2023
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SAN JOSE, CA, USA
FEBRUARY 27-MARCH 2, 2023

Early Detection of Functional Corner Case Bugs using Methodologies of the ISO 26262

Moonki Jang, Samsung Electronics

SAMSUNG

accellera
SYSTEMS INITIATIVE

Agenda

- Introduction of ISO 26262
- Systematic Failure Analysis
- Systematic failure model generation using Machine Learning
- SFA for requirements-driven verification
- Conclusion

Requirements of Functional Safety

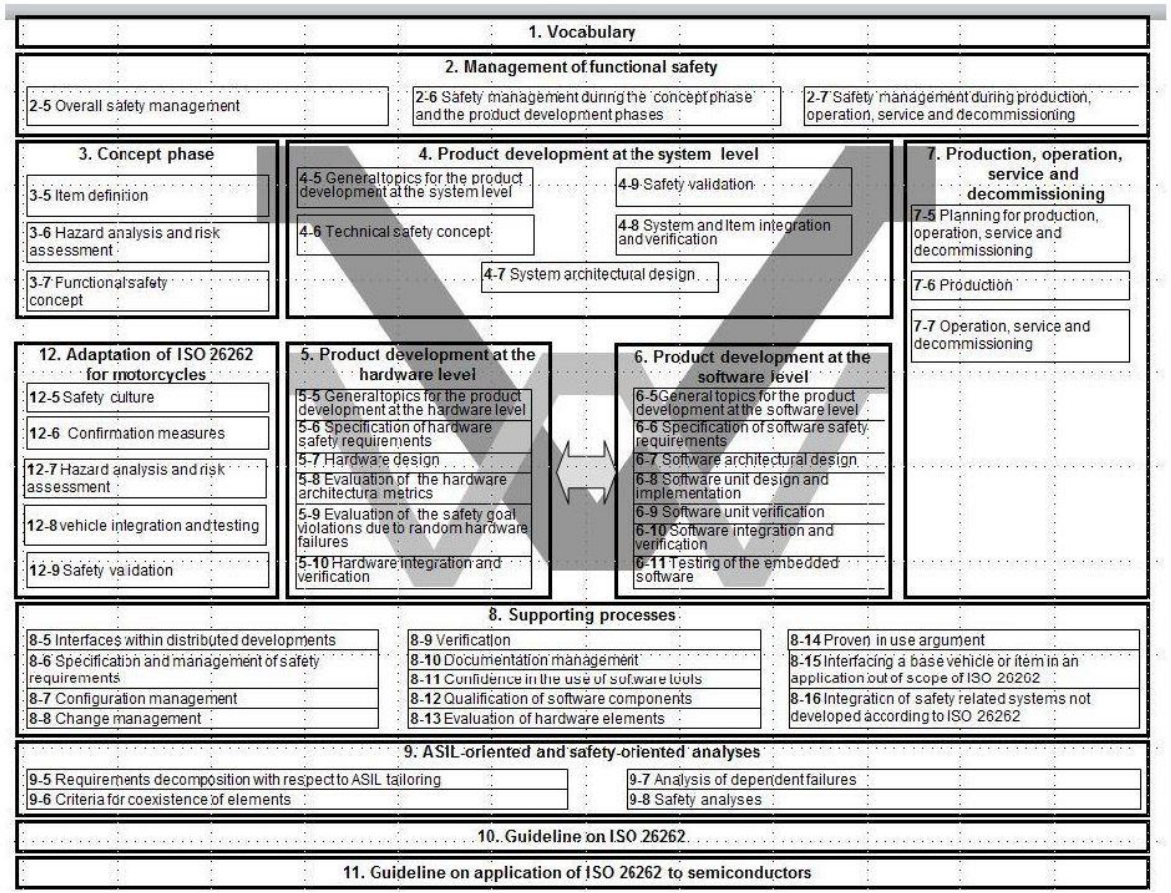
- For a long time, electronics were a comfort feature
  - Now, they are a safety feature

What is ISO 26262?

- Functional Safety standard for Road vehicles
  - Aims to address possible hazards caused by the malfunctioning behaviour of electronic and electrical systems in vehicles.
  - The first edition was published on 11 November 2011.
  - The second edition, published in December 2018, added ‘Part 11. Guidelines on application of ISO 26262 to semiconductors’
- Based on IEC 61508 : Functional Safety of Electrical / Electronic / Programable Electronic Safety-related system
  - Both IEC 61508 and ISO 26262 are risk-based safety standard

Overall framework of ISO 26262

- ISO 26262 V diagram



Foundations of Functional Safety

- Functional Safety
  - Avoidance of Systematic Faults
  - Control of Systematic Faults
  - Control of Random Hardware Faults

Random Hardware Failures

- Random Hardware Failure
  - Failure that can occur unpredictably during the lifetime of a hardware element and that follows a probability distribution
- Measures against failures
  - Required runtime safety mechanisms (self-tests, diagonostic coverage)
  - Redundancy, safety layer
  - SPFM (Single Point Fault Metric) : shows robustness of the item to single-point faults
    - Single Point Fault : Fault in an element that is not covered by a safety mechanism
  - LFM (Latent Fault Metric) : shows robustness of the item to latent faults
    - Latent Fault :Multi-point fault whose presence is not detected by a safety mechanism
  - PMHF (Probability Metric for random Hardware Failures)
    - Calculating the system failure rates and assessing the ASIL for functional safety

Systematic Failures

- Systematic failure is a failure that arises from the activity itself that develops and produces a system.
    - Human error of personnel participating in development and production activities is the biggest cause.
- RTL bugs caused by incorrect design in the semiconductor design process are typical systematic failures

How to prevent these Systematic Failures?

- ISO 26262 relies on the traditional design verification methodologies
- However, as system complexity increases, errors caused by unintended action that occur during interactions conditions that are difficult to detect with existing verification methods are often found at the silicon level
- To detect above complex systematic fault, another new robust methodologies are required.

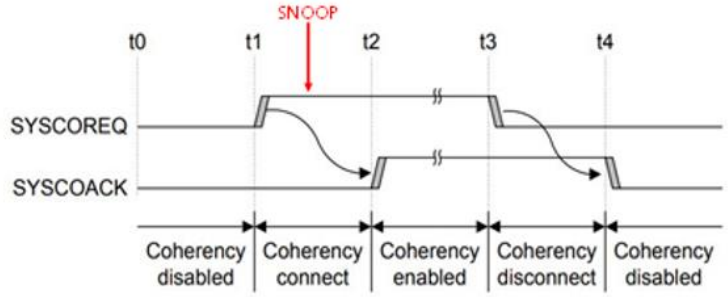
Example of functional corner case bugs - 1

- Errata cases reported from IP provider

Cluster might not response to snoop during coherency connection handshake

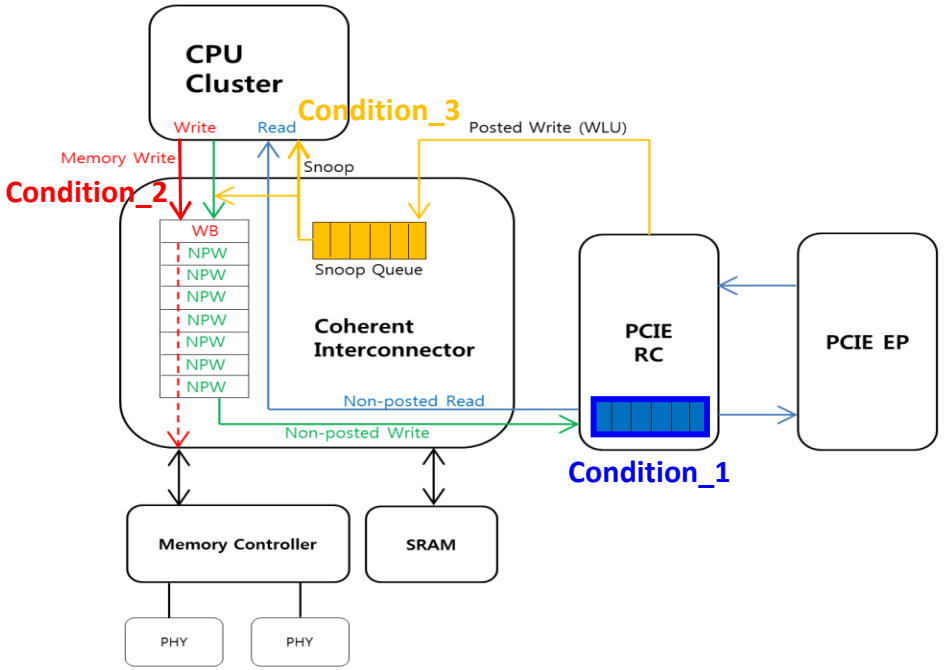
Conditions:

1. The Cluster is in the OFF, MEM_RET, or DEBUG_RECOV power mode
2. The Cluster is powered on by the system requesting a transition on the cluster P-channel to the ON power mode. This caused the Cluster to request to connect to system coherency (SYSCOREQ=1, SYSCOACK=0).
3. The interconnect sends a snoop to the Cluster after it has observed SYSCOREQ HIGH but before it has asserted SYSCOACK.
4. The interconnect has a dependency that causes it to delay asserting SYSCOACK until the snoop transaction is outstanding



Example of functional corner case bugs - 2

- Protocol conflict between PCIe and ACE interface



Condition_1: PCIe RC buffer overflowed

Condition_2: CPU generates Writeback transaction

Condition_3: Snoop generated from posted write of PCIE

Agenda

- Introduction of ISO 26262
- Systematic Failure Analysis
- Systematic failure model generation using Machine Learning
- SFA for requirements-driven verification
- Conclusion

Introduction of SFA (Systematic Failure Analysis)

- We created Systematic Failure Analysis (SFA) to expand the functional verification coverage by extracting risk factors from the IP level and predicting risks.
  - Failure mode definition
  - Risk assessment
  - FMEA (Failure Mode and Effect Analysis)
  - DFA (Dependent Failure Analysis)

Failure mode definition for SFA

- Failure mode is created to predict possible failures
  - FM1: Integration issues (connection, configuration..)
  - FM2: Accessibility issue (access path, access control…)
  - FM3: Functionality issue (wrong output, unintended behavior…)
  - FM4: State transition issues (power gating, clock gating, reset…)
  - FM5: Absence of independence or FFI (Freedom from Interference)

Risk factors

- Hazardous functionality
- Proven in use level
- Severity level
- Known issues in another project
- Applicable workaround

Definition of SFSL (Systematic Failure Severity Level)

- SFSL indicates the functional safety level guaranteed by the system.

$$Risk\ Level = \frac{P(4..1) + S(4..1) + H(4,0) + K(4,0)}{W(4..1)}$$

SFSL	Level Definition	Description
SFSL_A	Risk Level > 12	Very high risk of critical failures. Detailed verification is required
SFSL_B	Risk Level > 8	High risk of critical failures. Additional verification is required
SFSL_C	Risk Level > 4	Mid risk of critical failures. Impact analysis is required
SFSL_D	Risk Level <= 4	Low risk of critical failures.

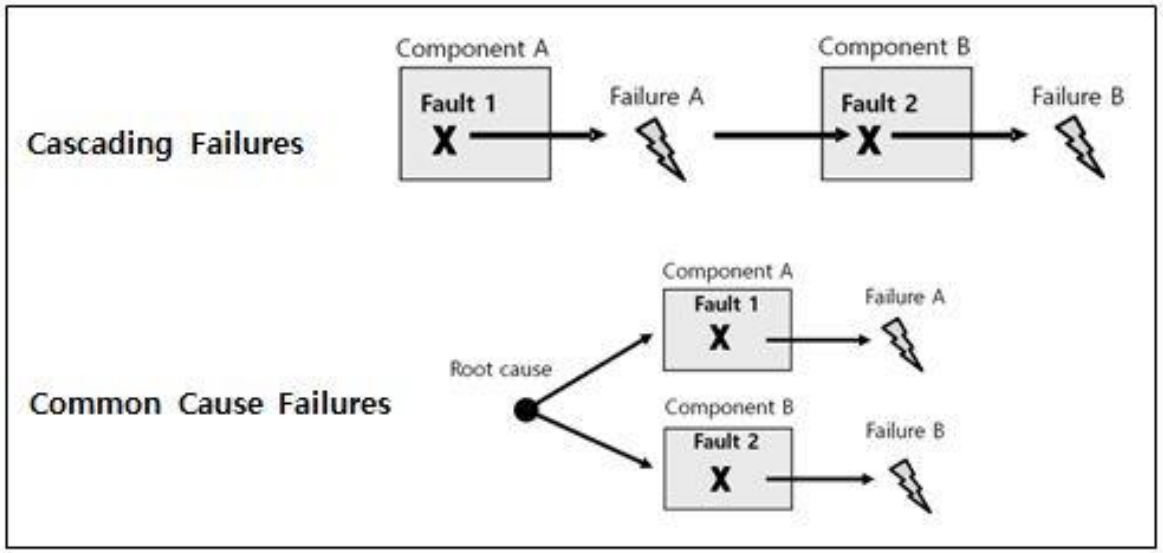
FMEA (Failure Mode and Effect Analysis)

- Failure Mode and Effects Analysis (FMEA) determines all possible ways a system component can fail and determines the effect of such failures on the system.

	FMEA												
Name / Function		Potential Failure Mode(s)	Potential Cause of Failure	Potential Effect of Failures	Risk Assessment						Related high level functions	Occurance Conditions
ID	Requirements				SFSL	P	S	H	K	W		
CPU_CPD_FM3	CPUCL_F1	P-ch handshaking has failed	wrong connection of P-CH interface	Power mode transition does not working	B	3	3	Y	N	1	SYSTEM idle/sleep mode	try power mode transition
CPU_CPD_FM5	CPUCL_F1	ACE interface stalled after snoop arrived between coherency disconnect and coherency disable	reported Errata: 1500609	Deadlock occurred between CPUCL and BUS	A	3	4	Y	Y	1	SYSTEM sleep mode	Generate snoop between SYSCOREQ and SYSCOACK
CMU_ACG_FM1	CPUCL_F2	wrong clock pll ratio	wrong PLL configuration	generate wrong clk out	D	1	2	N	N	3	Normal active mode	Check clk after CMU init
CMU_ACG_FM5	CPUCL_F2	unintended clock gating occurred during CPU is in active state	Interference occurred between clk gating sequence	Deadlock occurred due to incompleted transactions	A	2	4	Y	Y	1	SYSTEM sleep mode throttling enable	Access CPUCL0 register between cpucl0_clk_gating_en and cpucl0_clk_blocking_ext_en

DFA (Dependent Failure Analysis)

- The analysis of dependent failures aims to identify the single events or single causes that could bypass or invalidate a required independence or freedom from interference between elements and violate a safety requirement or a safety goal



DFA implementation for DV

- We've found DFI (Dependent Failure Initiator) and coupling factors by:
  - Fault injection
    - Uncorrectable ECC error injection (DRAM/L3DCache/L1,L2 Dcache)
    - Memory Management Unit(MMU) translation fault generation
    - RAS (Reliability, Availability, and Serviceability) error injection for CPU, Interrupt controller, System MMU
  - Generate interference stimulus for a shared memory region
    - False sharing coherency access
    - Distributed Virtual Memory(DVM) transaction broadcasting
    - Exclusive access
    - CPU cluster power down

Output of DFA

- FTR (Fault Tolerance Report)

Fault Tolerance Report (FTR)

Fault Injection					Interference stimulus					Simulation result		Scenario info	
FMEA_ID	type	target	expected failures	FTR_ID	stimulus_1	stimulus_2	stimulus_3	stimulus_4	stimulus_5	Recovery result	Fault Tolerance Report (FTTI)	Scenario name	Seed number
M001	ECC error	DRAM	error interrupt/ error response	M001_1	false sharing access					done	80	dram_1_ecc_1	3523
				M001_2	false sharing access	exclusive access				done	100	dram_1_ecc_2	3475
				M001_3	false sharing access	exclusive access	MMU page remap			done	105	dram_1_ecc_3	2531
				M001_4	false sharing access	exclusive access	MMU page remap	cluster powerdown		done	105	dram_1_ecc_4	3767
				M001_5	false sharing access	exclusive access	MMU page remap	cluster powerdown	DFS level change	done	110	dram_1_ecc_5	8236
				M001_6	exclusive access					done	50	dram_1_ecc_1	3257
				M001_7	exclusive access	MMU page remap				done	55	dram_1_ecc_2	3278
				M001_8	exclusive access	MMU page remap	false sharing access			done	90	dram_1_ecc_3	4291
				M001_9	exclusive access	MMU page remap	false sharing access	DFS level change		done	93	dram_1_ecc_4	3982
				M001_10	exclusive access	MMU page remap	false sharing access	DFS level change	cluster powerdown	done	97	dram_1_ecc_5	7218

Output of DFA

- DFA result

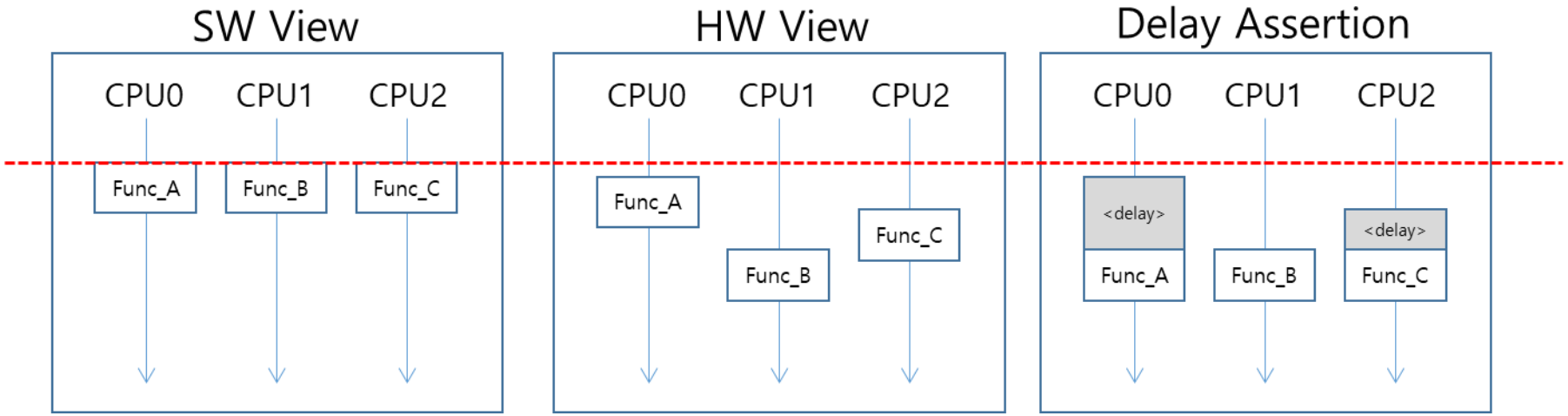
	Dependent Failure Analysis (DFA)						
FMEA_ID	Element	Redundant Element	Functional Dependency	Dependent Failure Initiator(DFI)		DFA	Verification Method
	Short name and description	Short name and description	Description	Systematic fault	Shared resource	Expected Dependent Failure	
CPU_CPD_FM5	BLK_CPUCL0	BLK_GPU	CPU should wake up GPU for requested GPU processing	Stalled ACE interface of CPU		GPU can't wakeup and system hang occurred	PSS_ML_fault_model
	BLK_CPUCL0	BLK_PCIe	PCIe will send posted write request and it will generate snoop to CPUCL0	Stalled ACE interface of CPU		PCIe will not available. Posted write will wait for snoop response from CPU.	PSS_ML_fault_model
MIF_FAULT_1	Memory scheduler:ECC logic	BLK_CPUCL0	False sharing		ECC error generated from shared DRAM region	CPU will access fault address during ECC error state	PSS_fault_injection_model
	Memory scheduler:ECC logic	BLK_CPUCL0	exclusive access		ECC error generated from shared DRAM region	CPU will access fault address during ECC error state	PSS_fault_injection_model

Agenda

- Introduction of ISO 26262
- Systematic Failure Analysis
- Systematic failure model generation using Machine Learning
- SFA for requirements-driven verification
- Conclusion

Requirements of systematic failure model

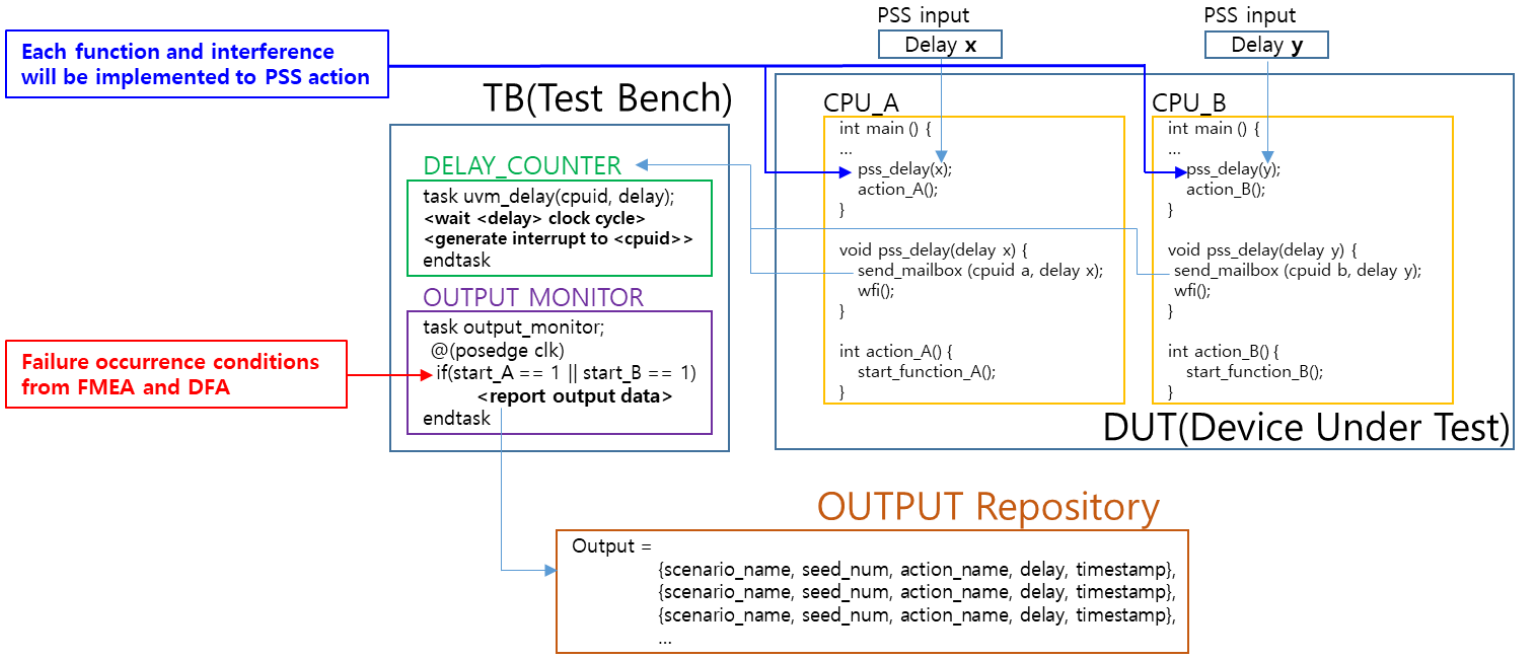
- Even if SW is executed at the same time, the resulting HW event occurs differently.



- We had to insert delay to make synchronized HW events.

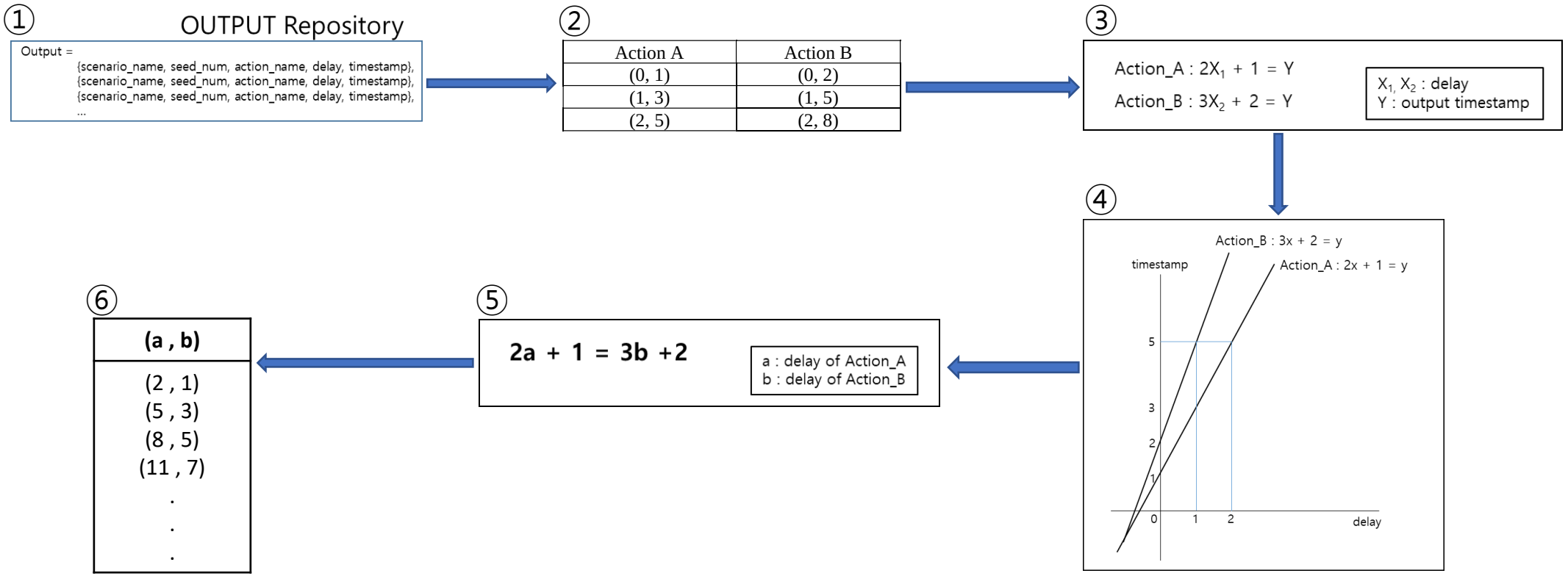
TB structure of the systematic failure model

- We've created UVM Delay counter/Output monitor and Output repository for Machine Learning and result analysis

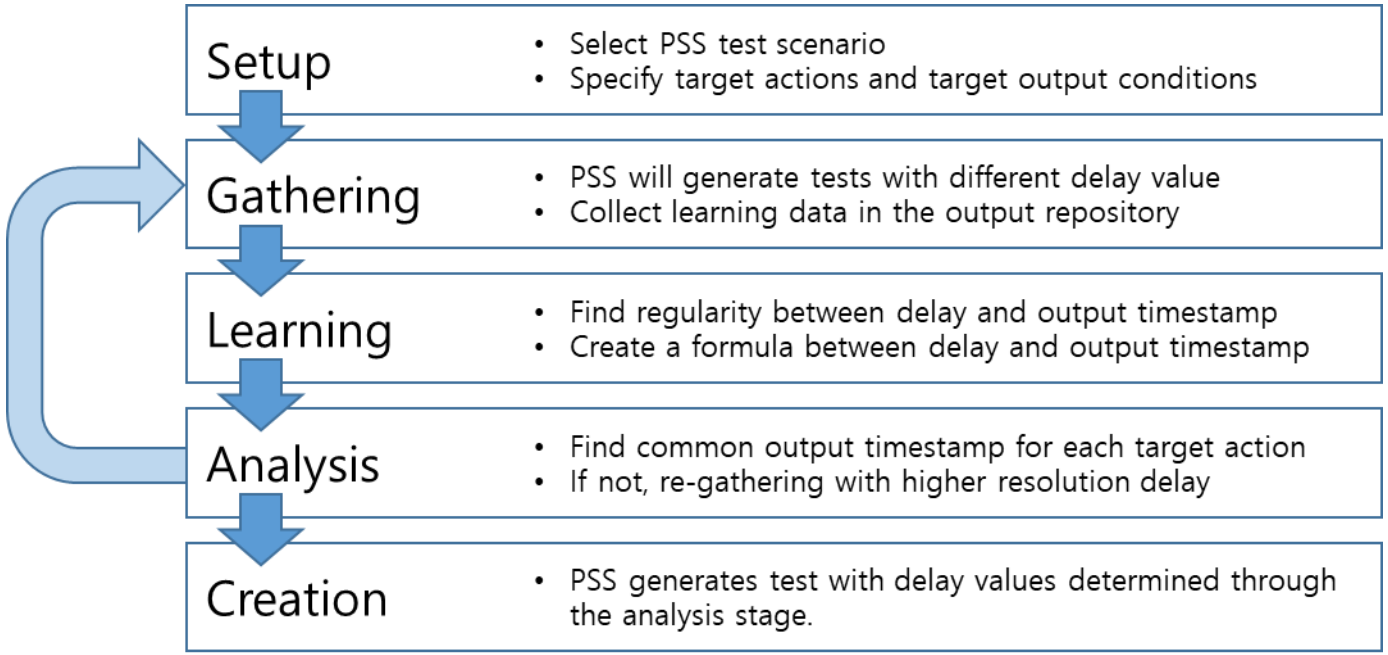


- Generated Tests from PSS
- Delay Counter
- Output Monitor
- Output Repository

Machine Learning implementation



ML sequence modeling flow



Agenda

- Introduction of ISO 26262
- Systematic Failure Analysis
- Systematic failure model generation using Machine Learning
- SFA for requirements-driven verification
- Conclusion

Reusable output of SFA

- SFA output could be reuse for various requirements based standards

Architecture and design specification

- Change management and impact analysis report
- Detailed hardware design specification and requirements

Verification plan

- Tool, methods and environments that used for verification
- Verification strategy for target design

Verification specification

- Risk analysis report
- Function list with correlations for target design
- Test cases, test data and objects

Verification report

- FMEA report
- DFA report
- Coverage report

Agenda

- Introduction of ISO 26262
- Systematic Failure Analysis
- Systematic failure model generation using Machine Learning
- SFA for requirements-driven verification
- Conclusion

Conclusion

- Higher levels of reliability will be required for semiconductors
    - Reinforced HARA(Hazard Analysis and Risk Assessment) process will be required
    - Innovative expansion of the verification coverage is needed
- ISO 26262 is not a reference. It will be a common requirements for our future development process.

- Questions and Answers
  - Please feel free to contact me (moonki.jang@gmail.com)