Final Report

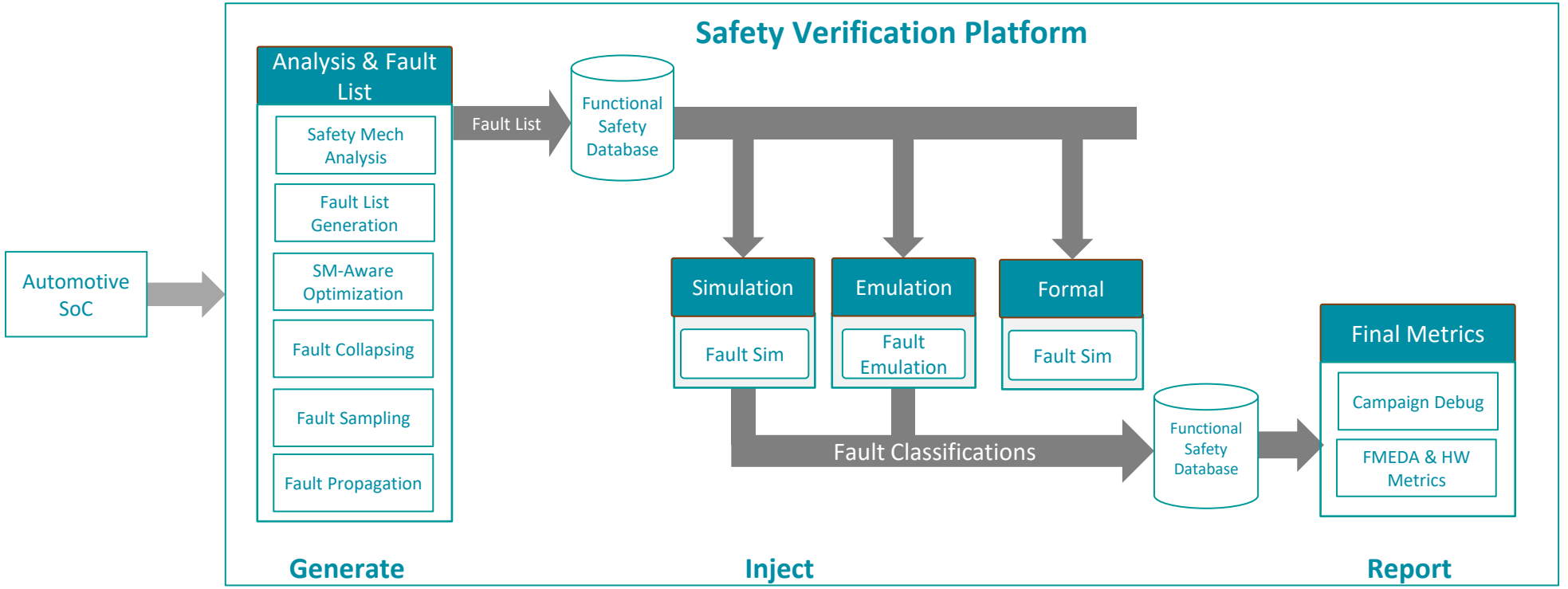
- Safety Metrics
- Final Automated FMEDA



Typical SOC representation

Safety Workflow

- Optimizes functional verification methodology
- Automates fault injection using formal, simulation and emulation
- Annotates results with common database

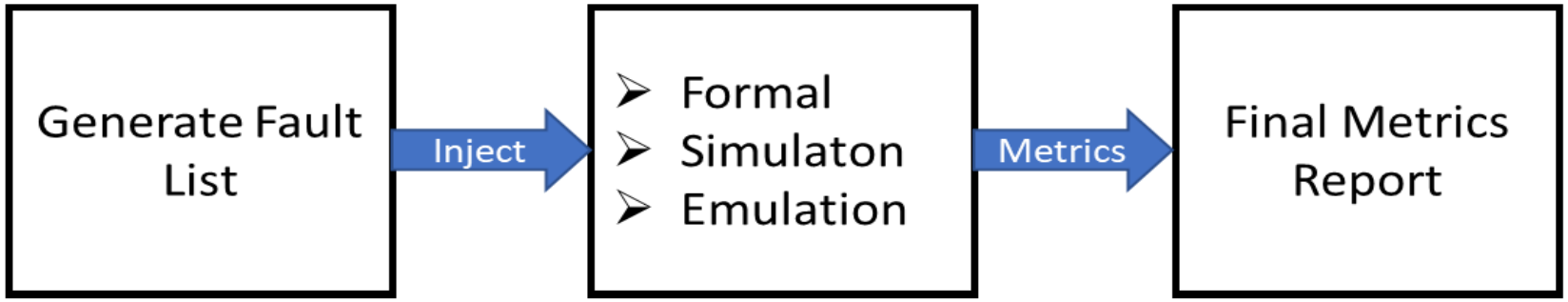


Optimizing the Safety Workflow

- Early RTL analysis (structural)
- Fault lists can be further optimized for the specific engine used for fault injection
- Fault list optimization
  - Safety mechanism aware faults
  - Fault collapsing
  - Identifying faults that won’t propagate
  - Statistical random sampling
  - Architecture vulnerability factors
- Formal techniques determining the testability of faults

Automated Fault Injection Flow

- ISO26262 Chapter 11 explains fault classification at the semiconductor
- Three-step flow
  - Step-1: Generated optimized fault lists
  - Step-2: Fault injection and classification
  - Step-3: Generating the metrics report



Safety Analysis : SafetyScope

- Pre RTL, RTL analysis
- What-if exploration and Gap analysis
- Assessment of Existing SM
- Accurate Early Metric Estimation
- FMEDA Generation

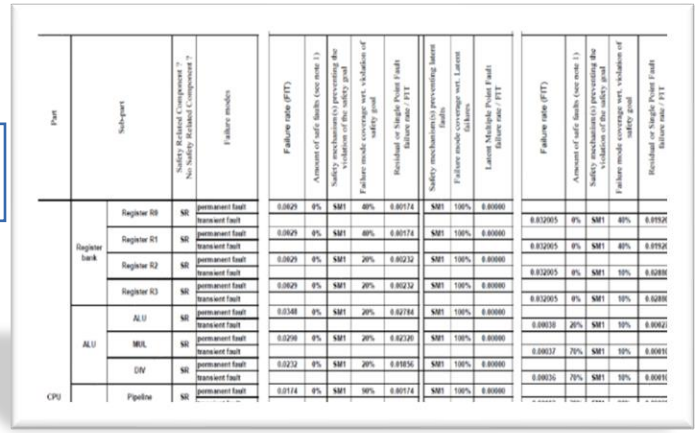
Early Exploration / FMEDA

- Pre RTL and Architectural
- Exploration of SM
- Modelling different FM

Fault List Generation

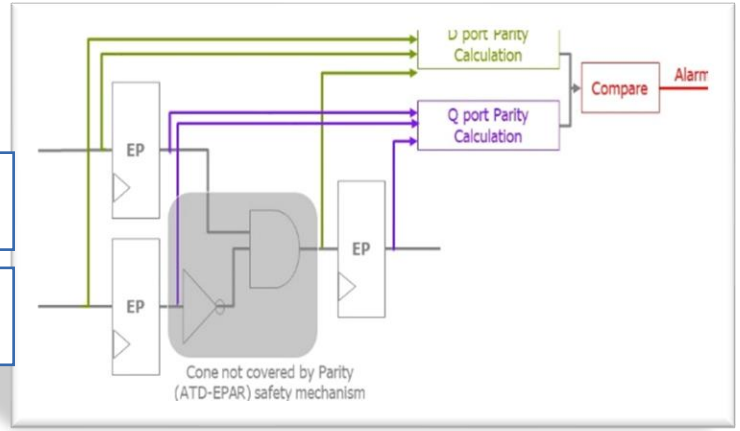
- Tops Down Methodology
- Integration to Validation tools
- Fault List generation

Automated FMEDA



Safety Explore

Targeted Fault List



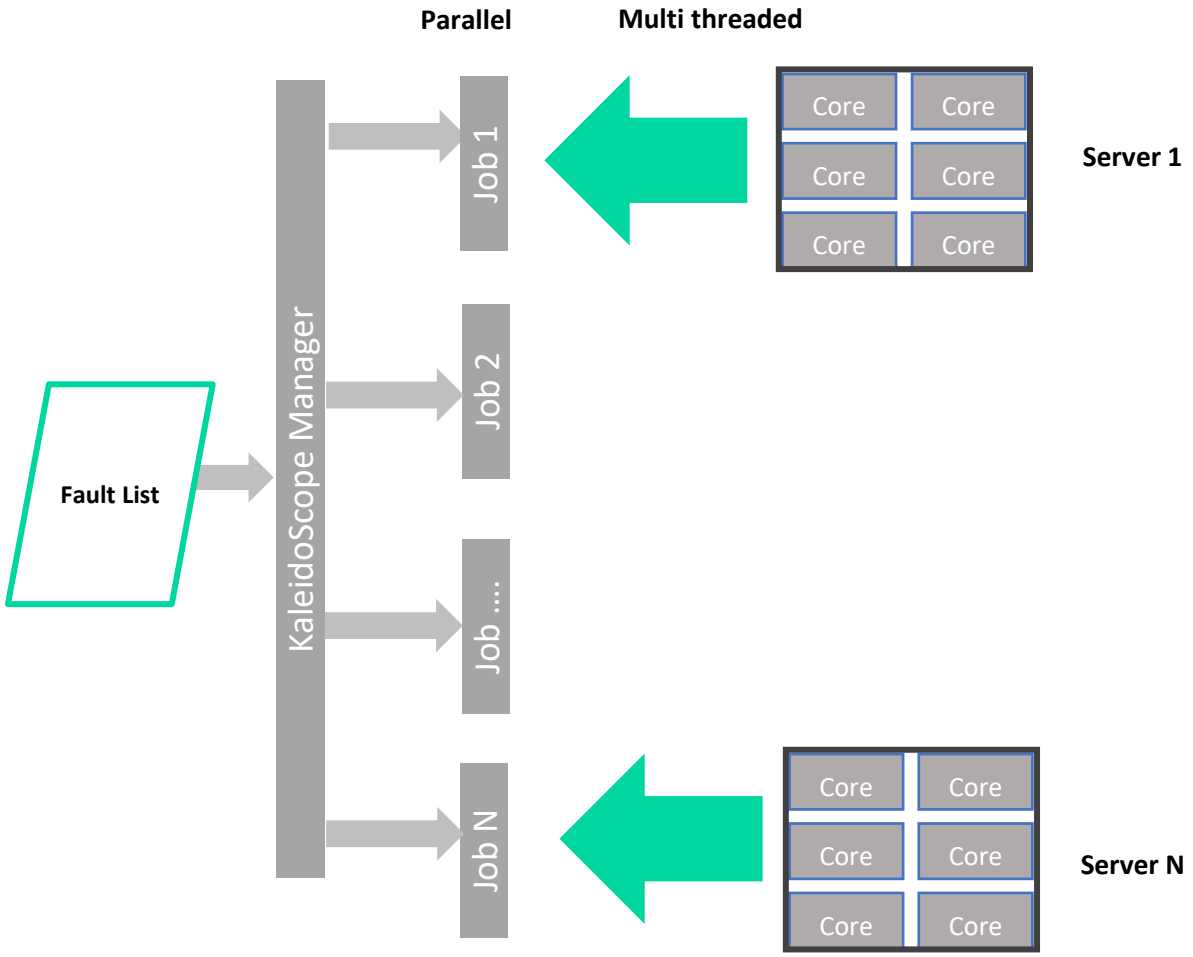
Safety Verification : KaleidoScope Manager

High Performance
- Concurrent & Distributed
- RTL and Gate Level
- Intelligent Injection

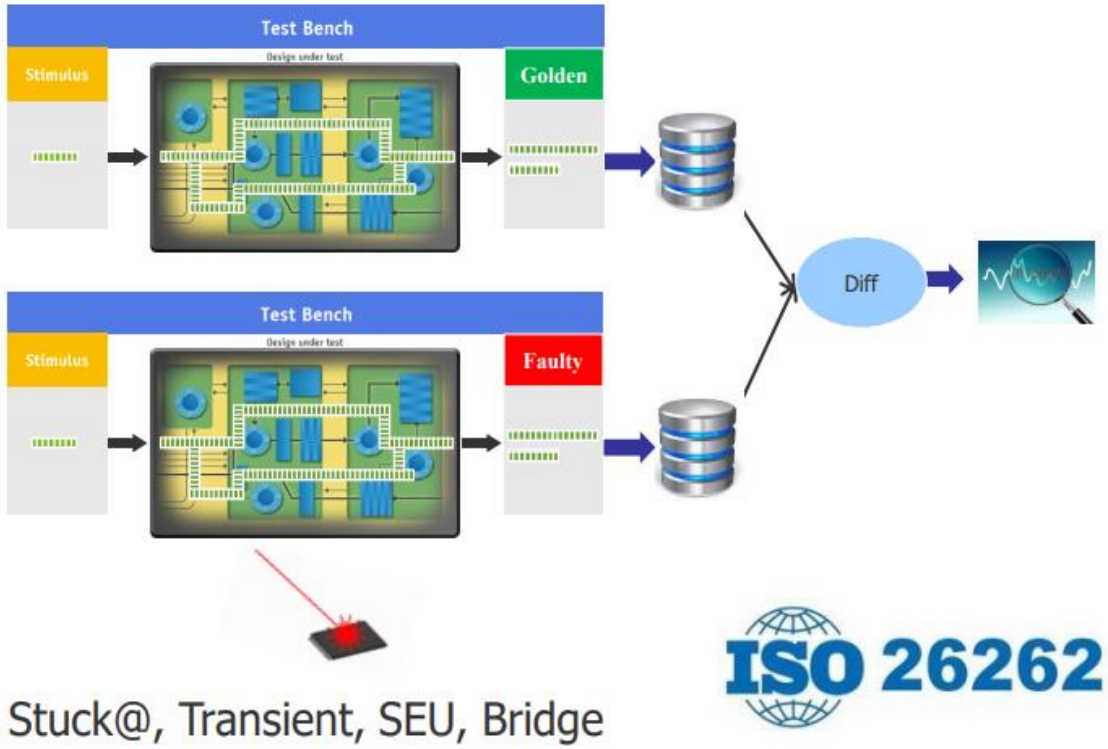
Optimized Simulation
- Stimulus Grading
- Statistical Random Sampling
- Adv. Fault Models

Debug
- Fault Propagation support
- Wave Compare

Fault Management
- Optimized Fault List
- Fault Classification



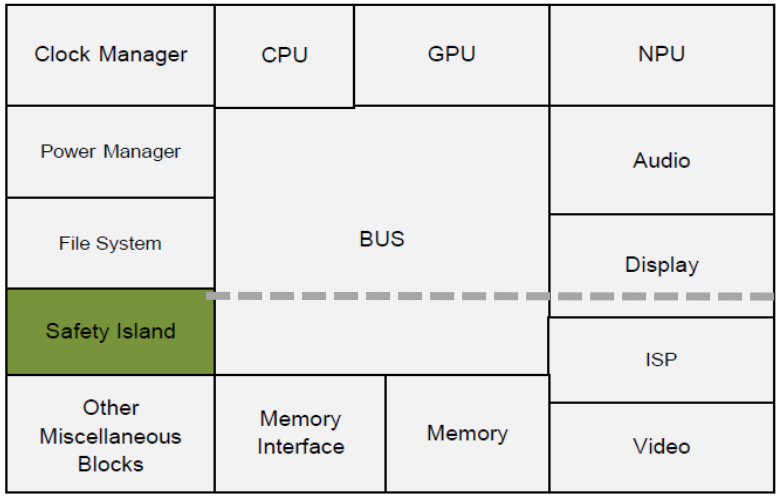
Safety Verification : Veloce Fault App



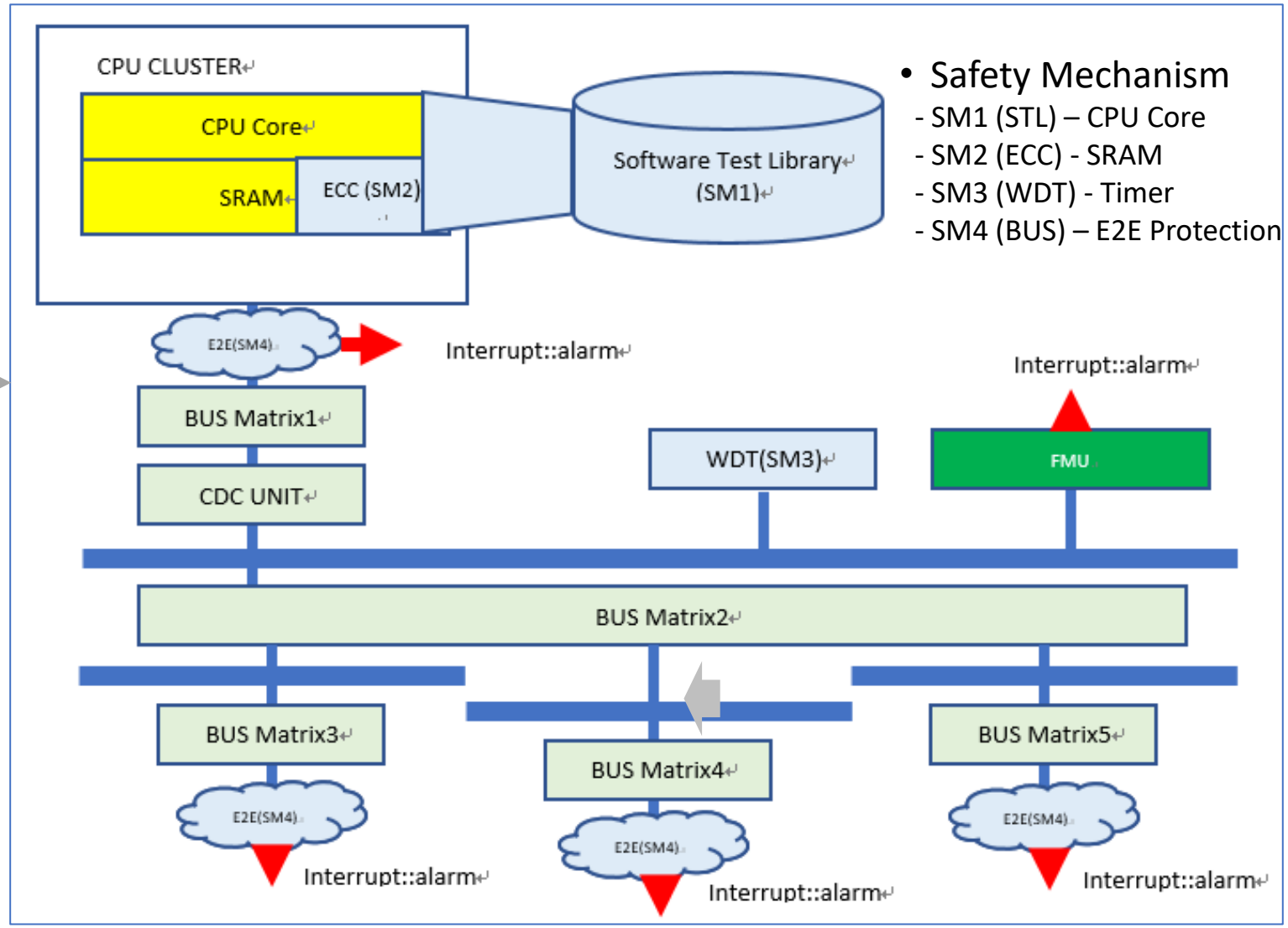
- Perform Mission Mode Safety circuit verification
- Analyze effectiveness of safety mechanisms in the design
- Mimic the effects of soft and hard faults on the design
- Targeting safety critical industries (automotive, aerospace, military)

Experiments of Automotive SOC

Safety Island



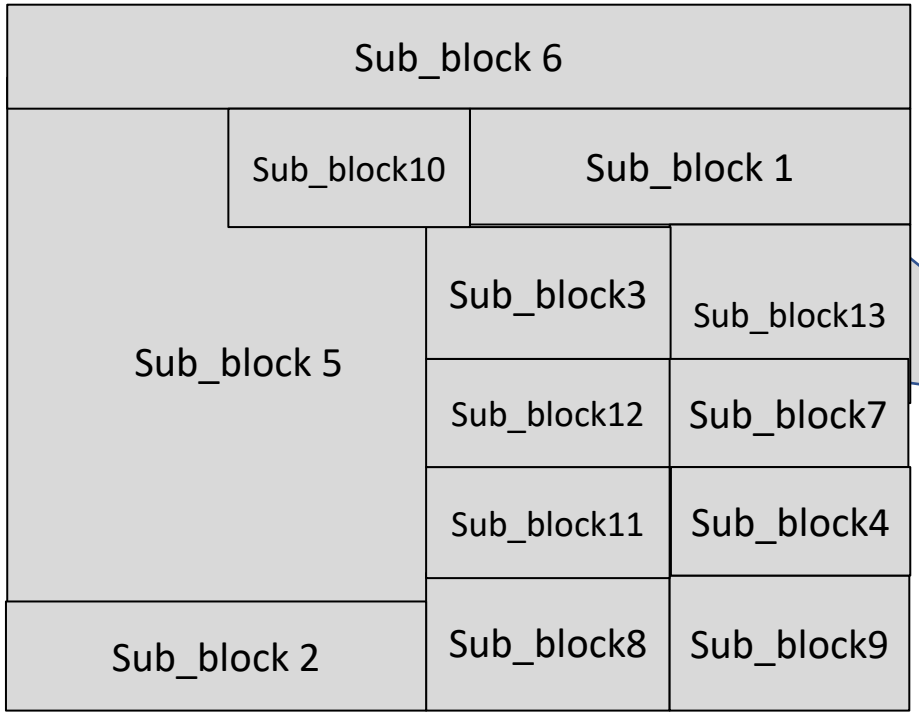
Block	Register(K)	Gates(M)
Safety Island	890	46.5
CPU/Cache	215	7.1
Bus	470	-
FMU	33	-

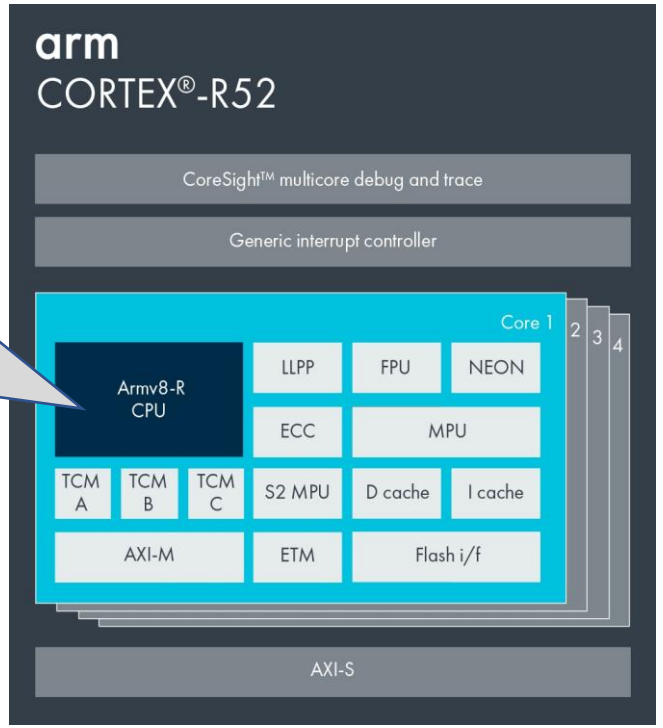


- Safety Mechanism
  - SM1 (STL) – CPU Core
  - SM2 (ECC) - SRAM
  - SM3 (WDT) - Timer
  - SM4 (BUS) – E2E Protection

CPU

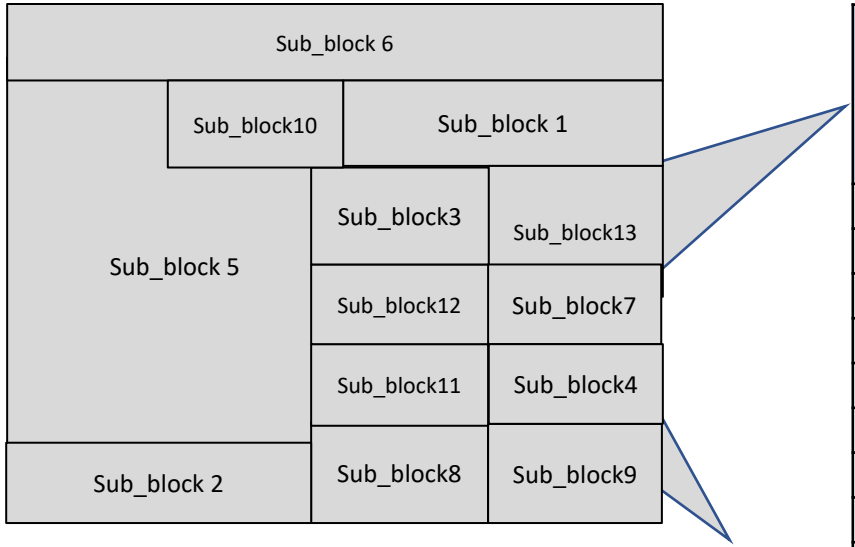
- ASIL-B
- Consisted of 13 blocks
- SM
  - Software Test Library
  - ECC
  - WDT
- Fault List Generation
  - Safetyscope
- Fault Injection
  - Veloce Fault App





(*Source: https://developer.arm.com/Processors/Cortex-R52)

CPU: Fault Distribution



Safety Mechanisms	Detected Fault Distribution (%)
ECC Correctable (SM 2)	16.86
ECC UnCorrectable (SM 2)	0.74
STL (SM 1)	80.64
WDT (SM 3)	1.76
Sub total	100.00

CPU Core + Cache	Total Faults (SA0 + SA1)	Total Fault Distribution (%)	Sampled faults (SA0 + SA1)	Sampled Fault Distribution (%)	Detected fault	Detected fault Ratio (%)
sub_block1	85,860	13.50	604	12.63	315	52.15
sub_block2	4,738	0.74	40	0.84	0	0.00
sub_block3	46,194	7.26	332	6.94	162	48.80
sub_block4	22,828	3.59	202	4.22	135	66.83
sub_block5	315,982	49.68	2,350	49.14	1,719	73.15
sub_block6	6,664	1.05	48	1.00	0	0.00
sub_block7	19,202	3.02	144	3.01	97	67.36
sub_block8	8,084	1.27	58	1.21	17	29.31
sub_block9	8,100	1.27	66	1.38	39	59.09
sub_block10	49,292	7.75	426	8.91	305	71.60
sub_block11	9,306	1.46	46	0.96	31	67.39
sub_block12	34,832	5.48	286	5.98	218	76.22
sub_block13	24,964	3.92	180	3.76	87	48.33
Total	636,046	100	4,782	100.00	3,125	65.35

Result: Fault classification

- Detected fault ratio of 65.35% with Margin of Error (MOE), ±1.19% using Veloce
- Performed stimulus grading to identify “Not Injected” faults

Fault Classification:	Total Faults (SA0 + SA1)	Total Fault Distribution (%)
Detected Observed	3,125	65.35
Detected Unobserved	0	0.00
Undetected Observed	79	1.65
Undetected Unobserved	616	12.88
Safe Fault -Dead Logic	234	4.89
Not Injected	728	15.22
Sub Total	4,782	100.00

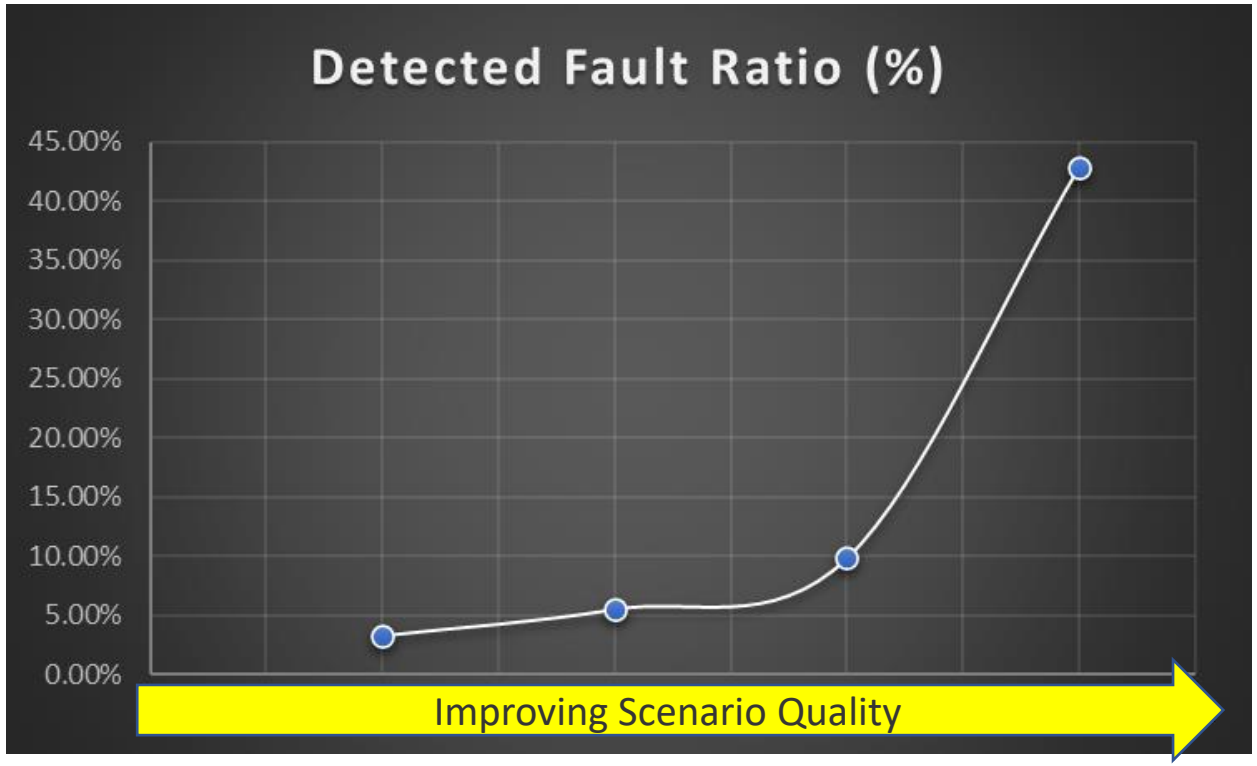
Result: Post Analysis

- Analysis of UU faults
  - COI Analysis
  - Signal Back Propagation
  - Safe Fault Configuration
  - Justification

Final Result	Faults (SA0 + SA1)	Fault Distribution(%)
Detected Observed	3125	77.18
Detected Unobserved	0	0.00
Undetected Observed (Residual) 10 fault moved to SAFE Fault	69	1.70
Undetected Unobserved (conservatively Residual)	280	6.92
Safe Fault - Dead Logic	234	5.78
Safe Fault Formal COI analysis Signal Back Propagation Safe Fault Configuration (Engineer's justification)	341	8.42
Sub Total 4782 – 733 (Not Injected)	4049	100.00

Result: BUS

- Fault List Generation
  - SafetyScope
- Fault Injection
  - Kaleidoscope
    (Concurrent engine)



BUS Related Logic	Total Faults (SA0 + SA1)	Scenario 1: Detected	Scenario 2: Detected	Scenario 3: Detected	Scenario 4: Detected	Scenario 5: Detected
Fault (%)	100%	3.27%	5.50%	9.80%	42.81%	(TBD)

Result: Final Metrics

- Achieved SPFM over 90% on CPU core using emulation

CPU + Cache mem	Final SPFM (%)
DCPerm	91.3805878
MOE	±1.293%

- Fault injection on BUS is in progress using simulation, FMU will be applied by formal
- Proposed three-step flow consisted of 1) faults optimization, 2) combining injection engine with automation, 3) report with common database successfully worked on our SOC

Conclusion

1. A comprehensive approach to designing and verifying a safe architecture may not be practical for large and complex systems with multiple safety mechanisms.

2. By using a system-on-chip level test case, we have shown how combining fault injection engines, optimization techniques, and automation can significantly reduce the overall time needed to complete safety analysis and certification.

3. Collaboration between product teams, methodology teams, and EDA vendors is crucial as tools, methods, and techniques are constantly evolving. This project utilized advanced methodologies such as safety analysis for optimization and fault pruning, concurrent fault simulation, fault emulation, and formal-based analysis to validate the safety requirements for the automotive system-on-chip.

4. Conducting safety analysis before running fault injection tests is essential and saves time.

5. Therefore, as demonstrated in this paper, the ability to use multiple engines and access results from a shared FuSa database is crucial for a project of this scale.

Questions

2023 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

SAN JOSE, CA, USA
FEBRUARY 27-MARCH 2, 2023

Thank You

Daeseo Cha
Principal Engineer
SAMSUNG

Vedant Garg
Principal Architect
SIEMENS

accellera SYSTEMS INITIATIVE