2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

How to Stay Out of the News with ISO26262-Compliant Verification

Charles Battikha (chuck_battikha@mentor.com)

Doug Smith (doug_smith@mentor.com)

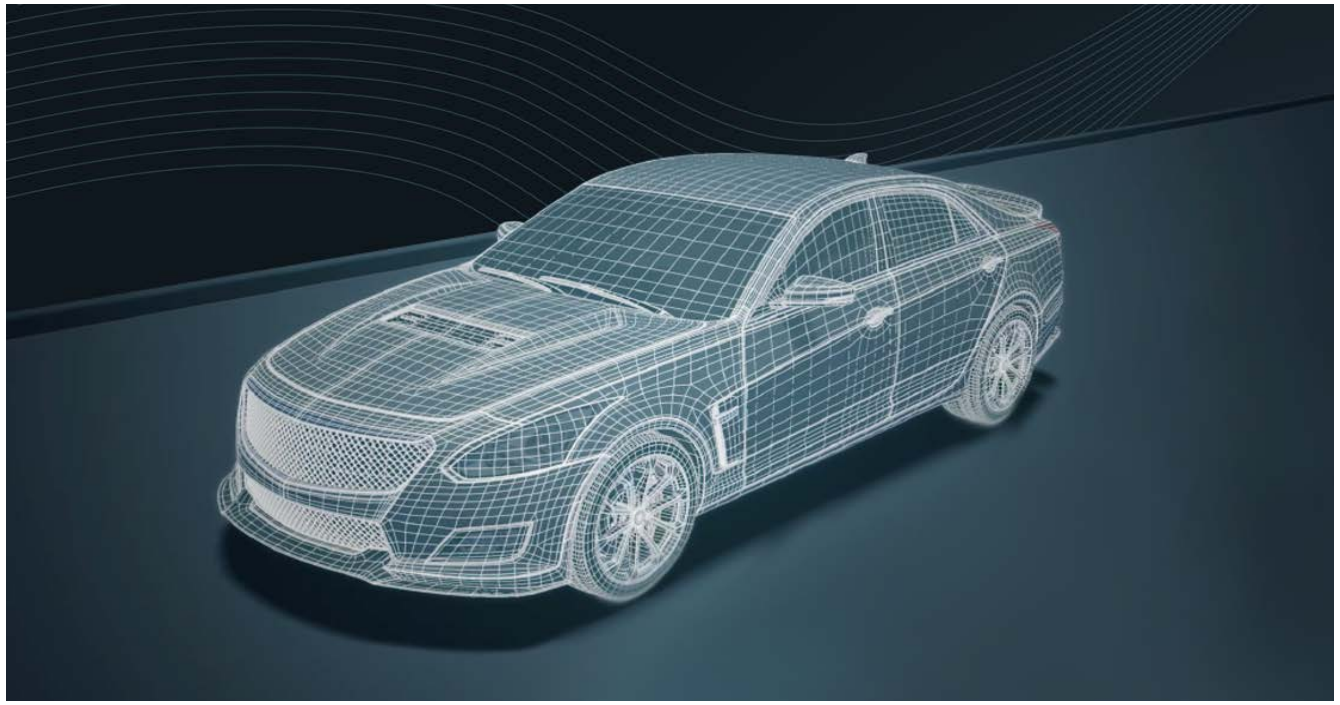
Mentor®
A Siemens Business

Agenda

- Taking New Products into the Automotive Market…
  Welcome to Functional Safety
- From Analysis to Fault Campaigns
- Break
- How Formal Reduces Fault Analysis for ISO 26262
- Requirement Tracing in the ISO26262 World

Entering the Automotive Market…

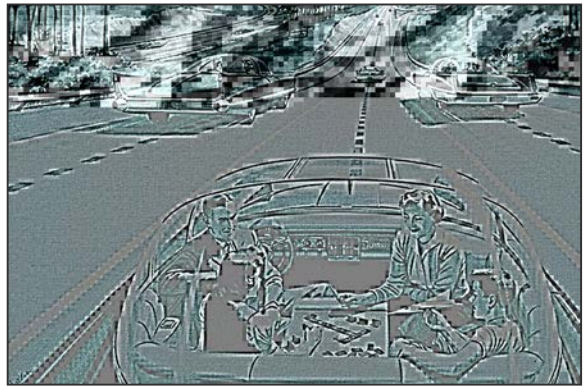
- Reviewing the challenges and requirements in the Automotive Market & ISO26262 Standard



Automotive Market Drivers

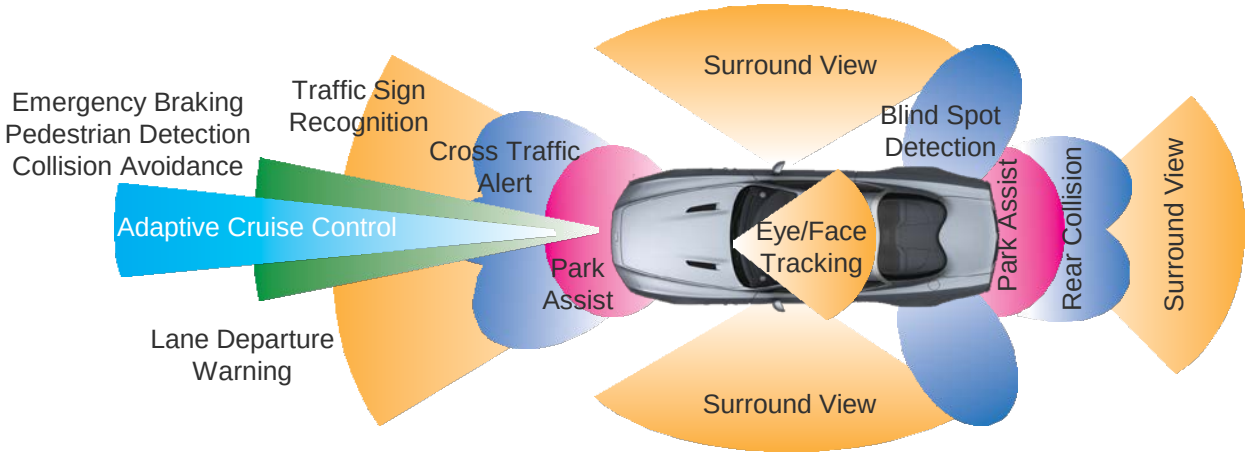
- Electrification
- Smart Sensors
- Sensor Fusion
- Vehicle Networking
- V2X Connectivity
- ADAS

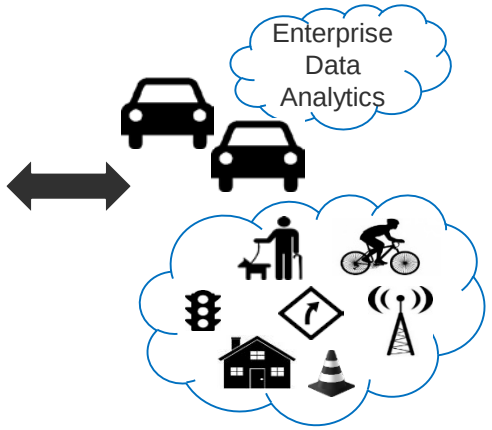
The Autonomous Car



*High-end cars will contain more than **$6,000** worth of electronics in five years, driving a **$160 billion** automotive electronics market in 2022"*

Luca De Ambroggi
Principal analyst Automotive electronics
IHS Markit





Complexity of Automotive Systems

LINES OF CODE

- Hubble Space Telescope
- 5M Mars Curiosity Rover
- 12M Smartphone OS
- 25M F=35 Fighter Jet
- Luxury car software 100M

6 radar beams

8 cameras

12 parking sensors

144 electronic control units

500 LEDs

734 wire harnesses

2,400 wires

5,000 meters of cables

source: Roland Berger
source: CarsGuide

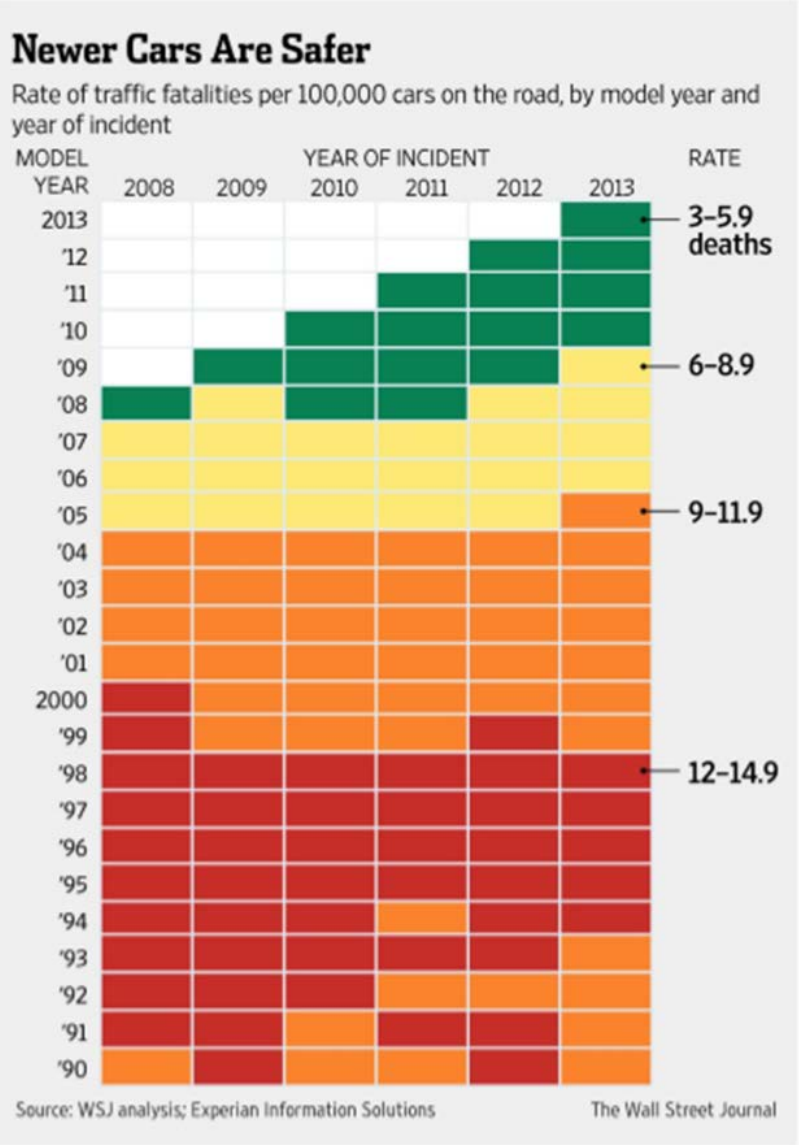
Functional Safety = Table Stakes

Severity of Injuries + Probability Exposure + Driver Control = ASIL Set by OEM

		C1	C2	C3
S1	E1	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	A
	E4	QM	A	B
S2	E1	QM	QM	QM
	E2	QM	QM	A
	E3	QM	A	B
	E4	A	B	C
S3	E1	QM	QM	A
	E2	QM	A	B
	E3	A	B	C
	E4	B	C	D

Goal is the create a ***highly assured*** design by removing ***unreasonable risk***

ASIL D
S3=Life-threatening, fatal injuries
E4=High probability of exposure
C3=Difficult for driver to control



Functional Safety Tradeoffs

Integrator Cost (Components)
Integrator Development Effort

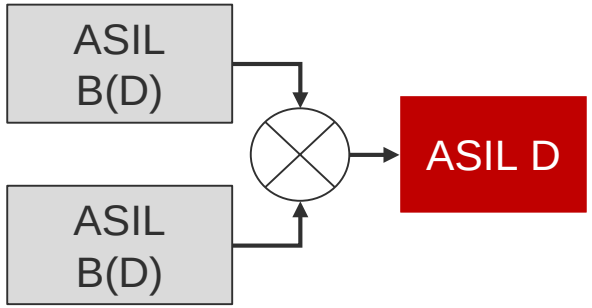
IP / IC Cost (Area)
IP / IC Power (Area)
IP / DC Development Effort
Increased ASIL

IC/IP with ISO Compliance and higher ASIL ratings reduces effort, cost, and complexity for integrators.

Increased Cost & Development Examples:

- Reviews, Audits, Assessments
- Additional Logic in Safety Mechanisms
- Additional Rigor & Deliverables

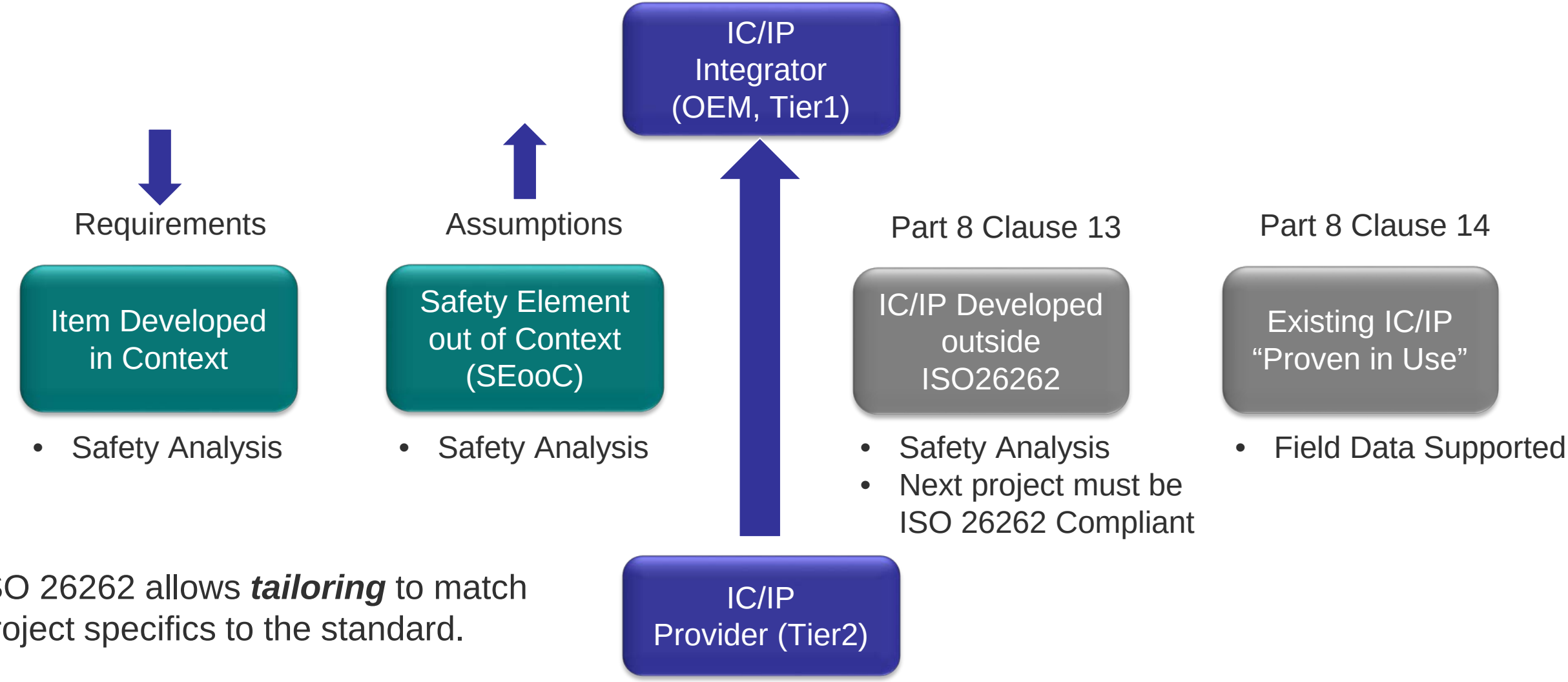
ASIL Decomposition



Redundancy creates:

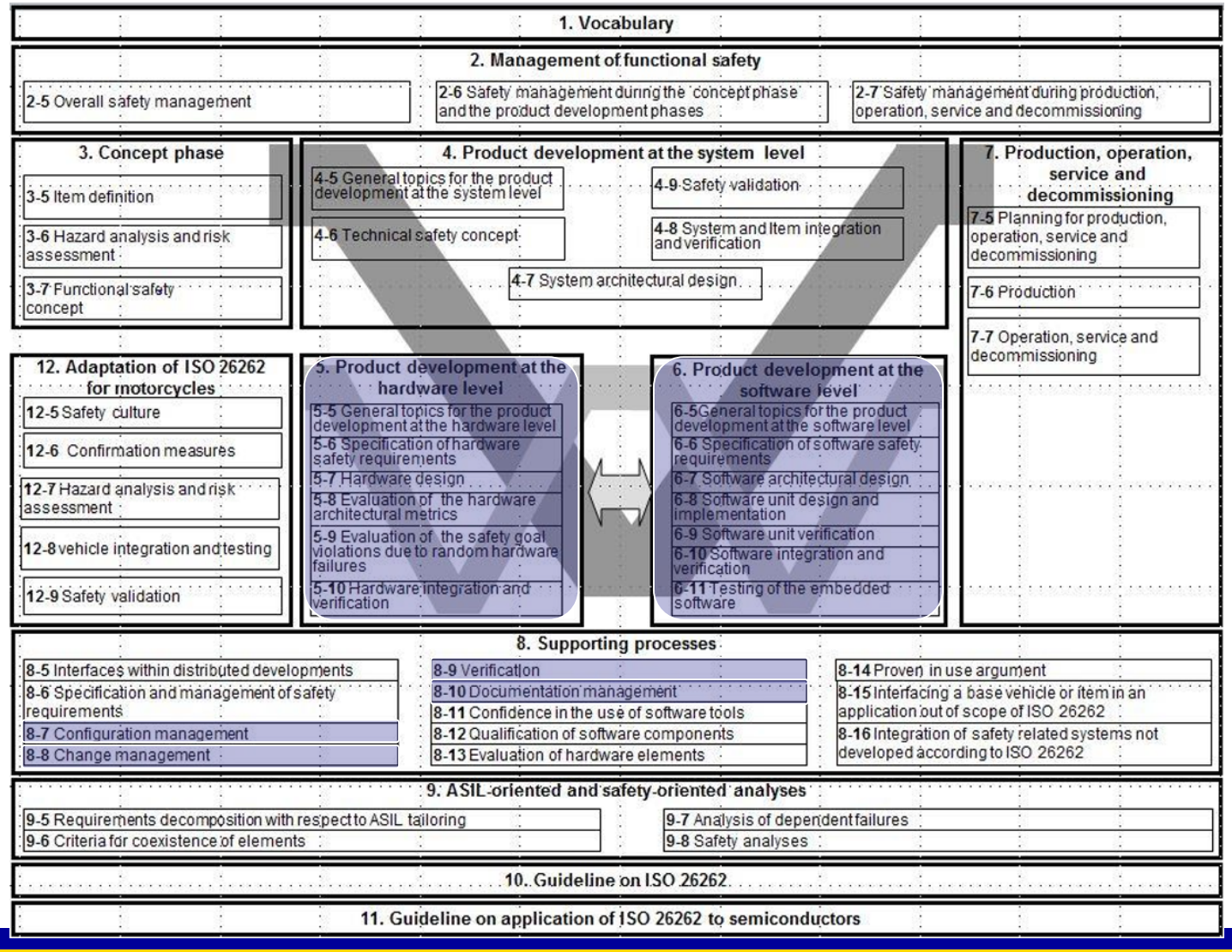
- Complexity
- Verification effort
- Area & cost penalty

IC/IP Categories



ISO 26262 allows ***tailoring*** to match project specifics to the standard.

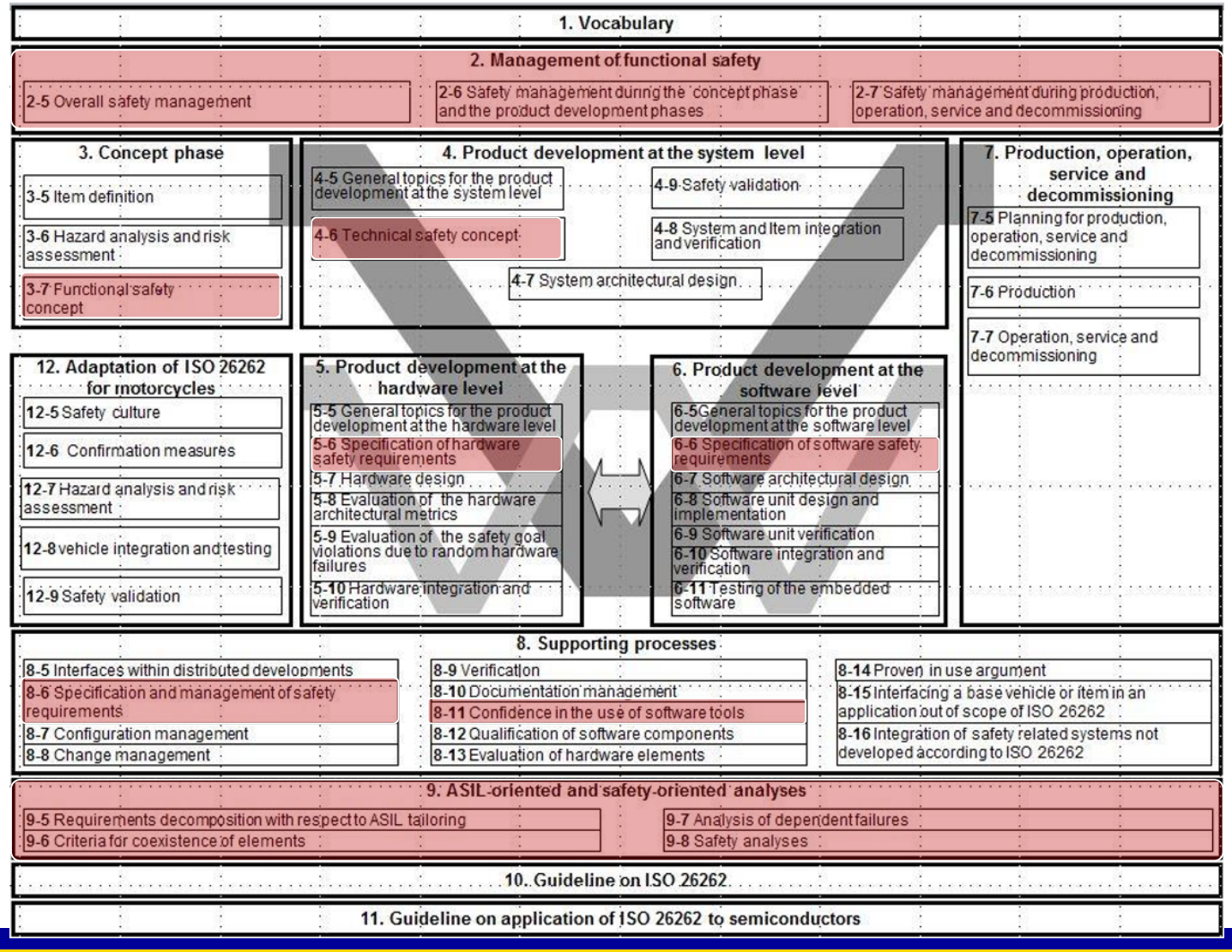
ISO26262 – The known



Established IC/IP Developers likely have strong development & verification processes:

- Development process well documented
- … & shown to be followed
- Create and maintain artifacts
- Requirements tracing
- Source Control
- Change Management
- Documentation Control

ISO26262 – The unknown



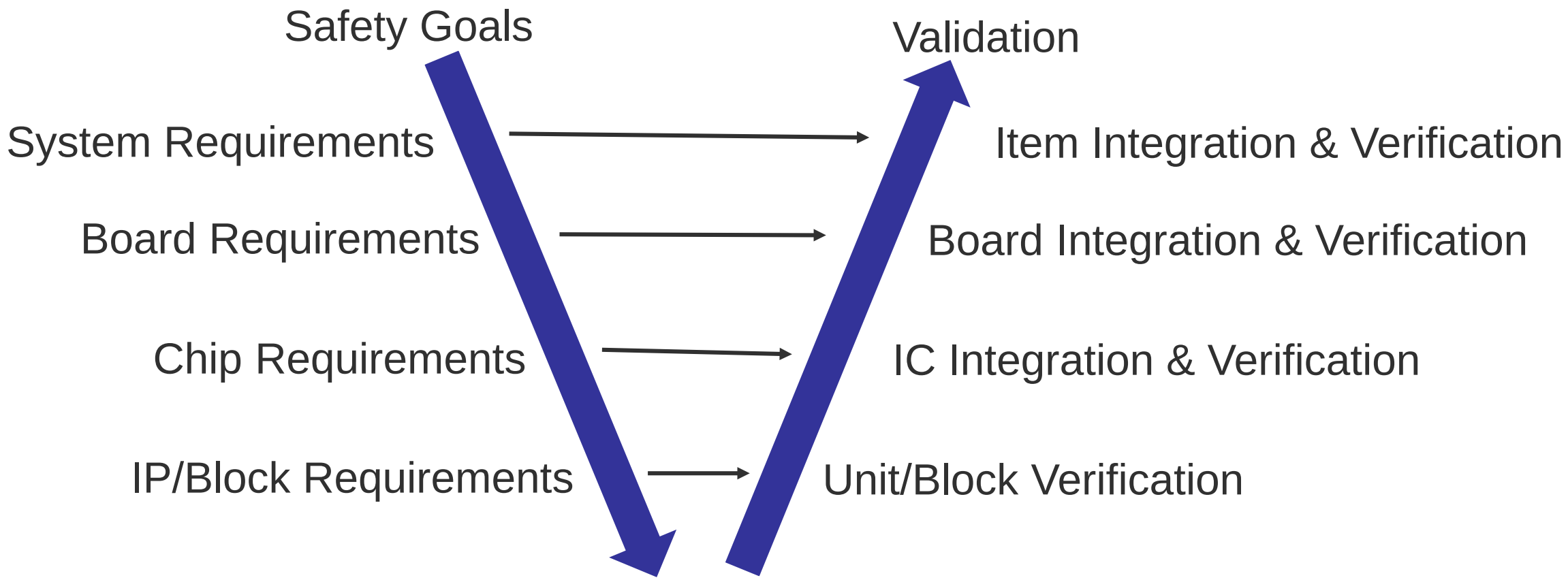
IC/IP developers new to the automotive market will find:

- Safety Culture
- Safety Requirements
- Safety Mechanisms
- Safety Analysis
- FMEA / FMEDA / DFA / FTA
- Fault Metrics
- Fault Insertion Campaign
- Safety Manual
- Tool Qualification

“V” Development & Verification

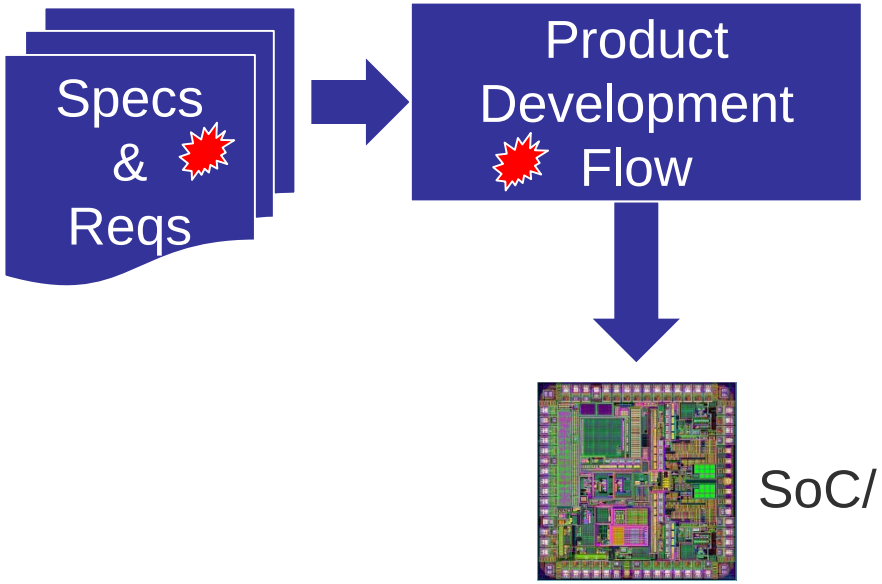
Flow Down Safety Requirements

Bottom Up Integration and Verification



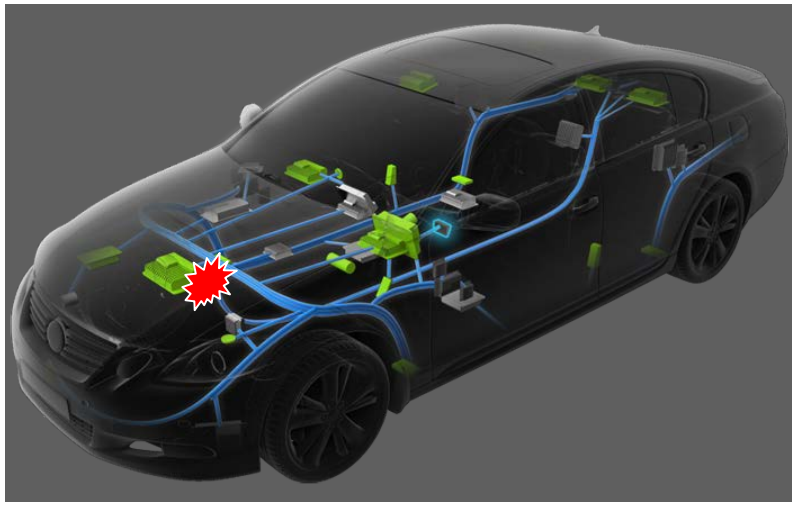
Requires Two Testing Approaches

Systematic Failures
Introduced in product development



- *Incorrect Requirements*
- *Inaccurate/incomplete specs*
- *RTL Errors*
- *Timing Errors*

Random Failures
Introduced by the environment



- *Vibration*
- *Moisture/Dirt*
- *Noise*
- *EMI*
- *Electro-migration*

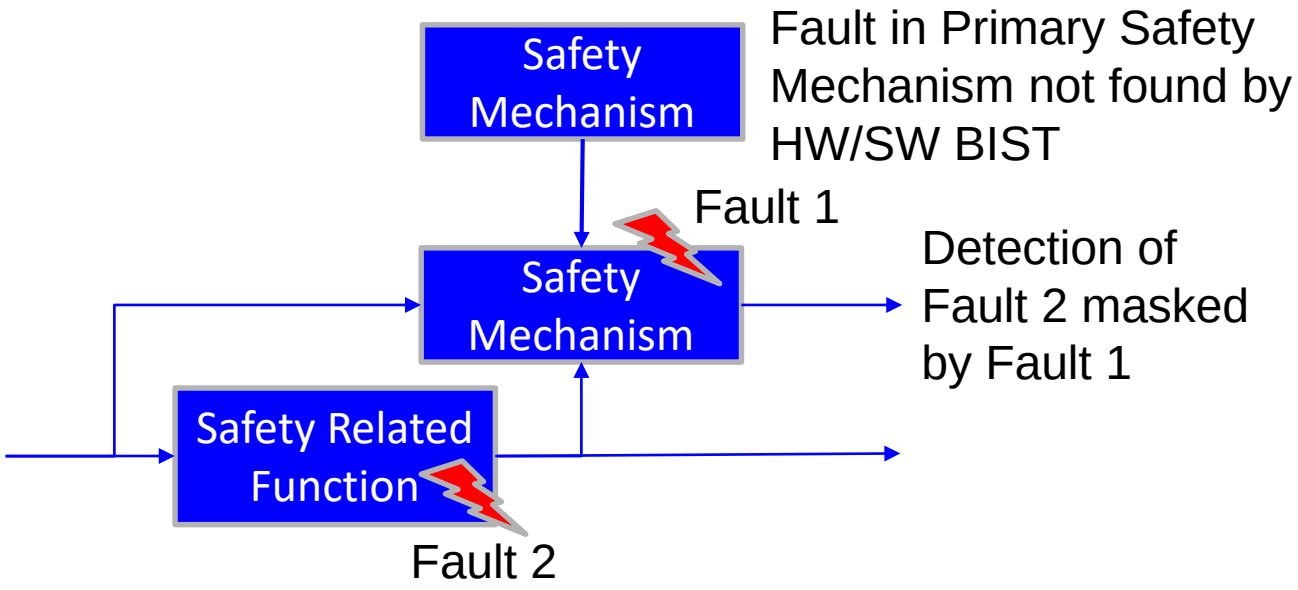
Fault Types

λ_S	**Safe Faults**; Does not effect the Safety Requirements
λ_{SPF}	**Single Point Fault**; Fault violating a Safety Requirements. Not covered by a Safety Mechanism. Should be addressed.
λ_{RF}	**Residual Faults**; Faults not detected by an intended Safety Mechanism and lead to a violation of Safety Requirements. Single Point Faults and Residual Fault are not differentiated from a fault analysis perspective. **Diagnostic Coverage** measures effectiveness of safety mechanism in detecting Residual Faults – permanent and transient.

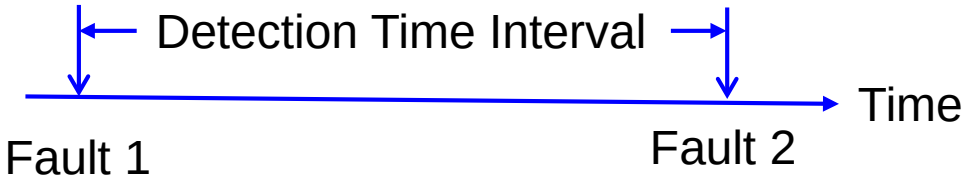
$\lambda_{SPF} + \lambda_{RF} \rightarrow$ SPFM (ASIL Goal)

Fault Types

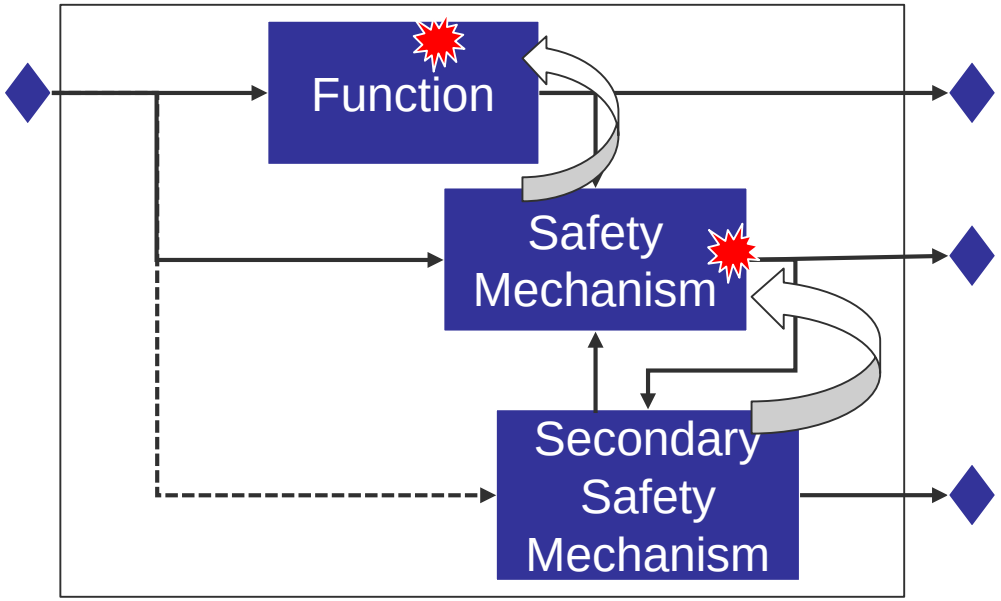
$\lambda_{DPF,DP}$	**Dual-Point Faults – Detected/Perceived**; Combination of independent faults that may lead to a violation of Safety requirements.
$\lambda_{DPF,L}$	**Dual-Point Faults – Latent**; Faults not detected by safety mechanisms that would lead to a dual-point failure. Considered to be a fault in primary safety mechanism that is undetectable.



$\lambda_{DPF,L}$ ➔ LFM (ASIL Goal)



Diagnostic Coverage



Safety Critical Block

$$DC_{RF} = \left(1 - \frac{F0}{F0 + F1 + F2 + F3}\right) \times 100$$

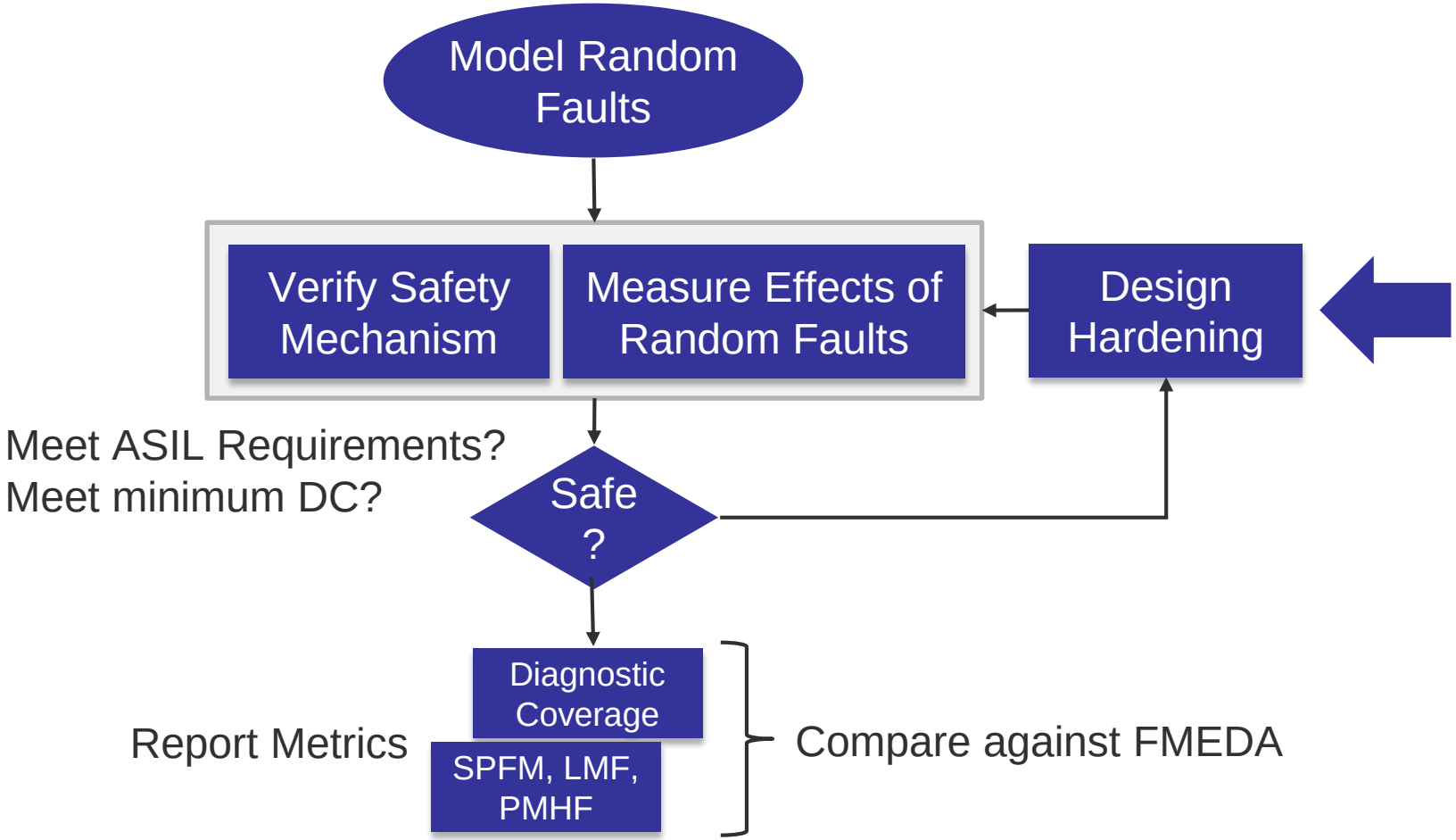
$$DC_{DPF,\,L} = \left(1 - \frac{F0}{F0 + F1 + F2 + F3}\right) \times 100$$

		Safety Mechanism	
		Does Not Detect Fault	***Detects Fault***
Safety Output	***Not Affected***	F1 F_{SAFE}	F2 F_{SAFE}
	Affected	F0 F_{RF}	F3 $F_{SAFE,DET}$

$F_{DPF,DP}$

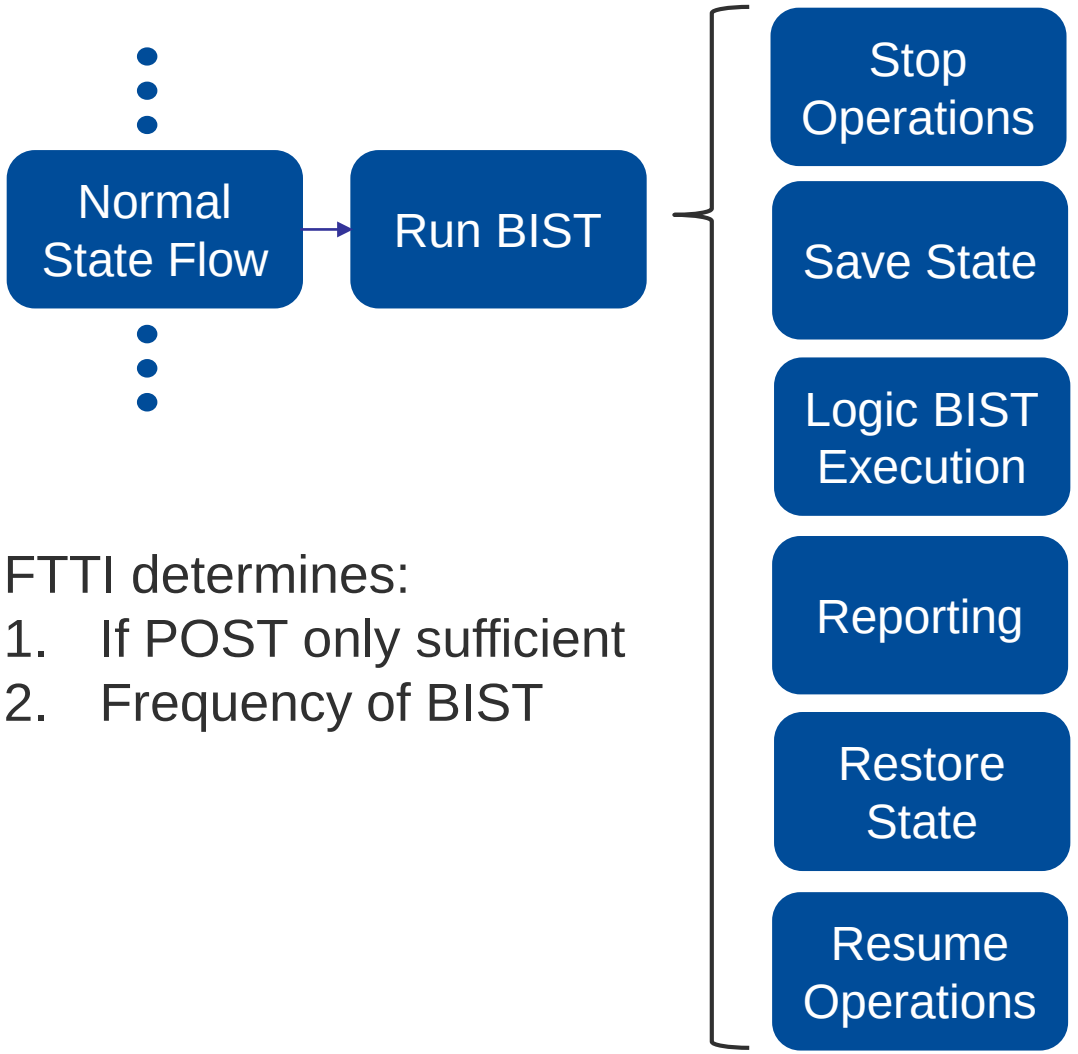
DC: Proportion of the failure rate that is detected or controlled by implemented safety mechanisms.

Verification of Random Hardware Faults



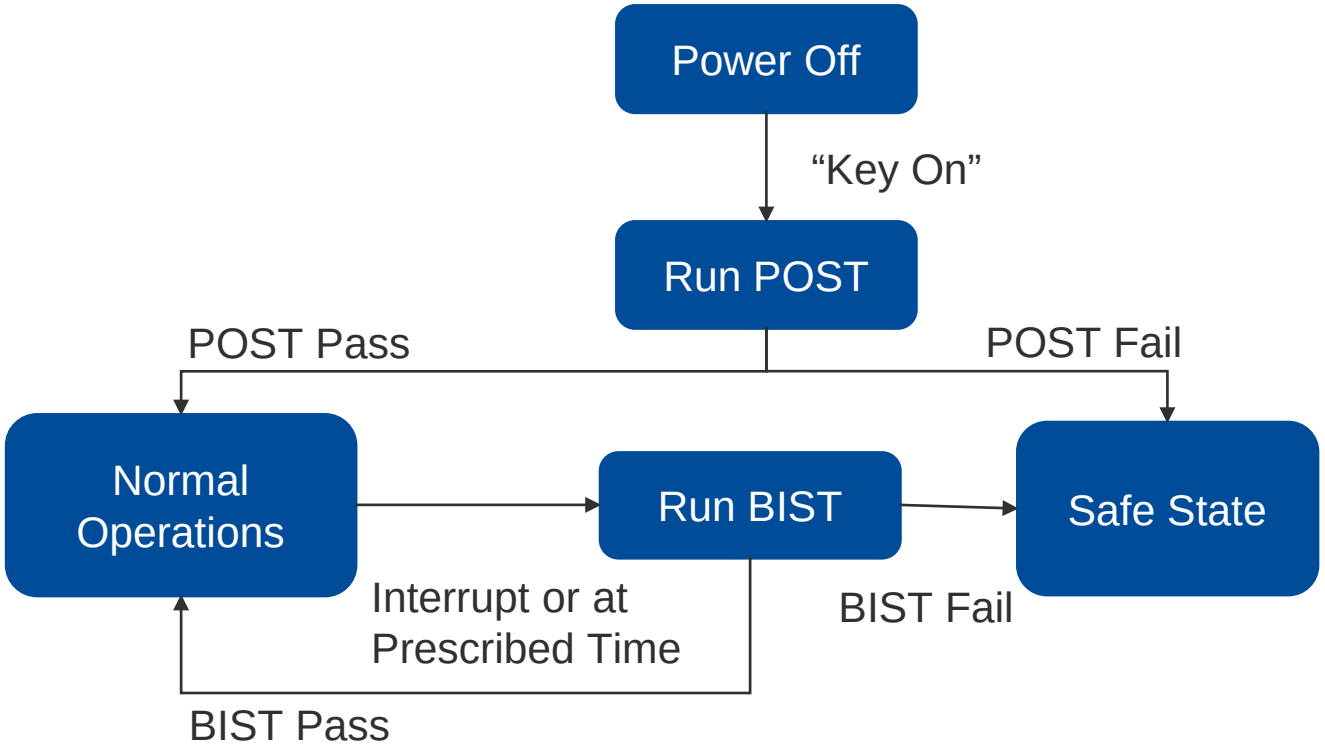
- Power On Self Test
- Interrupt-driven BIST
- ECC
- CRC / Checksums
- Watchdog Timers
- Voting
- Main versus Redundant Compare
- Software/Firmware Diagnostics
- Etc.

POST & Interrupt Driven BIST as SM

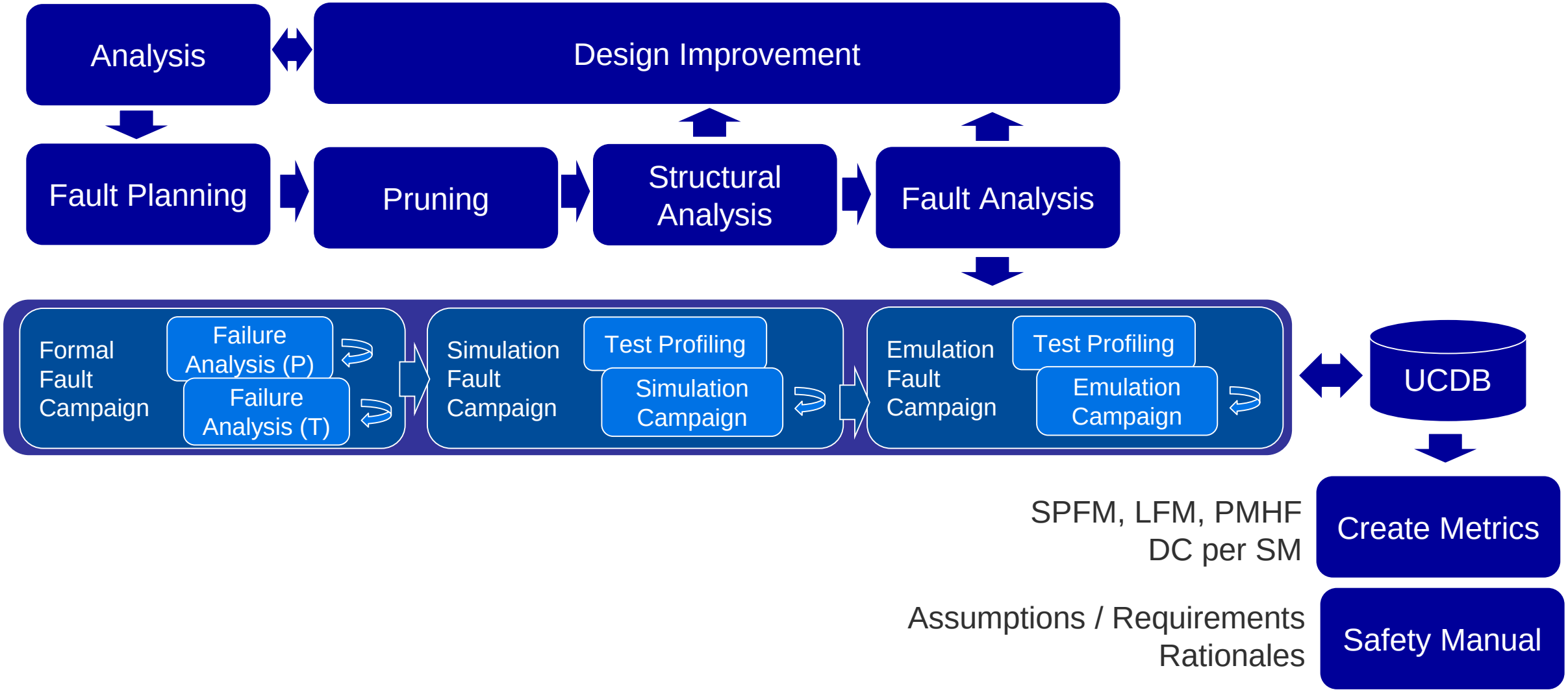


FTTI determines:

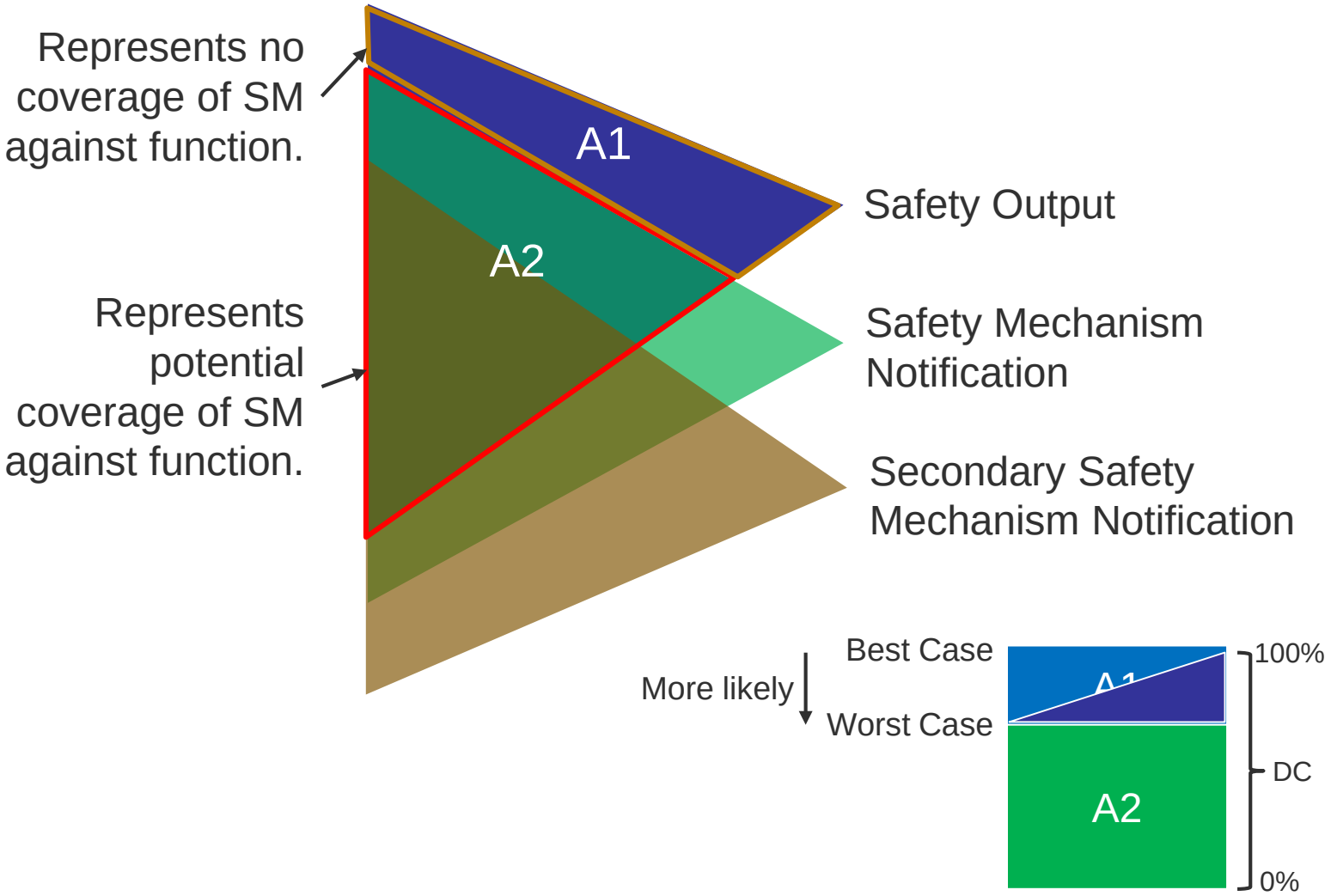
1. If POST only sufficient
2. Frequency of BIST



Mentor Functional Safety Process



Structural Analysis



$A1 = \text{combination}(F_{RF}\ F_{SAFE})$

$A2 = \text{combination}(F_{DP,DET}\ F_{RF}\ F_{SAFE})$

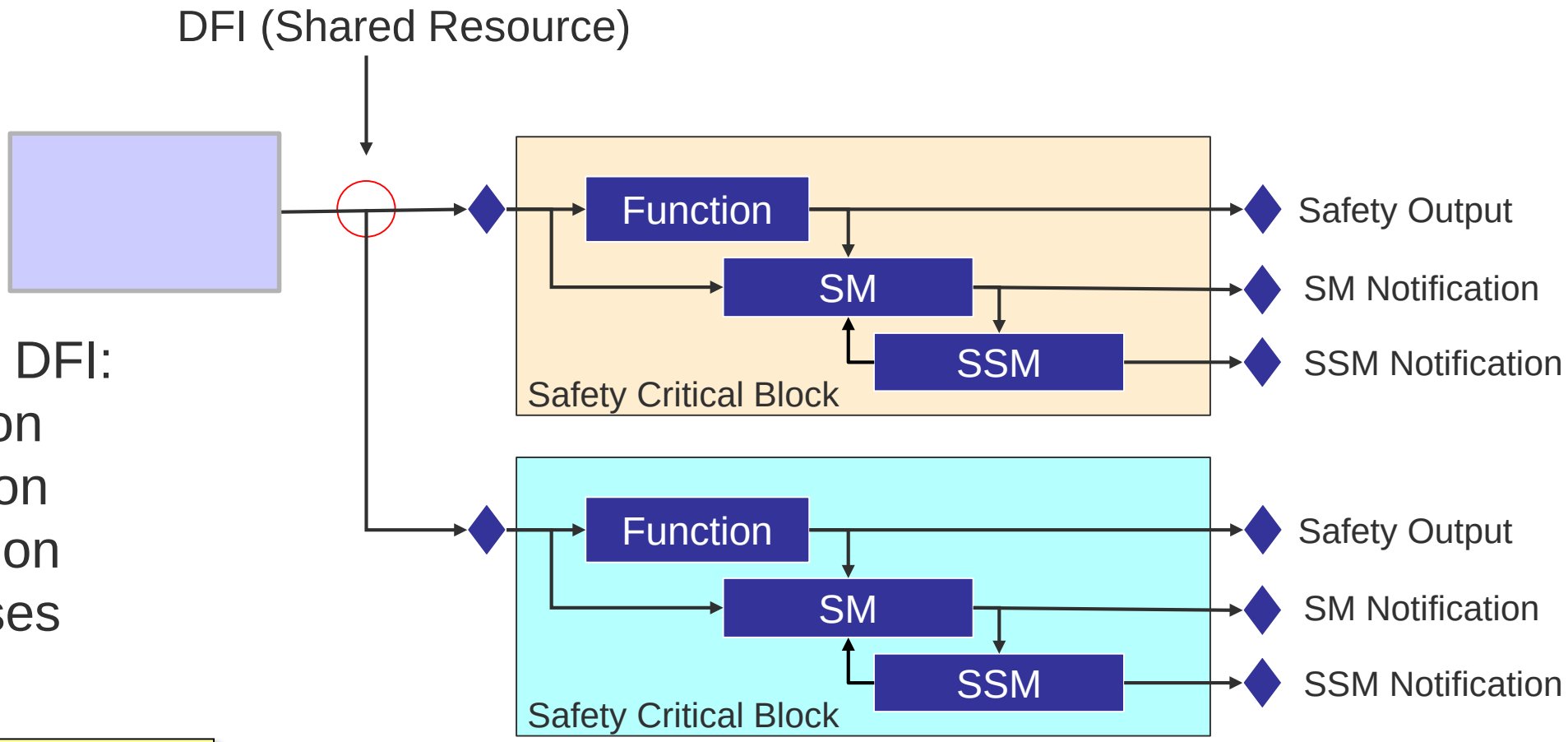
Likely residual fault distribution:

$F_{RF,A1} >> F_{RF,A2}$

Creates a max ceiling for DC.

Goal: Reduce area of A1 before starting fault campaign.

Dependent Fault Analysis (DFA)



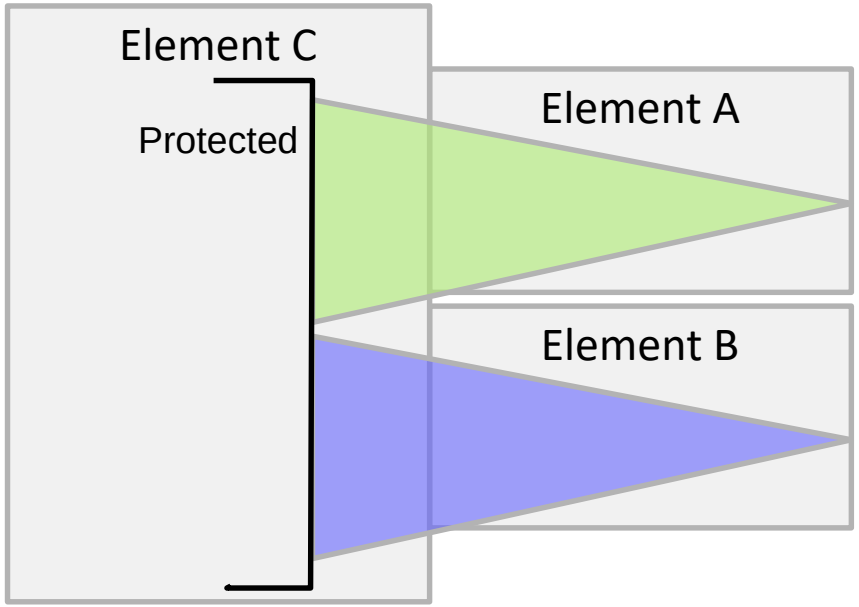
In an IC, common DFI:

- Clock Distribution
- Reset Distribution
- Power Distribution
- Main Data Busses

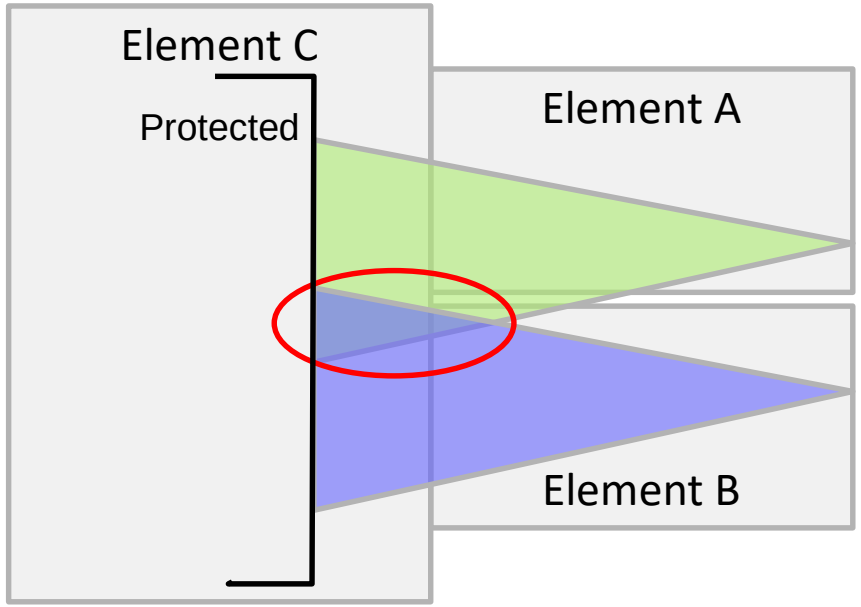
But easy in an IC to create DFI…

DFA & COI

- Use COI to find unintended overlap which implies shared resources
- Cutpoints & black-boxes stop COI tracing when function is protected



Free of Interference



Shared Resources

Summary

- ISO26262 ***is*** Functional Safety
- Requires many companies to create a Safety Culture
- Requires strong development and verification processes
- Requires analysis to address random hardware faults
- Reaching higher ASIL ratings will increase effort and costs

2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

From Analysis to Fault Campaigns

Charles Battikha (chuck_battikha@mentor.com)

Mentor®
A Siemens Business

Topics

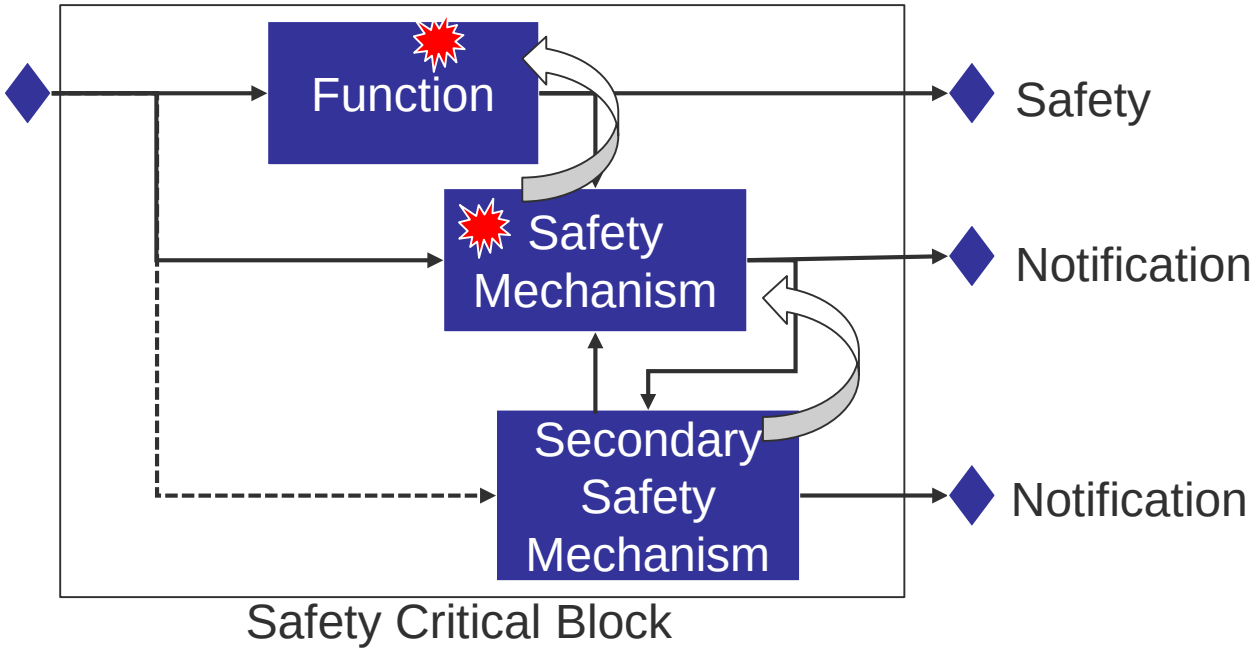
- Recap of Safety Analysis
- Usage of Metrics
- Analysis
- Fault Injection Campaign
- Summary

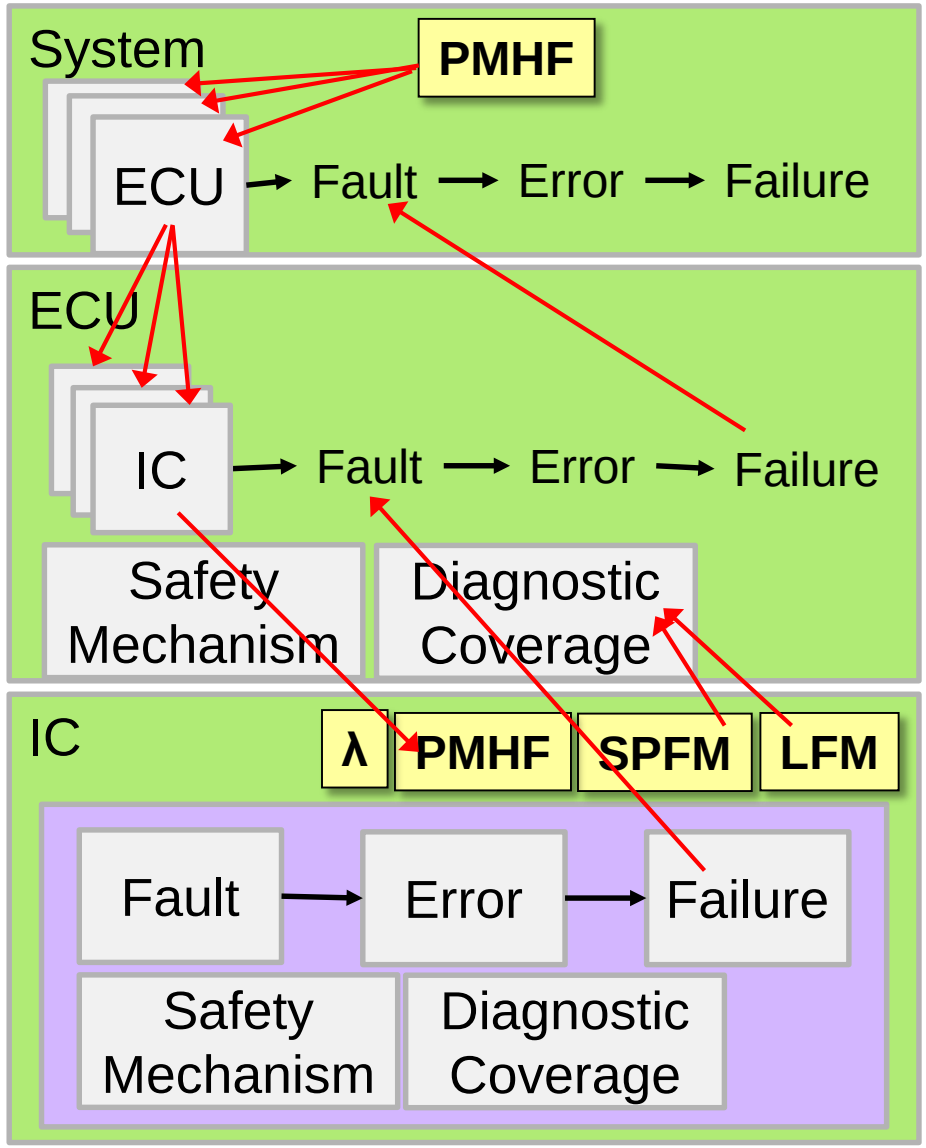
Safety Analysis

- Qualitative Analysis
  - Effects & Causes → FMEA, FTA
  - Dependent Failure Analysis
- Quantitative Analysis
  - Metrics → FMEDA, FTA
  - Analysis of Random Faults
- Fault Injection Testing
  - Verification of Safety Mechanisms
  - Metrics

Usage of Metrics

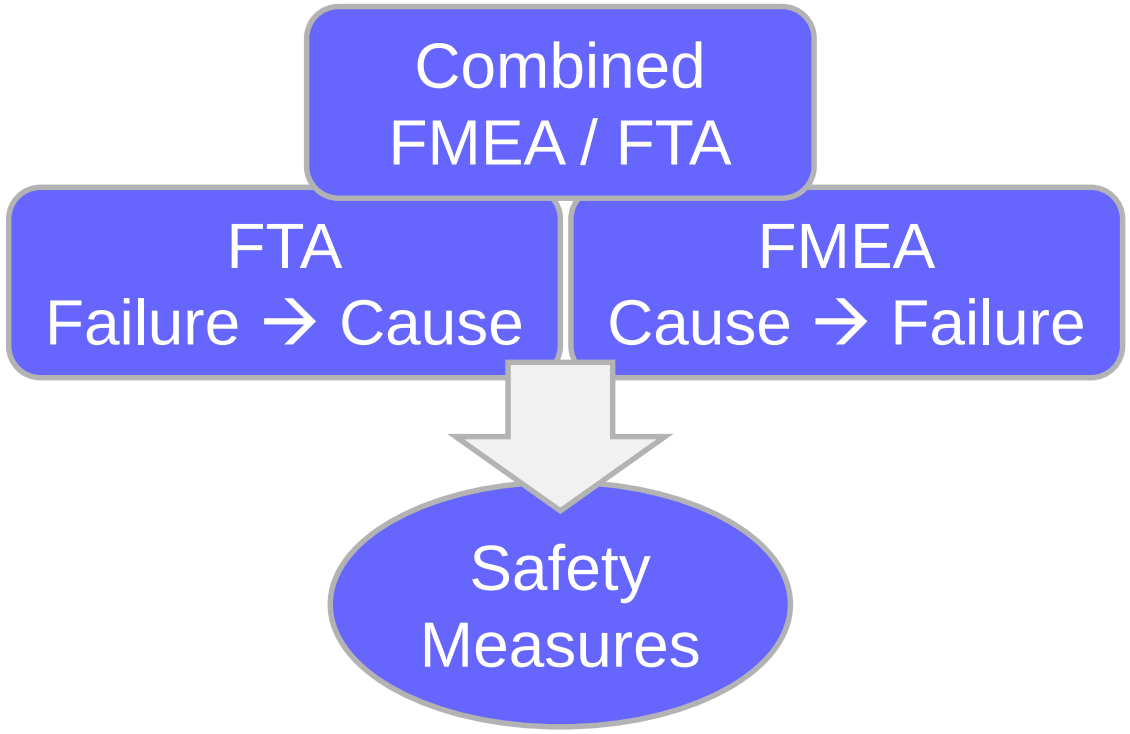
- PMHF – Targets distributed top-down
- SPFM/LFM –Bottom-up, abstracts details of diagnostic coverage within the IC/IP





FMEA & FTA

- Analysis Process to Identify
  - Failure Modes in a function
  - Effects of the failure
  - Potential Causes of the failure
- Information allows definition of
  - Safety Mechanisms
  - Reaction to failure / Safe States
  - Safety Requirements
- FMEA versus FTA versus FMEDA



FMEDA – General Structure

A	B	C	D	E	F	G	H	I	J	K	L	M
#	Block	Safety Related Element (Y/N)?	Failure Mode	Failure Mode Ratio (%)	Effect of Failure Mode	λ (FIT)	Potential to violate a Safety Goal in absence of safety mechanism (Y/N?)	Is there a safety mechanism in place to control failure mode (Y/N)?	Safety Mechanism(s) allowing the system to prevent the failure mode from violating the safety goals (e.g. SM1, SM2)	Failure mode (diagnostic) coverage (%)	λ_{SPF}	λ_{RF}
1	Primary Bridge	Y	Incorrect data written into transmit or configuration register(s)	15%	Incorrect or no SPI Transmission	3.50	Y	Y	SM5	80%	0.000	0.105

N	O	P	Q	R	S	T	U
Potential to vilate a SG, in combination w/ one other independent failure (Y/N)?	Is there a safety mechanism in place to control latent faults (Y/N)?	Safety mechanism(s) allowing to prevent the failure mode from being latent ?	Failure mode (diagnostic) coverage w latent failures	λ_{SAFE}	$\lambda_{MP,L}$	$\lambda_{MPF,DP}$	Justification / Rationale
Y	Y	SM3	70%	0.000	0.126	0.294	

FMEDA

Key document to share upward to integrators of IC & IP

Only Safety Related Elements

A	B	C	D	E	F	G	H	I	J	K	L	M
#	Block	Safety Related Element (Y/N)?	Failure Mode	Failure Mode Ratio (%)	Effect of Failure Mode	λ (FIT)	Potential to violate a Safety Goal in absence of safety mechanism (Y/N?)	Is there a safety mechanism in place to control failure mode (Y/N)?	Safety Mechanism(s) allowing the system to prevent the failure mode from violating the safety goals (e.g. SM1, SM2)	Failure mode (diagnostic) coverage (%)	λ_{SPF}	λ_{RF}
1	Primary Bridge	Y	Incorrect data written into transmit or configuration register(s)	15%	Incorrect or no SPI Transmission	3.50	Y	Y	SM5	80%	0.000	0.105

N	O	P	Q	R	S	T	U
Potential to vilate a SG, in combination w/ one dependent (Y/N)?	Is there a safety mechanism in place to control latent faults (Y/N)?	Safety mechanism(s) allowing to prevent the failure mode from being latent ?	Failure mode (diagnostic) coverage w latent failures	λ_{SAFE}	$\lambda_{MP,L}$	$\lambda_{MPF,DP}$	Justification / Rationale
	Y	SM3	70%	0.(			

Distribution of Failure Modes = Engineering Judgment

λ (lambda) = Determined by the IC technology, distributed by area/transistor count to each block

FMEDA - SPF

Single Point Failure Rate = Failure Mode % * Block's Lambda

A	B	C	D	E	F	G	H	I	J	K	L	M
#	Block	Safety Related Element (Y/N)?	Failure Mode	Failure Mode Ratio (%)	Effect of Failure Mode	λ (FIT)	Potential to violate a Safety Goal in absence of safety mechanism (Y/N?)	Is there a safety mechanism in place to control failure mode (Y/N)?	Safety Mechanism(s) allowing the system to prevent the failure mode from violating the safety goals (e.g. SM1, SM2)	Failure mode (diagnostic) coverage (%)	λ_{SPF}	λ_{RF}
1	Primary Bridge	Y	Incorrect data written into transmit or configuration register(s)	15%	Incorrect or no SPI Transmission	3.50	Y	Y	SM5	80%	0.000	0.105

Y?

Y?

N?

N	O	P	S	T	U
Potential to vilate a SG, in combination w/ one other independent	Is there a safety mechanism in place to control latent faults (Y/N)?	Safety mechanism(s) allowing to prevent the failure mode from being latent ?	Failure mode (diagnostic) coverage w latent failures	λ_{SAFE}	$\lambda_{MP,L}$	$\lambda_{MPF,DP}$	Justification / Rationale
	Y	SM3	70%	0.000	0.126	0.294	

Equations derived from ISO 26262:2018 Part 5 Annex B

Implication: SPF > RF > MPF

FMEDA - RF

Residual Failure Rate
= Failure Mode % * Block's Lambda *
(1- Safety Mechanism Diagnostic Coverage)

A	B	C	D	E	F	G	H	I	J	K	L	M
#	Block	Safety Related Element (Y/N)?	Failure Mode	Failure Mode Ratio (%)	Effect of Failure Mode	λ (FIT)	Potential to violate a Safety Goal in absence of safety mechanism (Y/N?)	Is there a safety mechanism in place to control failure mode (Y/N)?	Safety Mechanism(s) allowing the system to prevent the failure mode from violating the safety goals (e.g. SM1, SM2)	Failure mode (diagnostic) coverage (%)	λ_{SPF}	λ_{RF}
1	Primary Bridge	Y	Incorrect data written into transmit or configuration register(s)	15%	Incorrect or no SPI Transmission	3.50	Y	Y	SM5	80%	0.000	0.105

Y?

Y?

Y?

N	O	P			S	T	U
Potential to vilate a SG, in combination w/ one other independent failure (Y/N)?	Is there a safety mechanism in place to control latent faults (Y/N)?	Safety mechanism(s) allowing to prevent the failure mode from being latent ?	Failure mode (diagnostic) coverage w latent failures	λ_{SAFE}	$\lambda_{MP,L}$	$\lambda_{MPF,DP}$	Justification / Rationale
Y	Y	SM3	70%	0.000	0.126	0.294	

FMEDA - Safe

Safe Failure Rate
= Case 1: Failure Mode % * Block's Lambda
= Case 2: Failure Mode % * Block's Lambda
*** Safety Mechanism Diagnostic Coverage**

A	B	C	D	E	F	G	H	I	J	K	L	M
#	Block	Safety Related Element (Y/N)?	Failure Mode	Failure Mode Ratio (%)	Effect of Failure Mode	λ (FIT)	Potential to violate a Safety Goal in absence of safety mechanism (Y/N?)	Is there a safety mechanism in place to control failure mode (Y/N)?	Safety Mechanism(s) allowing the system to prevent the failure mode from violating the safety goals (e.g. SM1, SM2)	Failure mode (diagnostic) coverage (%)	λ_{SPF}	λ_{RF}
1	Primary Bridge	Y	Incorrect data written into transmit or configuration register(s)	15%	Incorrect or no SPI Transmission	3.50	Y	Y	SM5	80%	0.000	0.105

Y?

N?

N	O	P	R	S	T	U
Potential to vilate a SG, in combination w/ one other independent failure (Y/N)?	Is there a safety mechanism in place to control latent faults (Y/N)?	Safety mechanism(s) allowing to prevent the failure mode from being latent ?	Failure mode (diagnostic) coverage w latent failures	λ_{SAFE}	$\lambda_{MP,L}$	$\lambda_{MPF,DP}$	Justification / Rationale
Y	Y	SM3	70%	0.000	0.126	0.294	

N?

Case 1 – No violation of Safety Goal

FMEDA - Safe

Safe Failure Rate
= Case 1: Failure Mode % * Block's Lambda
= Case 2: Failure Mode % * Block's Lambda
*** Safety Mechanism Diagnostic Coverage**

A	B	C	D	E	F	G	H	I	J	K	L	M
#	Block	Safety Related Element (Y/N)?	Failure Mode	Failure Mode Ratio (%)	Effect of Failure Mode	λ (FIT)	Potential to violate a Safety Goal in absence of safety mechanism (Y/N?)	Is there a safety mechanism in place to control failure mode (Y/N)?	Safety Mechanism(s) allowing the system to prevent the failure mode from violating the safety goals (e.g. SM1, SM2)	Failure mode (diagnostic) coverage (%)	λ_{SPF}	λ_{RF}
1	Primary Bridge	Y	Incorrect data written into transmit or configuration register(s)	15%	Incorrect or no SPI Transmission	3.50	Y	Y	SM5	80%	0.000	0.105

Y?

Y?

Y?

N	O	P	S	T	U
Potential to vilate a SG, in combination w/ one other independent failure (Y/N)?	Is there a safety mechanism in place to control latent faults (Y/N)?	Safety mechanism(s) allowing to prevent the failure mode from being latent ?	Failure mode (diagnostic) coverage w latent failures	λ_{SAFE}	$\lambda_{MP,L}$	$\lambda_{MPF,DP}$	Justification / Rationale
Y	Y	SM3	70%	0.000	0.126	0.294	

N?

Case 2 – Detected with a no multi-point failure potential

FMEDA – MPF

MPF, Detected Failure Rate
= Failure Mode % *
Block's Lambda *
Diagnostic Coverage of both Safety Mechanisms.

A	B	C	D	E	F	G	H	I	J	K	L	M
#	Block	Safety Related Element (Y/N)?	Failure Mode	Failure Mode Ratio (%)	Effect of Failure Mode	λ (FIT)	Potential to violate a Safety Goal in absence of safety mechanism (Y/N?)	Is there a safety mechanism in place to control failure mode (Y/N)?	Safety Mechanism(s) allowing the system to prevent the failure mode from violating the safety goals (e.g. SM1, SM2)	Failure mode (diagnostic) coverage (%)	λ_{SPF}	λ_{RF}
1	Primary Bridge	Y	Incorrect data written into transmit or configuration register(s)	15%	Incorrect or no SPI Transmission	3.50	Y	Y	SM5	80%	0.000	0.105

Y? Y? Y?

N	O	P		S	T	U
Potential to vilate a SG, in combination w/ one other independent failure (Y/N)?	Is there a safety mechanism in place to control latent faults (Y/N)?	Safety mechanism(s) allowing to prevent the failure mode from being latent ?	Failure mode (diagnostic) coverage w latent failures	λ_{SAFE}	$\lambda_{MP,L}$	$\lambda_{MPF,DP}$	Justification / Rationale
Y	Y	SM3	70%	0.000	0.126	0.294	

Y? Y?

FMEDA - DC

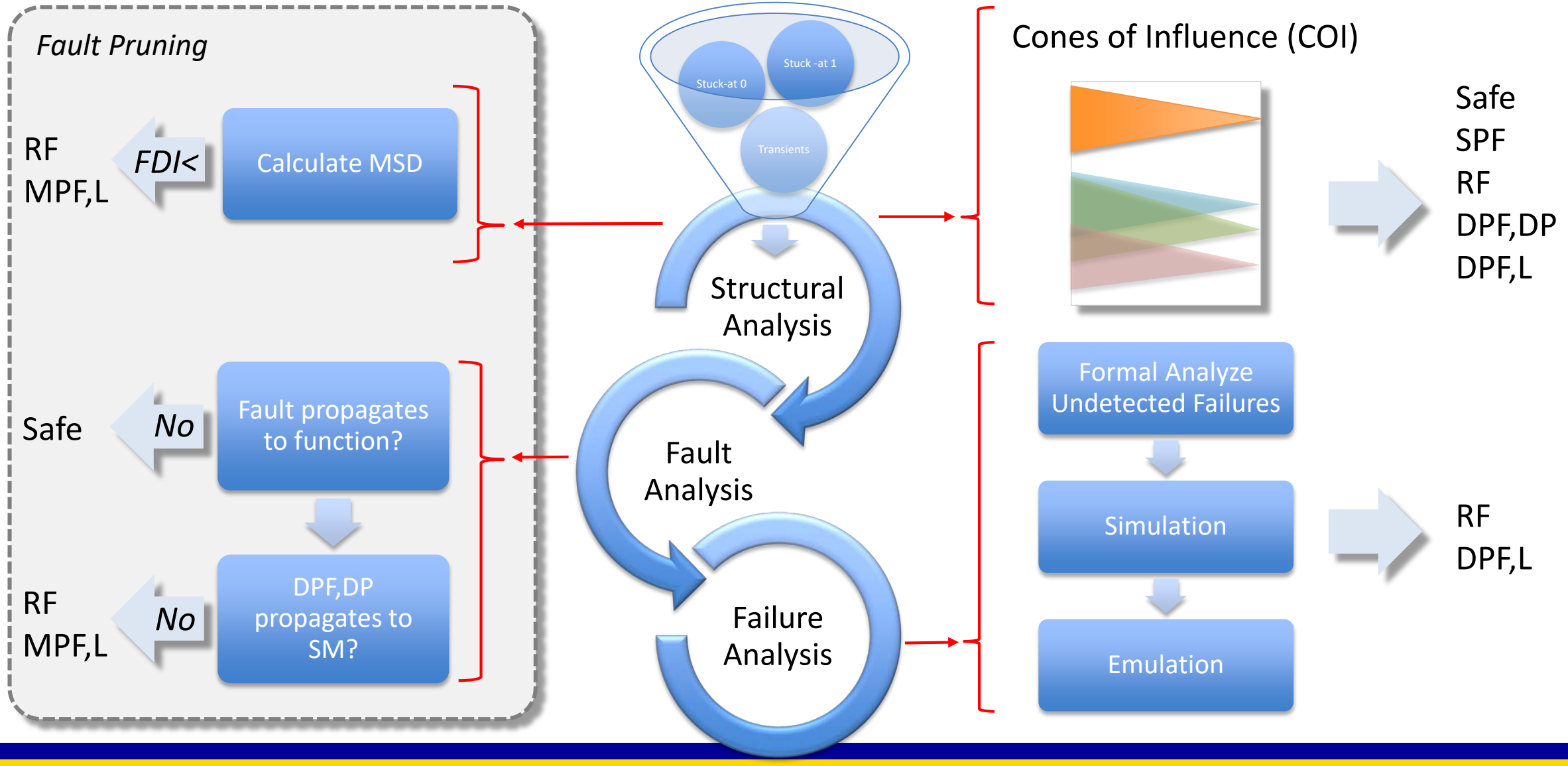
Where does Diagnostic Coverage come from?
Answer: ISO 26262 Part 5 Annex D & Part 11
OR Fault Campaign OR Expert Judgement

A	B	C	D	E	F	G	H	I	J	K	L	M
#	Block	Safety Related Element (Y/N)?	Failure Mode	Failure Mode Ratio (%)	Effect of Failure Mode	λ (FIT)	Potential to violate a Safety Goal in absence of safety mechanism (Y/N?)	Is there a safety mechanism in place to control failure mode (Y/N)?	Safety Mechanism(s) allowing the system to prevent the failure mode from violating the safety goals (e.g. SM1, SM2)	Failure mode (diagnostic) coverage (%)	λ_{SPF}	λ_{RF}
1	Primary Bridge	Y	Incorrect data written into transmit or configuration register(s)	15%	Incorrect or no SPI Transmission	3.50	Y	Y	SM5	80%	0.000	0.105

N	O	P	Q	R	S	T	U
Potential to vilate a SG, in combination w/ one other independent failure (Y/N)?	Is there a safety mechanism in place to control latent faults (Y/N)?	Safety mechanism(s) allowing to prevent the failure mode from being latent ?	Failure mode (diagnostic) coverage w latent failures	λ_{SAFE}	$\lambda_{MP,L}$	$\lambda_{MPF,DP}$	
Y	Y	SM3	70%	0.0			

Safety Mechanisms that are standard / well understood can rely solely on the standard / documented sources. Position tends to vary with customers.

Layered Fault Campaign



Increase Confidence

Failure Modes	Structural Analysis	Fault Analysis	Failure Analysis
Stuck-at 0 Stuck-at 1 Transients	*Categorize based on cones of influence*  → Safe, SPF, RF, DPF,DP, DPF,L	Fault propagates to function? — *No* → Safe	Undetectable Fault Failures? — *Yes* → RF
Stuck-at 0 Stuck-at 1 (No transients)	*Calculate Minimum Sequential Distance* *MSD > FDI* → RF, DPF,L	DPF,DP propagates to SM? — *No* → RF, DPF,L	Undetectable DPF,DP Failures? — *Yes* → DPF,L
Diagnostic Coverage	Worst Case Most Pessimistic	Better Less Pessimistic	Highest Confidence Least Pessimistic

Design Hardening

- Beyond providing/validation of Metrics, Fault Campaigns provide
  - Verification of safety mechanisms
  - Insight into improving coverage

- Need insight into where faults fall

Fault Details

Faults outside the cone of influence of any safety critical path

Safe Faults (252)

dat_i[14]
dut.i_run_bist_i
dut.s_wbspi.first_edge
dat_i[15]
miso

Safety critical path name:	***TSR-1***
Safety critical expression:	dat_o
Safety detection expression:	|p_error
Total:	**2221**

Unverified

Single-point faults (no safety mechanism) (0)

Residual fault (not covered by safety mechanism) (8)

Dual point fault (detected/perceived) (211)

Fault Id	Property	Signal Name
0	-	dut.p_wbspi.wb_dat_o[0]
1	-	dut.p_wbspi.shift.data[0]
2	-	dut.p_wbspi.shift.data[32]
3	-	dut.p_wbspi.shift.data[64]
4	-	dut.p_wbspi.shift.data[96]
5	-	dut.p_wbspi.divider[0]
6	-	dut.p_wbspi.ctrl[0]
7	-	dut.p_wbspi.scratch_reg[0]
8	-	dut.p_wbspi.ss[0]
9	-	dut.p_wbspi.clgen.clk_out
10	-	dut.p_wbspi.clgen.pos_edge

Aggregating / Mapping Coverage

FMEDA

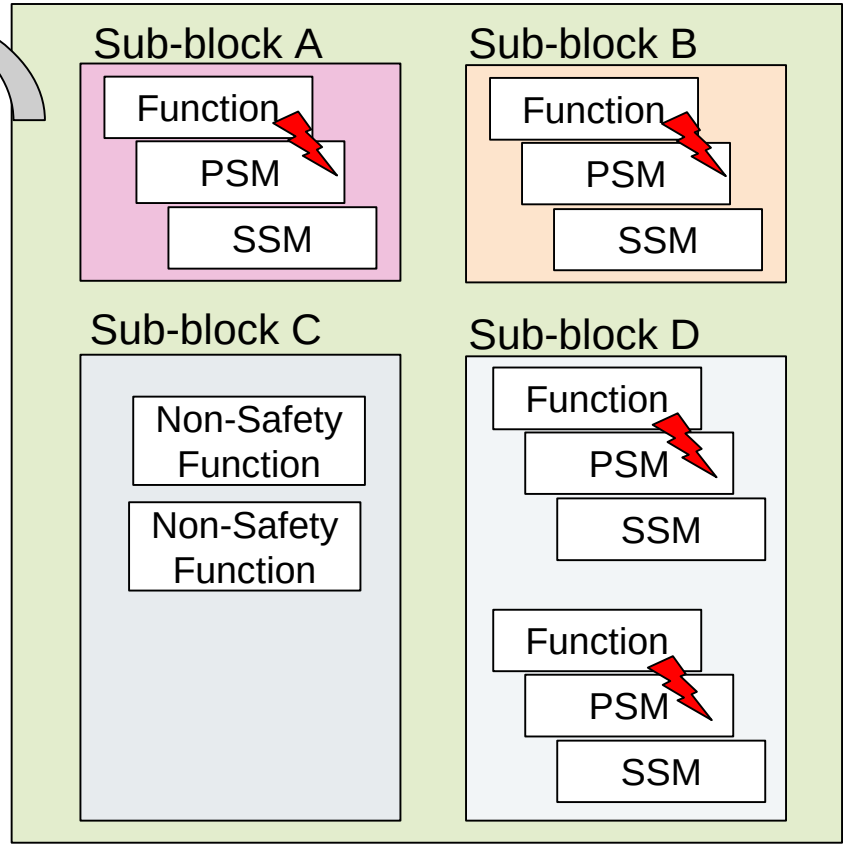
Design Block	DC of SM
Block 1	X%
Block 2	Y%
Block 3	Z%
Etc.	

FMEDA generates SPFM, LFM, & PMHF (est.)

Compare

Aggregated Diagnostic Coverage

Block 1 (Safety Related, ASIL)



Fault Campaign ran on Blocks A, B, D – individually or grouped – based on Function / SM

Campaign collects across each block, the sum of:

F_{SPF} F_{RF} F_{SAFE} $F_{DP,DET}$ $F_{DP,LAT}$

Aggregates Faults to create Diagnostic Coverage for Block 1:

$DC_{RF,Block1}$ $DC_{LF,Block1}$

Management & Tracing

- Challenges:
  - Managing the Size/Complexity of FMEDA Spreadsheet
  - Supporting internal reviews and audits
  - Supporting external assessments

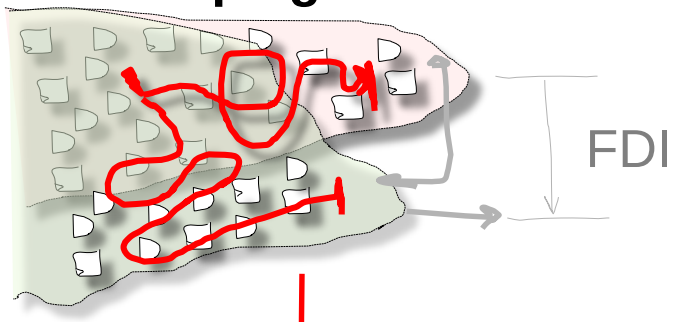
Fault campaign process (1)

FMEDA

Part Analysis (Use Blocks from Elementary Level)														Calculated or from Elementry Level			
Part (Reference)	Sub-Part (Reference)	Elementary sub-parts (Block Name) (Reference)	Elementary sub-parts (Instance name)	Safety Related Block (Y/N)?	Permanent / Transient	Memory / Logic	Failure Mode	λ (FIT) (May be from Elementry Level)	From Elementry Level	Safe Faults (%)	Safety Mechanism(s) allowing the system to prevent the failure mode from violating the safety goals (e.g. None, SM1, SM2)	Diagnostic Coverage (%)	Diagnostic Coverage from Fault Campaign (Comparison)	Permanent Logic $\lambda_{SPF} + \lambda_{RF}$	Transient Logic $\lambda_{SPF} + \lambda_{RF}$	Memory Permenant $\lambda_{SPF} + \lambda_{RF}$	Memory Transient $\lambda_{SPF} + \lambda_{RF}$
IC	example_top	spi_top	p_wbspi	Y	P	L	Permanent Fault in Block	4.476	Y		See Elementary Level			0.036	0.002	0.000	0.000
IC	example_top	spi_top	p_wbspi	Y	T	L	Transient Fault in Block X	4.476	Y		See Elementary Level			0.036	0.002	0.000	0.000
IC	example_top	spi_top	s_wbspi	Y	P	L	Permanent Fault in Block	4.476	N	100%	None	0%		0.000	0.000	0.000	0.000
IC	example_top	spi_compare	p_vs_s_spi_co	Y	P	L	Permanent Fault in Block	0.265	Y		See Elementary Level			0.000	0.000	0.000	0.000
IC	example_top	wb_comapre	p_vs_s_wb_co	Y	P	L	Permanent Fault in Block	0.468	Y		See Elementary Level			0.000	0.000	0.000	0.000
IC	example_top	reg_compare	p_vs_s_reg_co	Y	P	L	Permanent Fault in Block	0.272	Y		See Elementary Level			0.000	0.000	0.000	0.000
IC	example_top	error_handler	err_h0	Y	P	L	Permanent Fault in Block	0.034	Y		See Elementary Level			0.000	0.000	0.000	0.000
IC	example_top	error_handler	err_h1	Y	P	L	Permanent Fault in Block	0.034	Y		See Elementary Level			0.000	0.000	0.000	0.000
IC	example_top	bist_handler	bist_h	Y	P	L	Permanent Fault in Block	0.407	Y		See Elementary Level			0.000	0.000	0.000	0.000
								0.000						0.000	0.000	0.000	0.000
								0.000						0.000	0.000	0.000	0.000
								0.000						0.000	0.000	0.000	0.000
								0.000						0.000	0.000	0.000	0.000
								0.000						0.000	0.000	0.000	0.000

Safety Info

Formal Fault Campaign



Fault List

Testbenches
UCDB

Back-annotate

Tracking

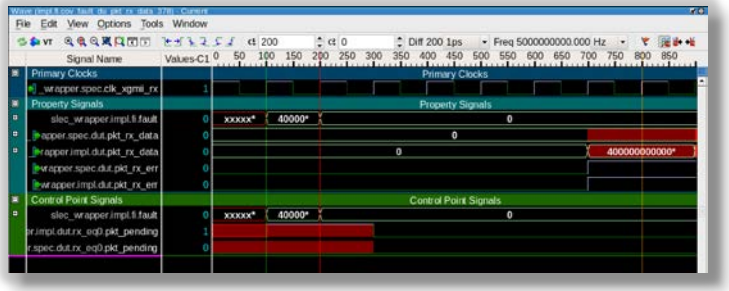
Verification Management Tracker

Sec#	Testplan Section / Coverage Link	Type	Coverage	Goal	% of Goal	Status	Weight
0	testplan	Testplan	50%	-	50%		1
1	SR-1	Testplan	50%	-	50%		1
1.1	SR-1.1	Testplan	100%	100%	100%		1
1.2	SR-1.2	Testplan	0%	100%	0%		1

Diagnostic
Coverage

UCDB

Simulation/Emulation Fault Campaign



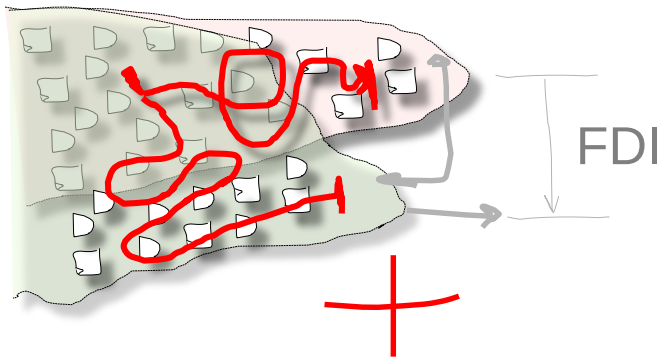
Fault campaign process (2)

Safety Definitions

Section	Title	Description	Safety Path Expression	Primary Safey Mechanism Expression	Secondary Safey Mechanism Expression	Fault Detection Time	Multi-Point Fault Detection	Link
1	SR-1	top_module						
1.1	SR-1.1	Permanent Fault leading to wrong results in Register.	dat_o & (stb && cyc)	|p_error	|s_error	5	5	TEST_ATTRIBU
1.2	SR-1.2	Permanent Fault leading to wrong results in Register.	ack & (stb && cyc)	|p_error	|s_error	5	5	TEST_ATTRIBU
1.3	SR-1.3	Permanent Fault leading to wrong results in Register.	interrupt	|p_error	|s_error	5	5	TEST_ATTRIBU
1.4	SR-1.4	Permanent Fault leading to wrong results in Register.	ss	|p_error	|s_error	5	5	TEST_ATTRIBU
1.5	SR-1.5	Permanent Fault leading to wrong results in Register.	sclk	|p_error	|s_error	5	5	TEST_ATTRIBU
1.6	SR-1.6	Permanent Fault leading to wrong results in Register.	mosi	|p_error	|s_error	5	5	TEST_ATTRIBU

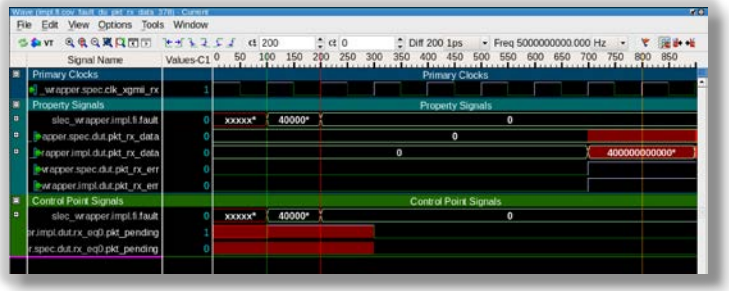
Safety Info

Fault Campaigns



Diagnostic Coverage

UCDB



Tracking

Verification Management Tracker

Sec#	Testplan Section / Coverage Link	Type	Coverage	Goal	% of Goal	Status	Weight
0	testplan	Testplan	50%	-	50%		1
1	SR-1	Testplan	50%	-	50%		1
1.1	SR-1.1	Testplan	100%	100%	100%		1
1.2	SR-1.2	Testplan	0%	100%	0%		1

FMEDA

Part Analysis (Use Blocks from Elementary Level) | Calculated or from Elementry Level

Part (Reference)	Sub-Part (Reference)	Elementary sub-parts (Block Name) (Reference)	Elementary sub-parts (instance name)	Safety Related Block (Y/N)?	Permanent / Transient	Memory / Logic	Failure Mode	λ (FIT) (May be from Elementry Level)	From Elementry Level	Safe Faults (%)	Safety Mechanism(s) allowing the system to prevent the failure mode from violating the safety goals (e.g. None, SM1, SM2)	Diagnostic Coverage (%)	Diagnostic Coverage from Fault Campaign (Comparison)	Permanent Logic $\lambda_{SPF} + \lambda_{RF}$	Transient Logic $\lambda_{SPF} + \lambda_{RF}$	Memory Permenant $\lambda_{SPF} + \lambda_{RF}$	Memory Transient $\lambda_{SPF} + \lambda_{RF}$
IC	example_top	spi_top	p_wbspi	Y	P	L	Permanent Fault in Block	4.476	Y		See Elementary Level			0.036	0.002	0.000	0.000
IC	example_top	spi_top	p_wbspi	Y	T	L	Transient Fault in Block X	4.476	Y		See Elementary Level			0.036	0.002	0.000	0.000
IC	example_top	spi_top	s_wbspi	Y	P	L	Permanent Fault in Block	4.476	N	100%	None	0%		0.000	0.000	0.000	0.000
IC	example_top	spi_compare	p_vs_s_spi_co	Y	P	L	Permanent Fault in Block	0.265	Y		See Elementary Level			0.000	0.000	0.000	0.000
IC	example_top	wb_comapre	p_vs_s_wb_co	Y	P	L	Permanent Fault in Block	0.468	Y		See Elementary Level			0.000	0.000	0.000	0.000
IC	example_top	reg_compare	p_vs_s_reg_co	Y	P	L	Permanent Fault in Block	0.272	Y		See Elementary Level			0.000	0.000	0.000	0.000
IC	example_top	error_handler	err_h0	Y	P	L	Permanent Fault in Block	0.034	Y		See Elementary Level			0.000	0.000	0.000	0.000

Template + Annotate

Summary

- Qualitative Analysis with a FMEA/FTA & Quantitative Analysis with a FMEDA are standard practices
- FMEDA is a key document to allows integrators of IC/IP to understand functional safety metrics
  - Especially important when considering configuration / feature options
- Connecting information from Fault Injection Campaigns to the FMEDA
  - Validates early predictions of Diagnostic Coverage and Hardware Architectural Metrics
  - With challenging architectures, the only means to determine coverage
- Fault Injection Campaigns serve as verification of safety mechanisms

Break

2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

How Formal Reduces Fault Analysis for ISO 26262

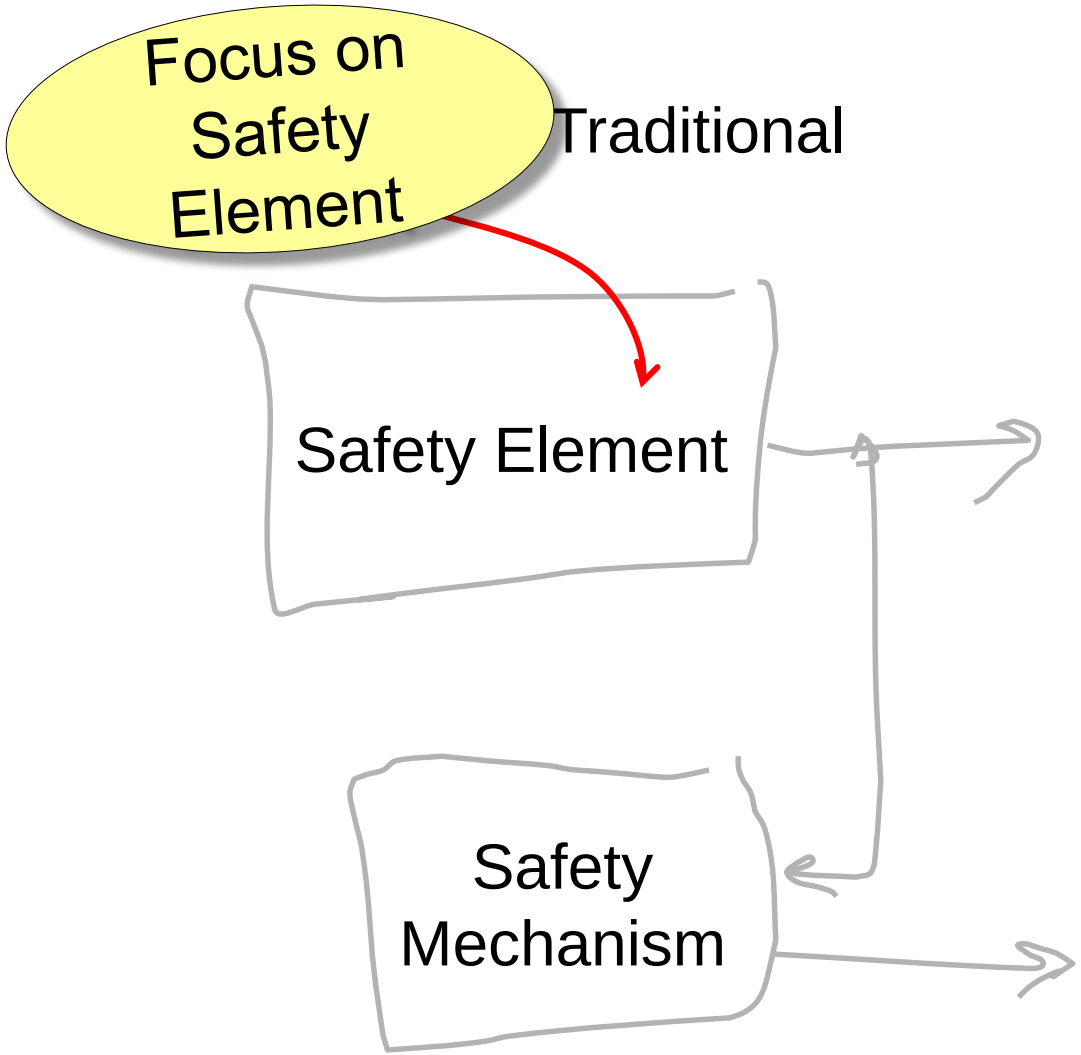
Doug Smith

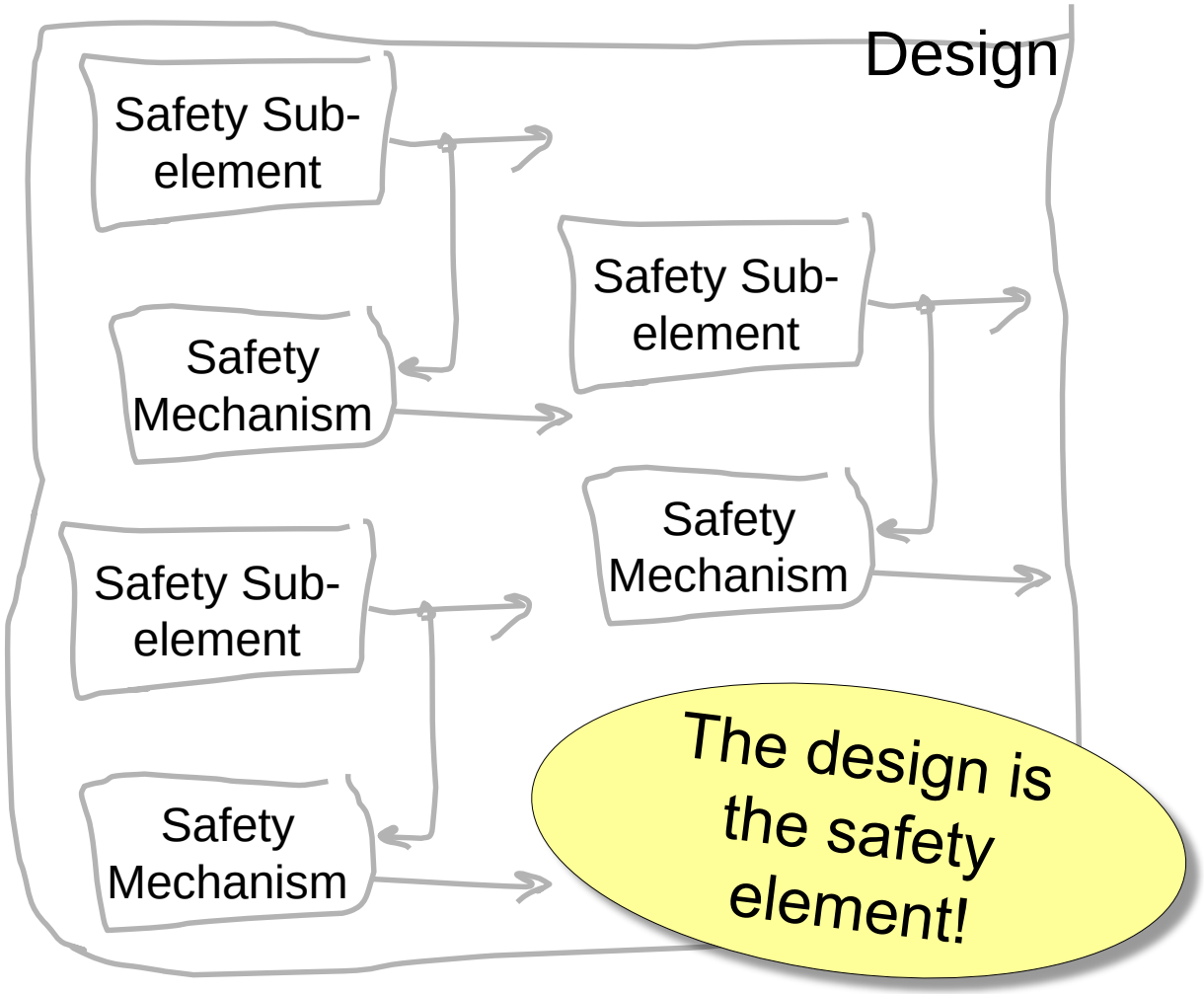
Doug_Smith@mentor.com

Verification Consultant
Mentor Consulting

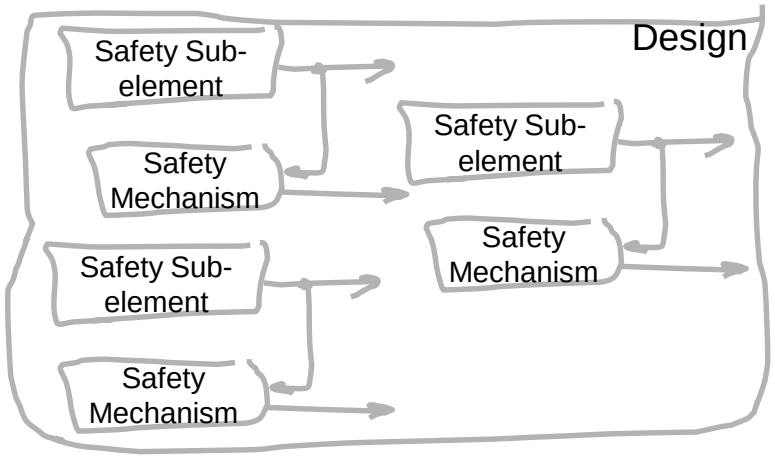
Mentor Graphics®
A Siemens Business

Safety on semiconductors





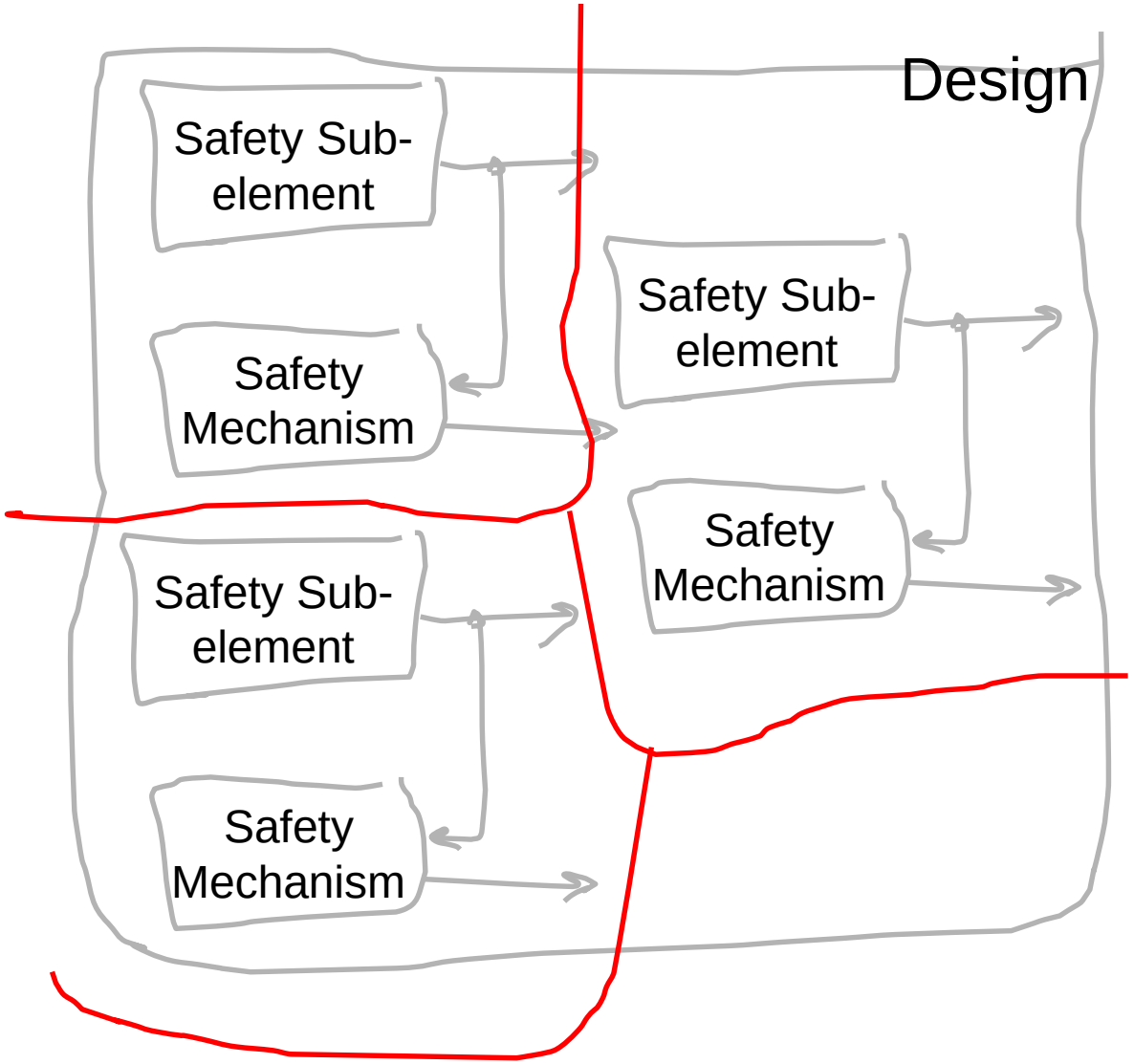
ICs are harder



- Potentially lots of
  - Safety critical functions
  - Safety mechanisms
  - Secondary safety mechanisms
- Large designs → thousands of random faults to inject!
- How to categorize faults shared between shared logic?
- Need tests that allow faults to propagate and be detected
- Large simulation time to test software safety mechanisms
- May have large fault detection time intervals

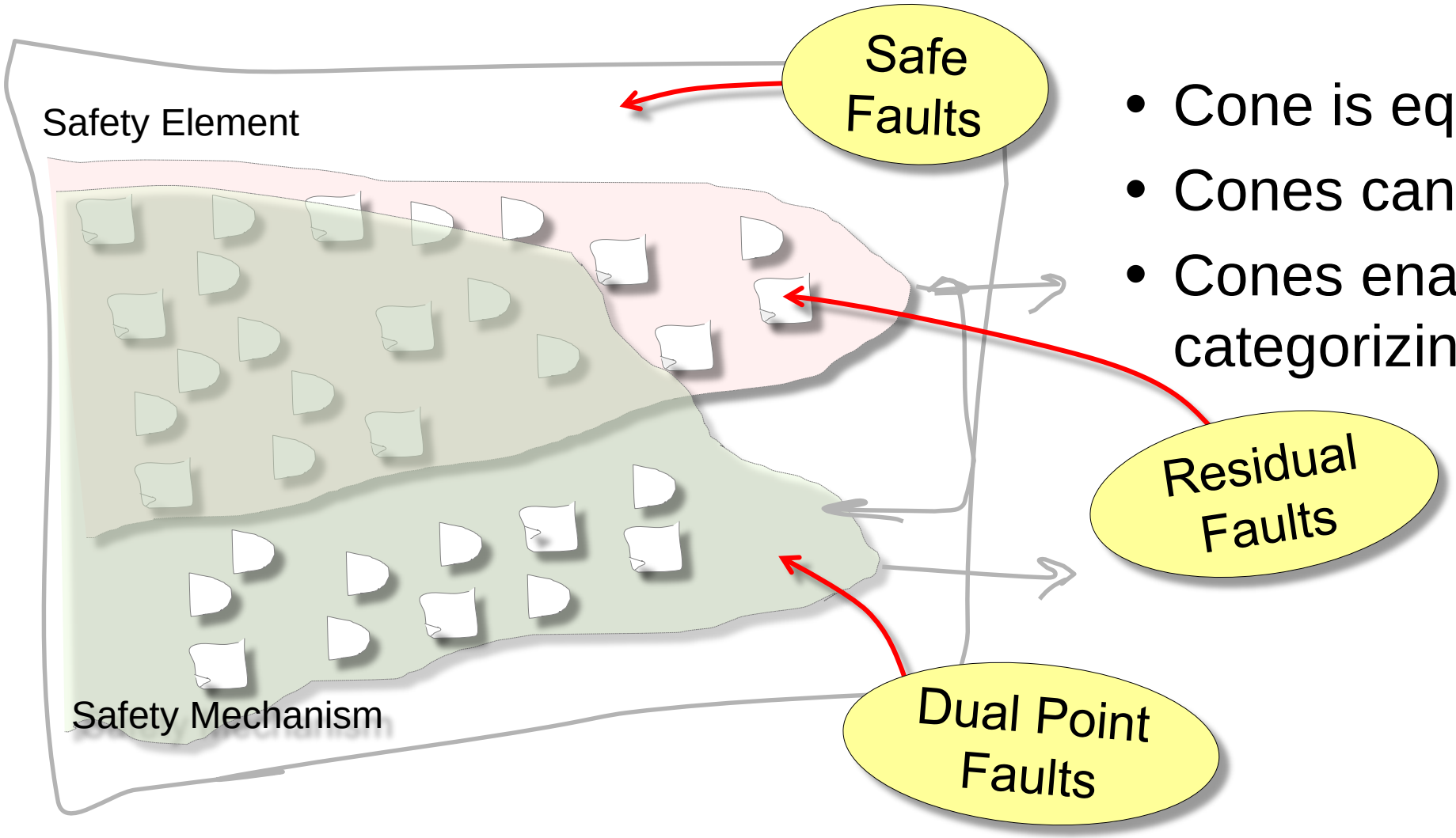
Try breaking up the problem!

- Not allowed ☹
- Must show independence with Dependent Fault Analysis (DFA)



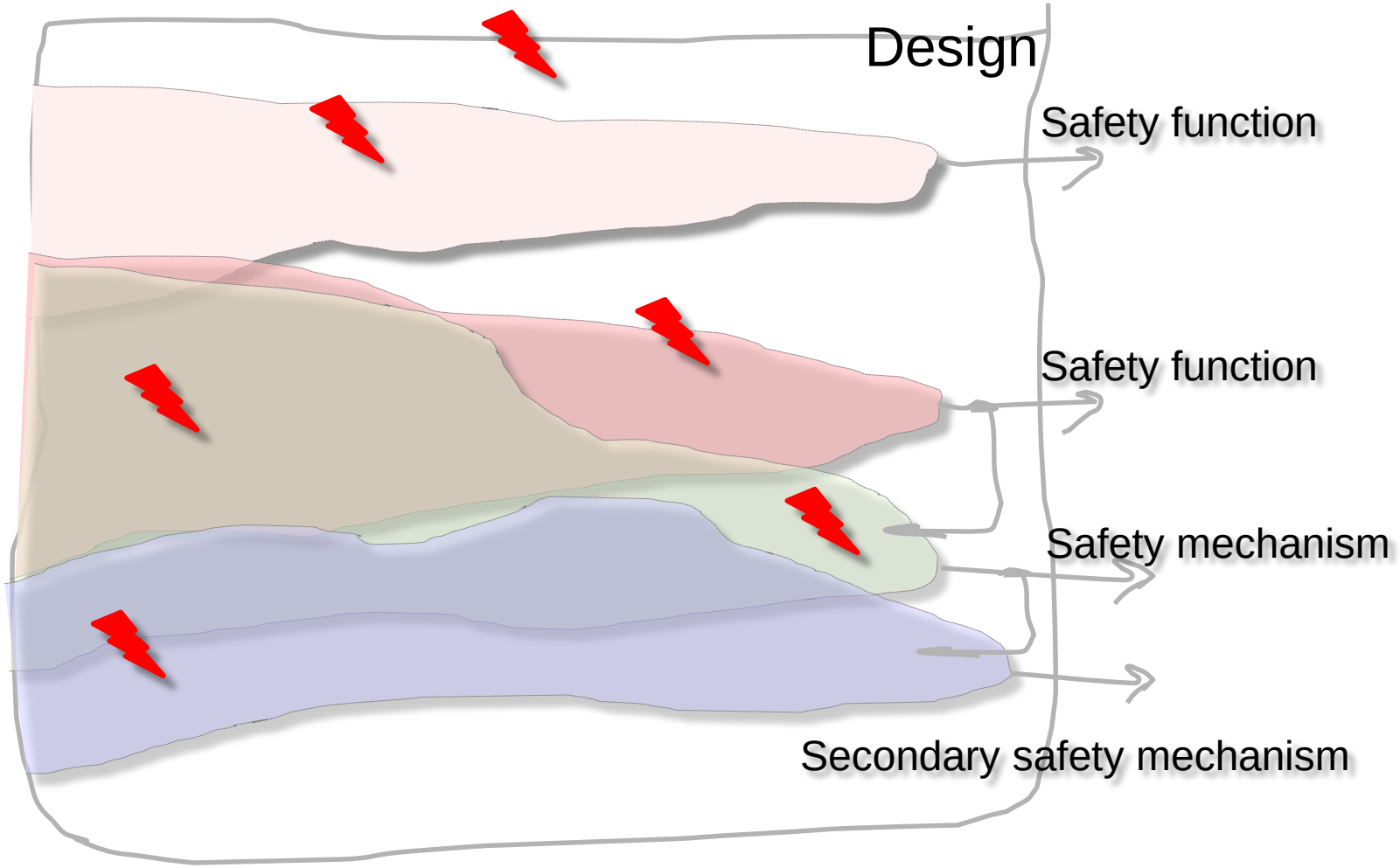
Solution – Formal cones of influence

Formal cone of influence (COI)



- Cone is equivalent to DFA
- Cones can overlap
- Cones enable quick fault categorizing

COI fault analysis

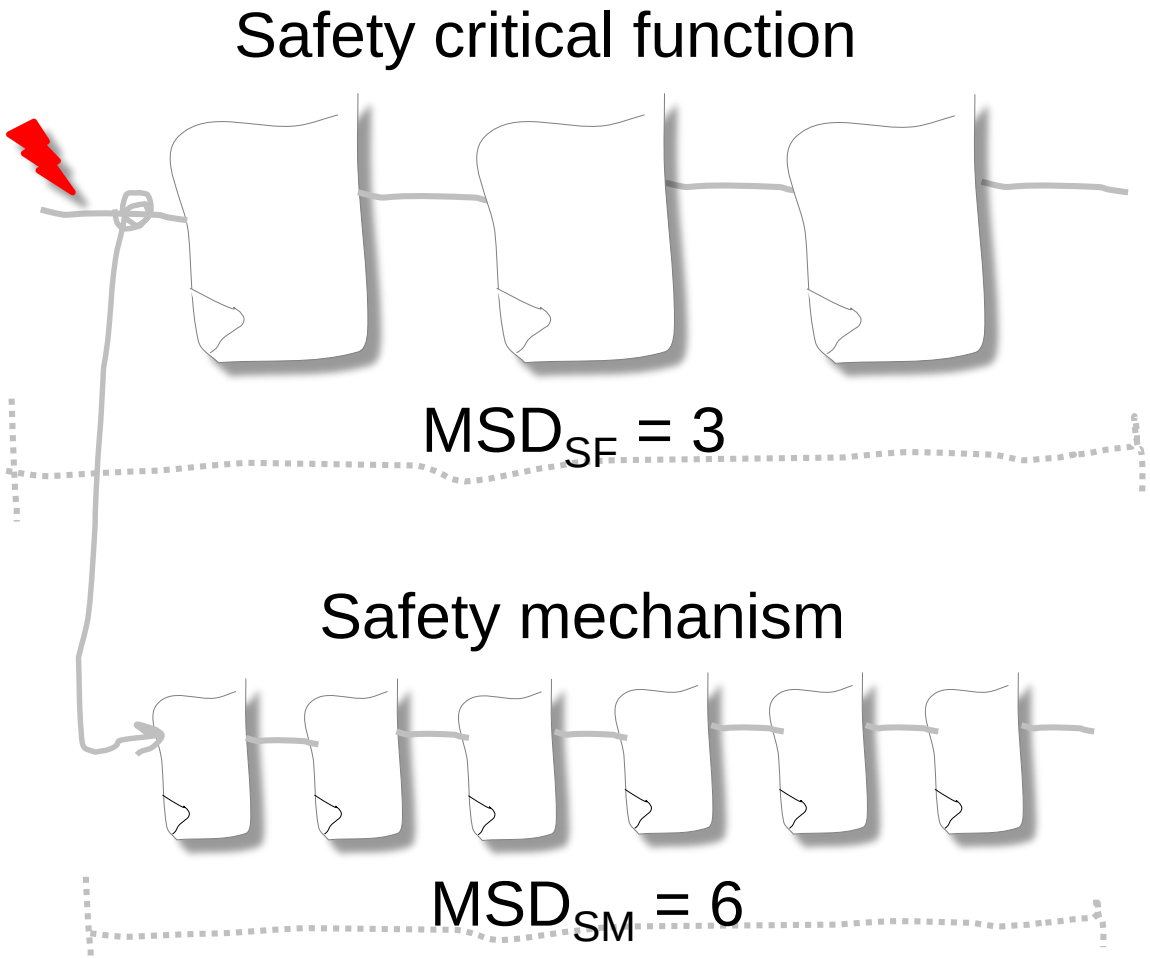


- ✓ Safe fault
- ✓ Single-point fault
- ✓ Residual fault
- ✓ Dual-point fault in safety function
- ✓ Dual-point fault in safety mechanism
- ✓ Latent fault

Approximate, but quick!

Minimum sequential distance (M

Not FTTI!
From HW spec



- Fault Detection Interval (FDI)
- Violation if
  $MSD_{SM} – MSD_{SF} > FDI$

E.g.,

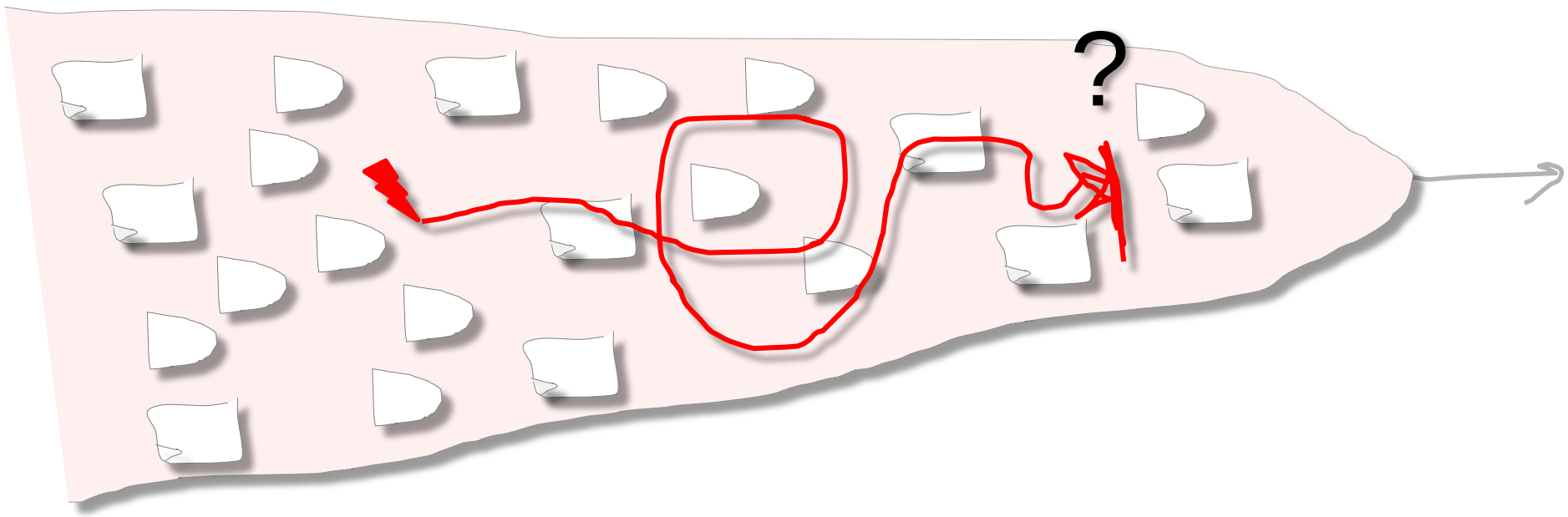
FDI = 2

6 - 3 > 2

Residual or
Latent Fault

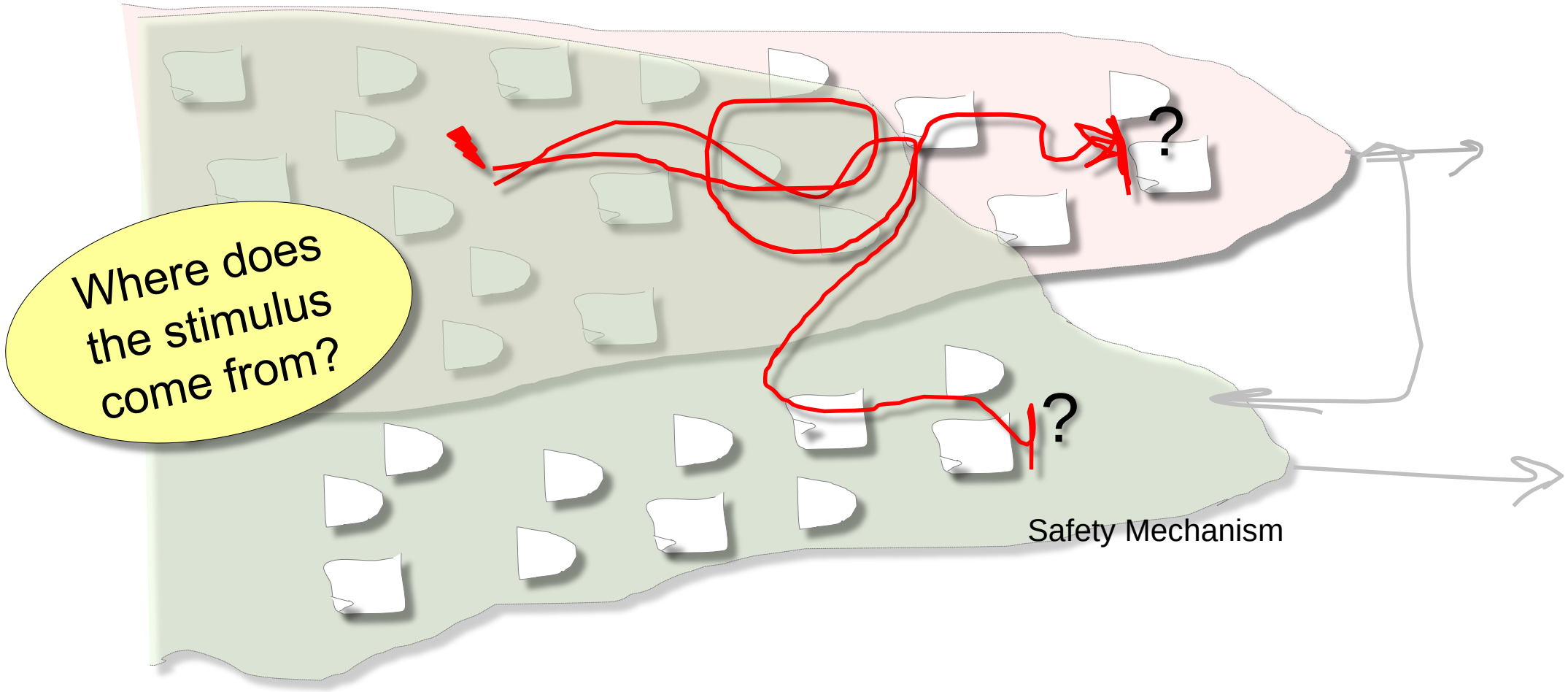
∴, Too long to propagate….
safety goal violation!

Safety function fault propagation



No propagation -> Safe fault!

Safety mechanism fault propagation



No propagation -> undetectable fault!

Traditional formal

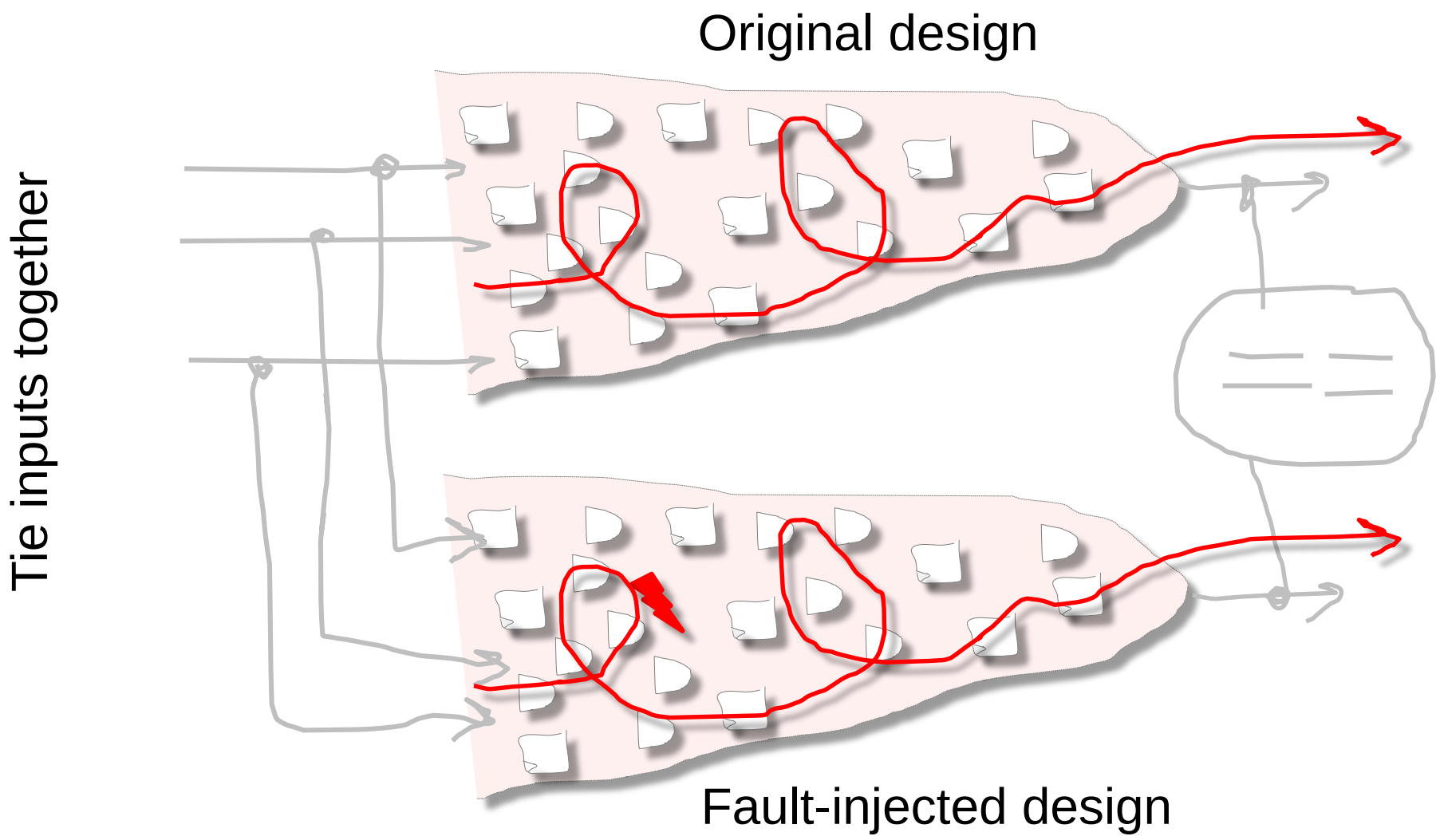
- Input constraints and assumptions

```
asm_drive_data          : assume property ( pkt_val |-> pkt_data == data  );
asm_pkt_stable          : assume property ( pkt_val |-> $stable(packet)   );
asm_payload_stable      : assume property ( pkt_val |-> $stable(payload)  );
asm_pkt_kind_stable     : assume property ( pkt_val |-> $stable(pkt_type) );
```

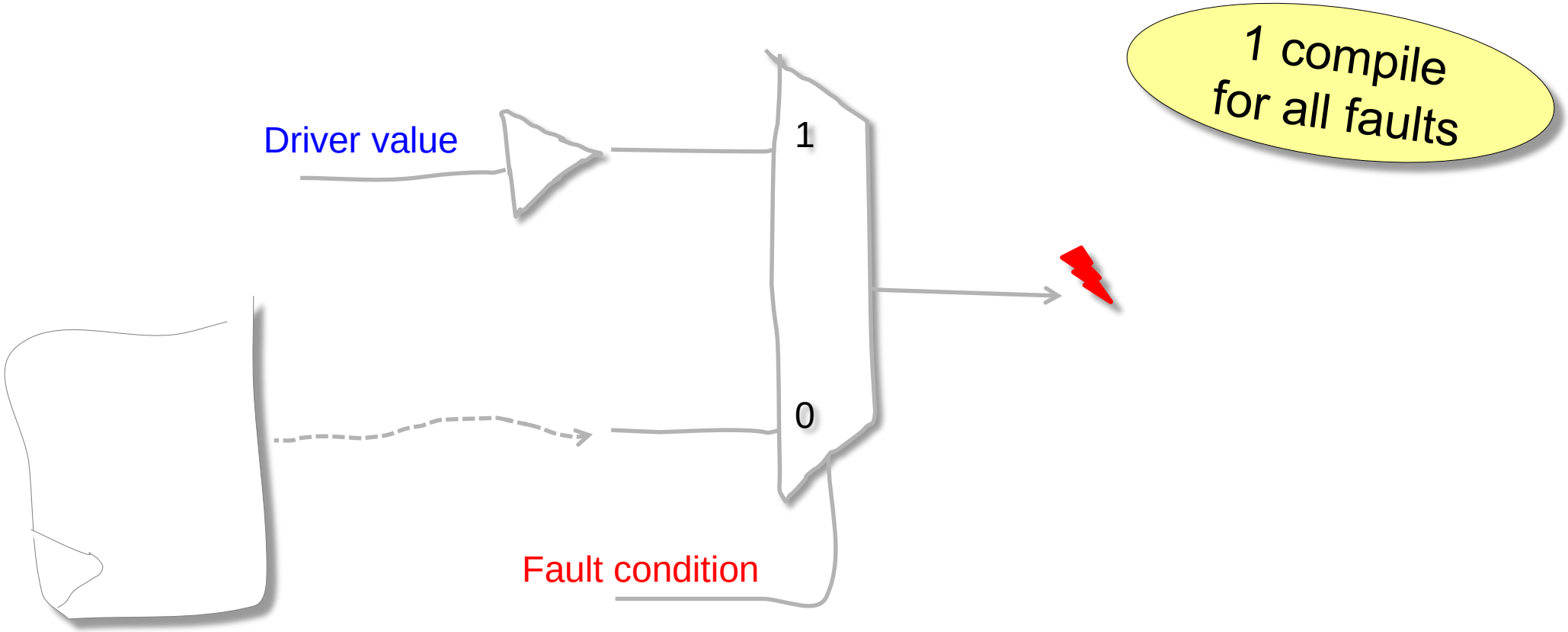
- Issues
  - Need input requirements
  - Labor intensive
  - Not automated
  - Typically incomplete – formal tries everything!

Solution - SLEC

Sequential Equivalency Checking



Injecting faults




```
netlist cutpoint dut.reg_o -cond { fault[100] } -driver 1'b0
```

Formal fault injector

Formal control point

```
module fault_injector(...);
   default clocking cb @($global_clock); endclocking
   bit [N:0] fault = '0;

   asm_single_point_fault: assume property ( $onehot0(fault)          );
   asm_fault_stuckat:      assume property ( fault |=> $stable(fault) );
```

- Conditional cutpoint

```
netlist cutpoint {impl.dut.tx_data_fifo0.fifo0.genblk2.mem0.rdata[2]} \
                 -cond {impl.fi.fault[1]} -driver 1'b0
```

- SLEC target

```
slec map  spec.safecheck.safety0 \
          impl.safecheck.safety0 -cond { impl.safecheck.fi.fault[1] }
```

Parallel fault analysis

```
module fault_injector(...);
    default clocking cb @($global_
    bit [N:0] fault  = '0;

    asm_single_point_fault: assume pr
    asm_fault_stuckat:      assume pro
```

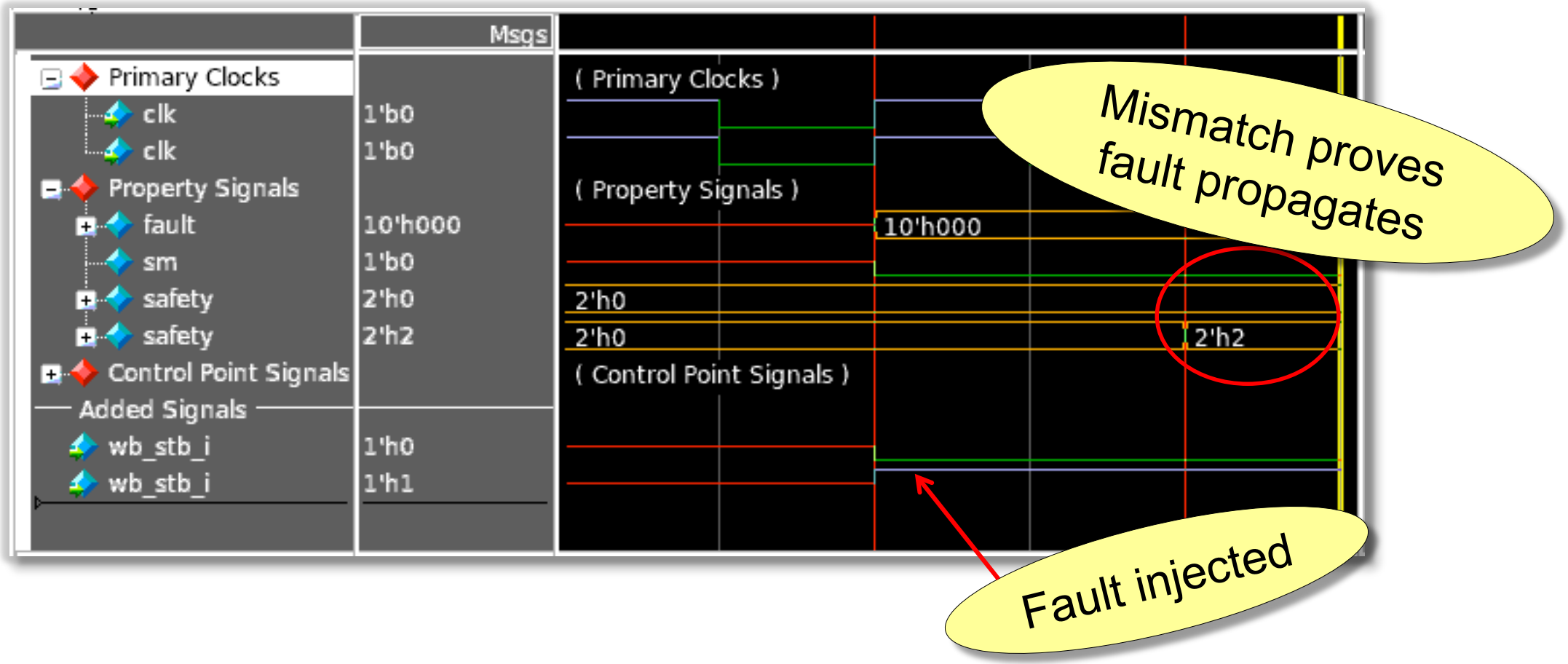
```
slec map -cond { fault[0] } …
slec map -cond { fault[1] } …
slec map -cond { fault[2] } …
slec map -cond { fault[3] } …
slec map -cond { fault[4] } …
slec map -cond { fault[5] } …
slec map -cond { fault[6] } …
slec map -cond { fault[7] } …
[8] } …
[9] } …
[10] } …
[11] } …
[12] } …
…
```

```
slec map   spec.safecheck.safety0 \
           impl.safecheck.safety0 -cond { impl.safecheck.fi.fault[1] }
```

- One compile
- Thousands of parallel fault targets analyzed by formal

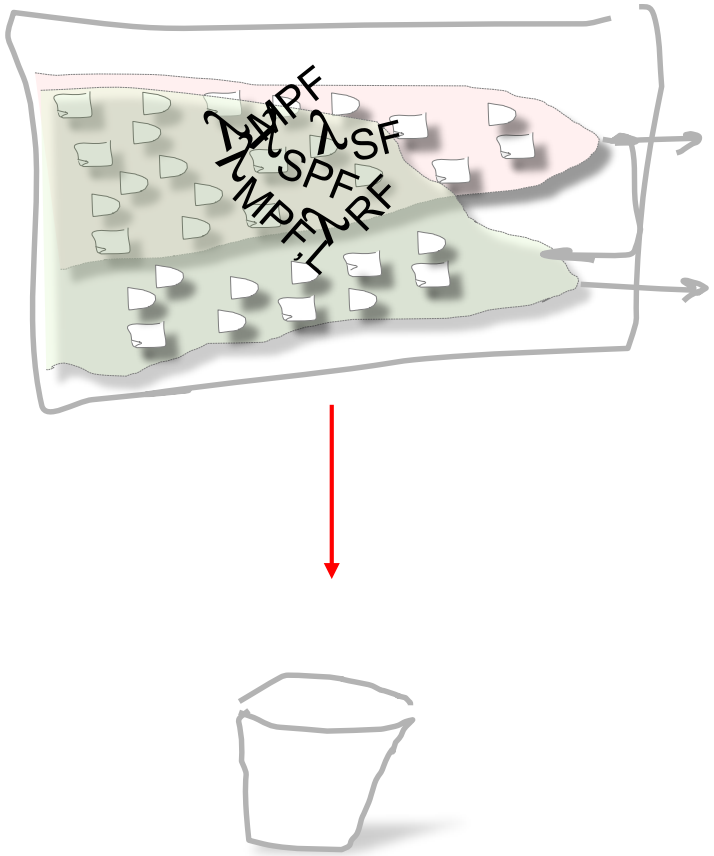
Proving a fault propagates

SLEC



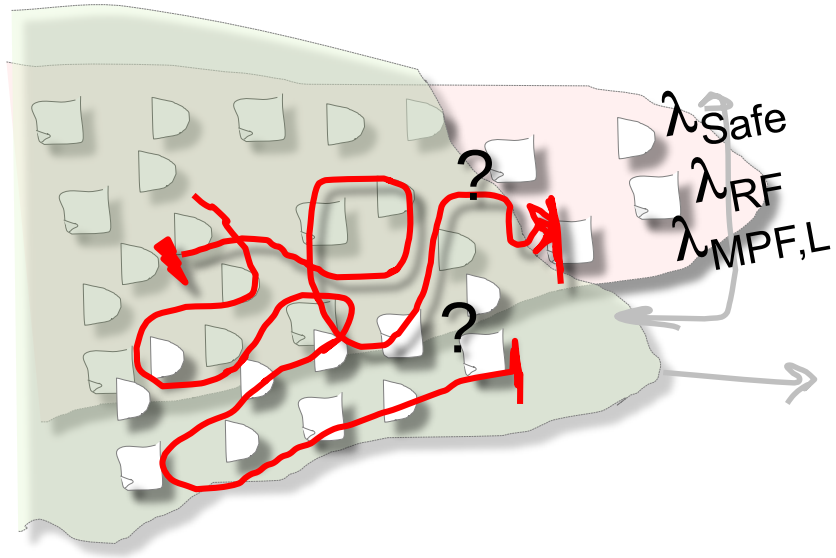
First steps …

Structural analysis



- Quick and easy
- But do all faults really propagate?

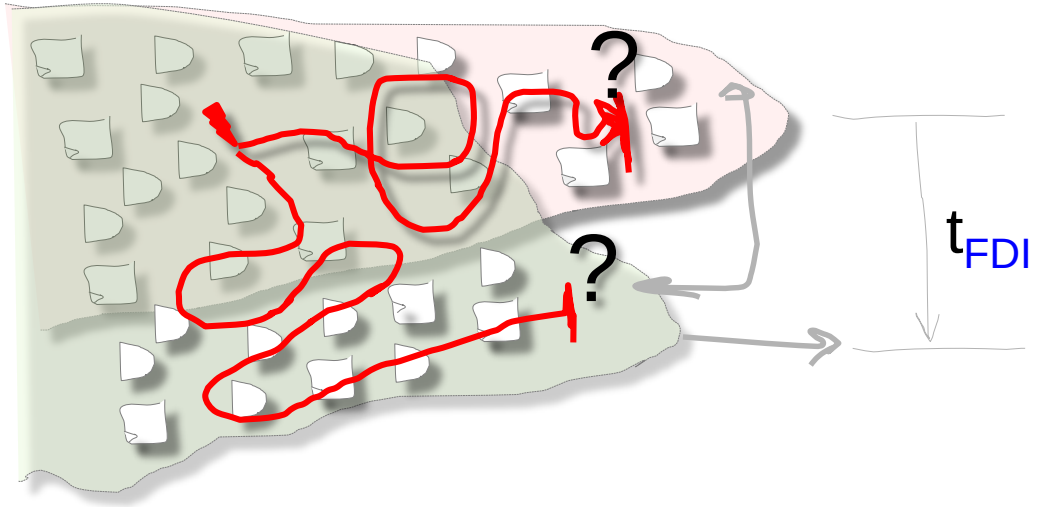
Fault analysis



Need failure analysis …

- Simultaneous propagation to output and safety mechanism?
  - Within time window?

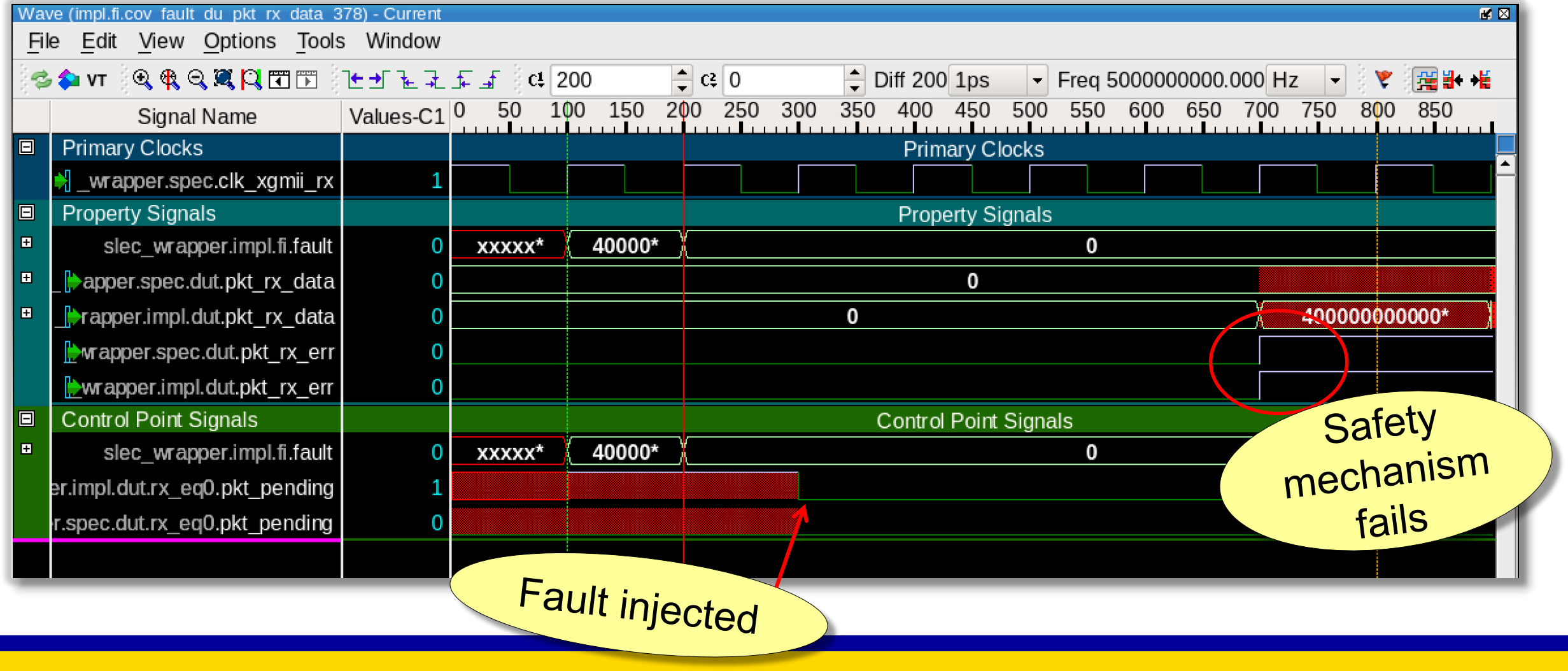
Need further refinement!



Failure analysis

```
// Find residual faults
cover property ( fault && seq_fault_propagates |-> seq_fault_not_detected[*FDI] );

// Find latent faults
cover property ( fault && seq_psm_fault_propagates |->
                                        seq_ssm_fault_not_detected[*MPFDI] );
```

Example undetected failure

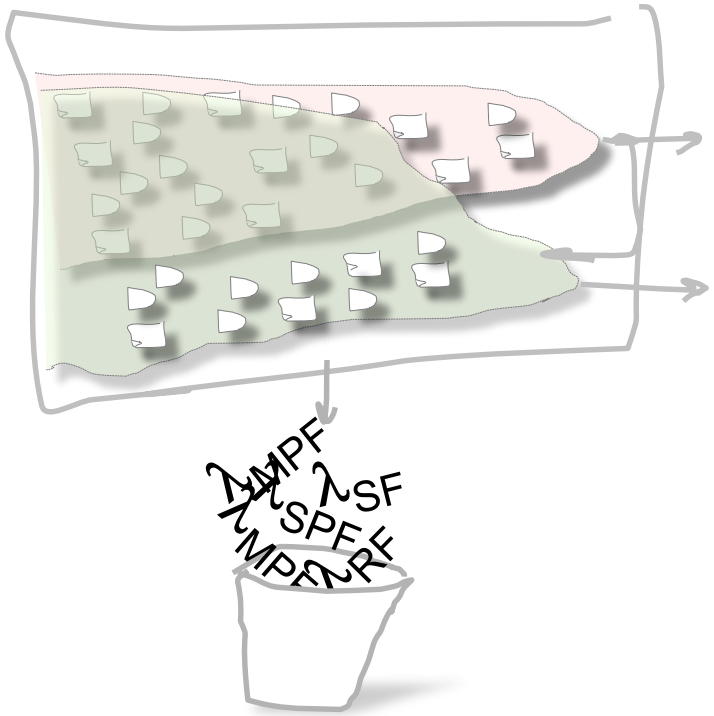


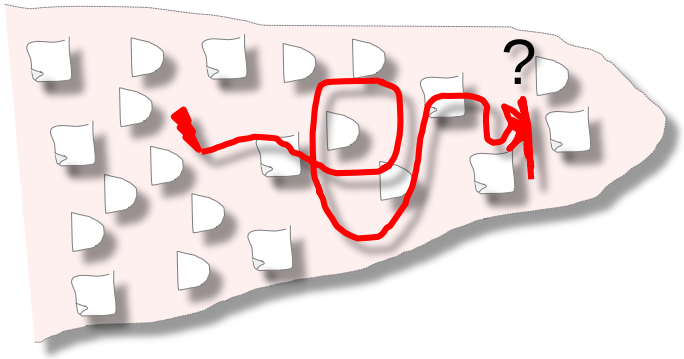
Building confidence

Structural analysis

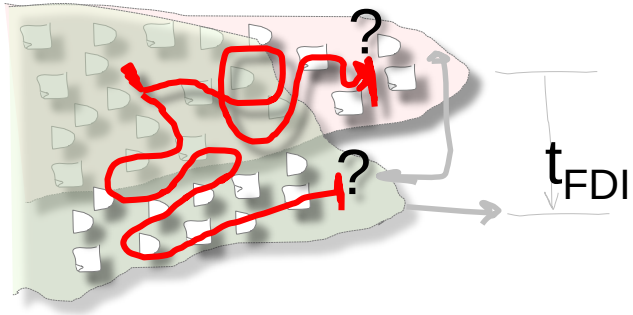
Fault analysis

Failure analysis





λ_{Safe}
λ_{RF}
$\lambda_{MPF,L}$



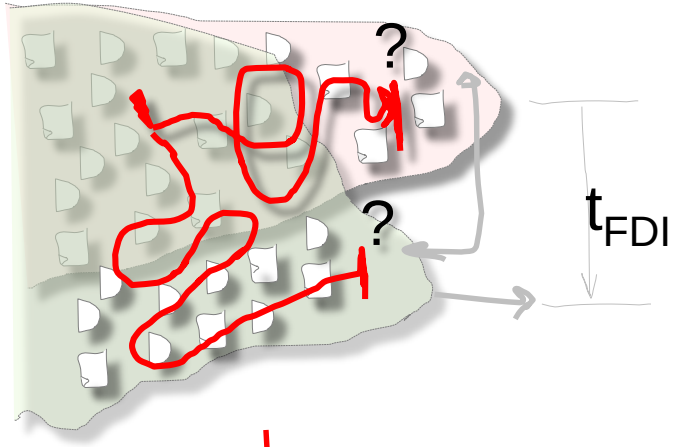
λ_{Safe}
λ_{RF}
$\lambda_{MPF,L}$

Least Confidence

Most

Diagnostic coverage

- DC = % of safety element covered by safety mechanism



λ_{RF}

$\lambda_{MPF,L}$

N_{RF} – # residual faults

$N_{MPF,L}$ – # latent faults

Residual

$DC_{RF} = 1 - (N_{RF} / N_{All})$

Latent

$DC_{MPF,L} = 1 - (N_{MPF,L} / N_{All})$

A range for diagnostic cover

Potential RF

Potential Latent

	Structural analysis		Fault analysis		Failure analysis	
	Unverified	Verified	Unverified	Verified	Unverified	Verified
Safe	286	286	0	299	0	301
Residual	8	0	0	12	0	17
Dual-point in Safey Function	219	0	215	0	134	84
Dual-point in Safety Mechanism	2013	0	1704	28	1554	132
Latent	0	0	0	296	0	307
$DC_{Residual}$	91.0% - 99.7%		91.0% - 99.5%		94.0% - 99.3%	
DC_{Latent}	20.3% - 100%		20.8% - 88.2%		26.3% - 87.8%	

Continuous refinement

Example report

SAFECHECK

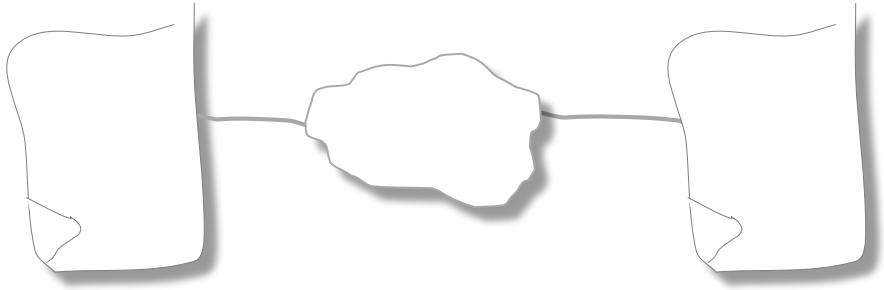
- Project
- Fault Summary
- Fault Details
- Transcript

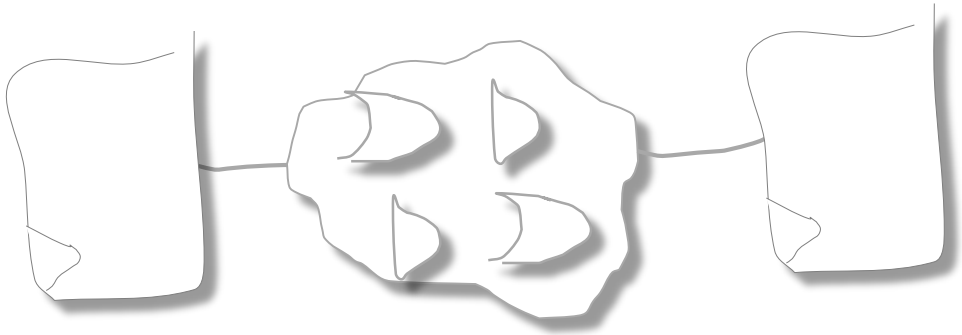
All Faults (All Safety Critical Paths)

Fault Type	Previous Analysis		Current Analysis	
	Unverified	Verified	Unverified	Verified
Safe faults (outside cone of influence)	0	287	0	287
Safe faults (fault detected by a safety mechanism)	0	0	0	120
Single-point faults (no safety mechanism)	0	0	0	0
Residual fault (not covered by safety mechanism)	8	0	8	0
Dual point fault (detected/perceived) in safety function	202	0	192	0
Dual point fault (detected/perceived) in the safety mechanism	2035	0	1841	0
Dual point fault latent	0	0	0	84
Subtotal	***2245***	***287***	***2041***	***491***
Total	**2532**		**2532**	
Number of randomly sampled faults	**2245 (88.7%)**			
Design bits	**2245 (unsafe) / 2532 (all)**			
Residual Diagnostic Coverage	**92.10% - 100.00%**			
Latent Diagnostic Coverage	**23.97% - 96.68%**			

RTL or gates?

- Formal can run on gates, but …





- RTL more likely pessimistic
- Gates likely mask faults

$$DC_{RTL} = N_{RF\text{-}RTL} / N_{RTL}$$

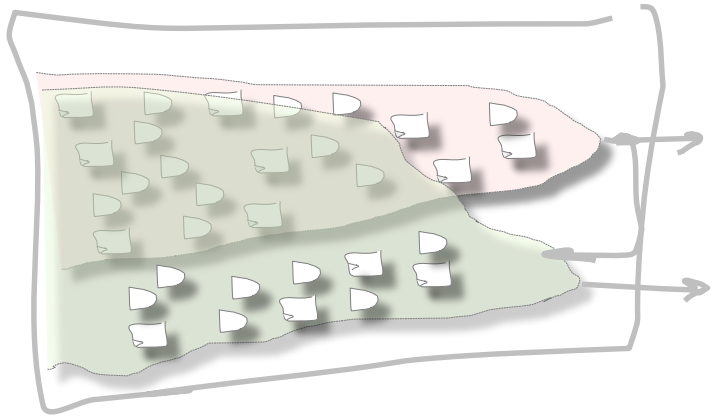
$$DC_{Gates} = N_{RF\text{-}Gates} / N_{Gates}$$

$$N_{RTL} < N_{Gates}$$

$$\therefore,\ DC_{Gates} > DC_{RTL}$$

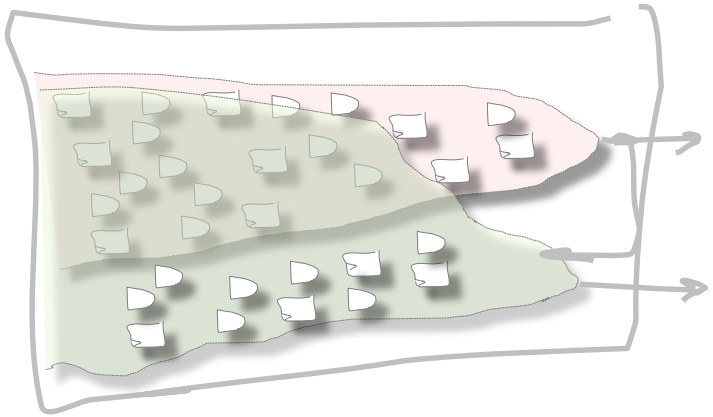
RTL to gates equivalency

RTL Structural analysis



$DC_{Residual}$	91% - 99.7%
DC_{Latent}	20% - 100%

Gates Structural analysis



$DC_{Residual}$	94% - 97%
DC_{Latent}	18% - 98%

- If RTL more pessimistic, gates are unnecessary …

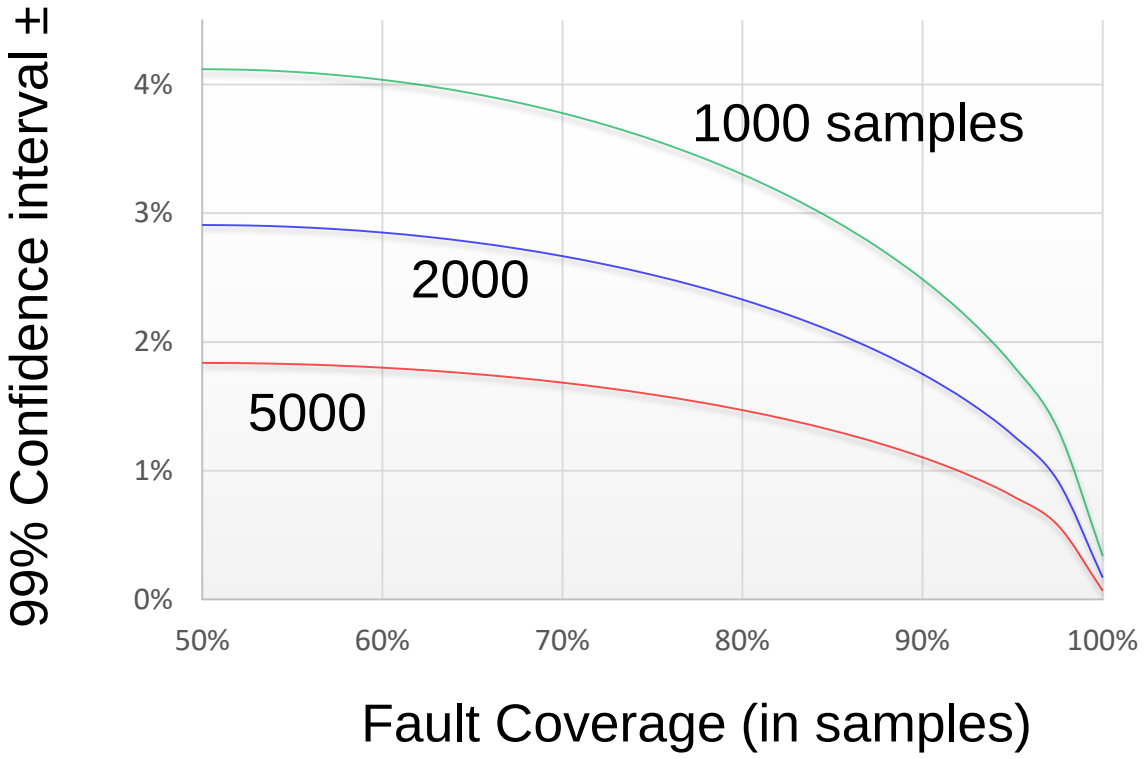
Potential limitations using formal …

- Large number of formal targets
- Long formal run times
- Large number of inconclusives
- Results biased towards formal friendly designs and design areas

Random sampling

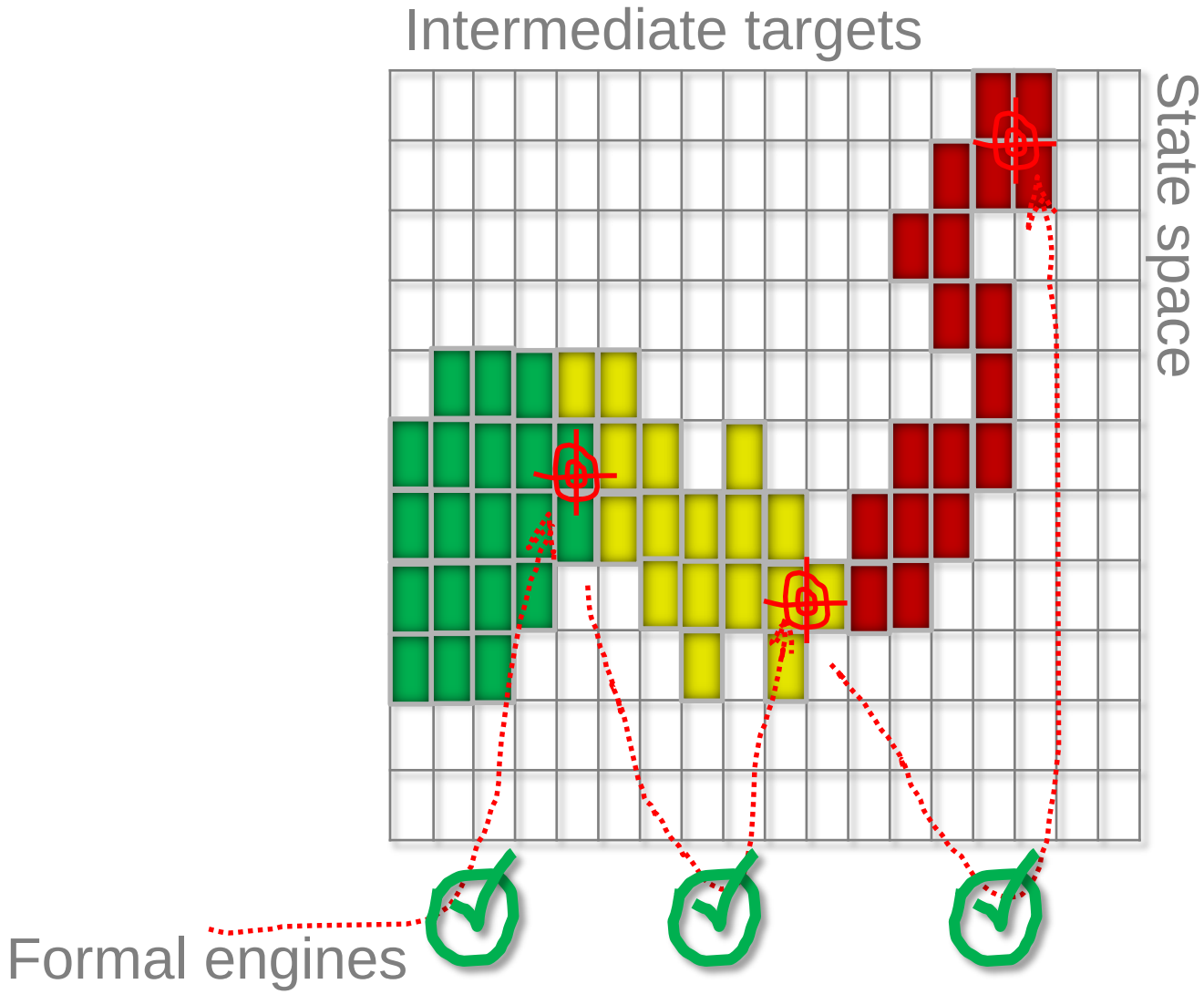
- Confidence interval
  - Allows picking random samples
- Solves
  - Large numbers of formal targets
  - Large numbers of inconclusives
  - Unmanageable results

$$CI_{99\%} = c \pm \frac{3.38}{n}\sqrt{1 + 0.59nc(1-c)}$$

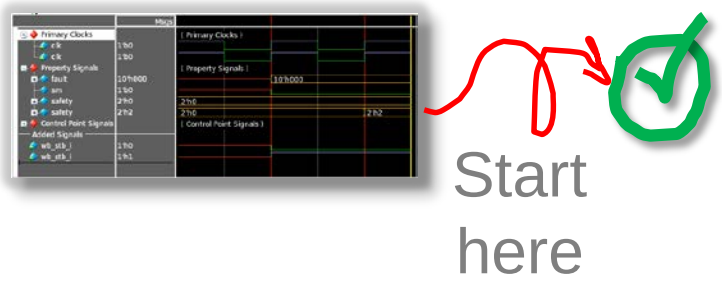


Agrawal & Kata, D&T 1990

Goal posting



- Possibilities
  - Write temp targets
  - Automatic goal-posting formal engines
  - Seed formal with waveforms
    - Find activity around faults



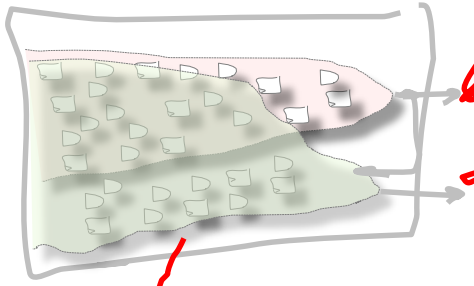
Automated flow

Automatic properties & constraints!

Safety Definitions

Section	Title	Description	Safety Path Expression	Primary Safey Mechanism Expression	Secondary Safey Mechanism Expression	Fault Detection Time	Multi-Point Fault Detection	Link
1	SR-1	top_module						
1.1	SR-1.1	Permanent Fault leading to wrong results in Register.	dat_o & (stb && cyc)	|p_error	|s_error	5	5	TEST_ATTRIBU
1.2	SR-1.2	Permanent Fault leading to wrong results in Register.	ack & (stb && cyc)	|p_error	|s_error	5	5	TEST_ATTRIBU
1.3	SR-1.3	Permanent Fault leading to wrong results in Register.	interrupt	|p_error	|s_error	5	5	TEST_ATTRIBU
1.4	SR-1.4	Permanent Fault leading to wrong results in Register.	ss	|p_error	|s_error	5	5	TEST_ATTRIBU
1.5	SR-1.5	Permanent Fault leading to wrong results in Register.	sclk	|p_error	|s_error	5	5	TEST_ATTRIBU
1.6	SR-1.6	Permanent Fault leading to wrong results in Register.	mosi	|p_error	|s_error	5	5	TEST_ATTRIBU

Structural analysis



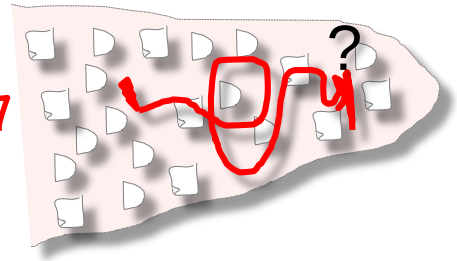
```
netlist cutpoint -cond ...
```

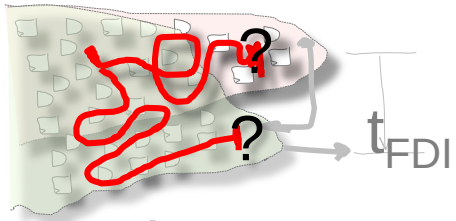
```
dut.top.shift.en
dut.abp1.psel
...
```

```
cover property ( ... );
```

```
module fault_injector(...);
```

Fault analysis

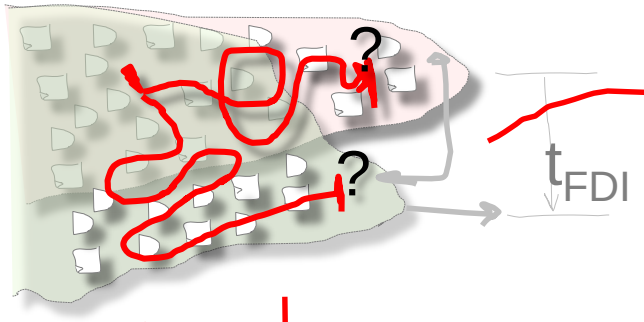




Failure analysis

Handoff to simulation and emulation

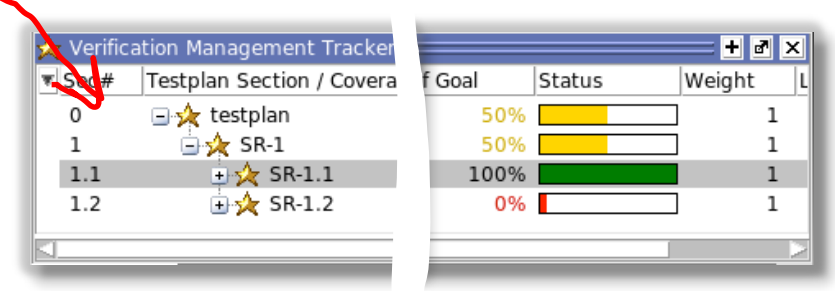
Formal Fault Campaign



Fault lists

```
dut.top.shift.en
dut.abp1.psel
...
```

UCDB and Tracking



Testbenches

```
module zi_replay_vlog;
  initial begin
    #1;
    force spi_top.wb_rst_i = 1'b1;
    ...
```

Fault injection logic

```
initial begin
   uvm_hdl_force( signal, 1'b0 );
   ...
```

Automated fault injection for simulation

Activity Analyzer

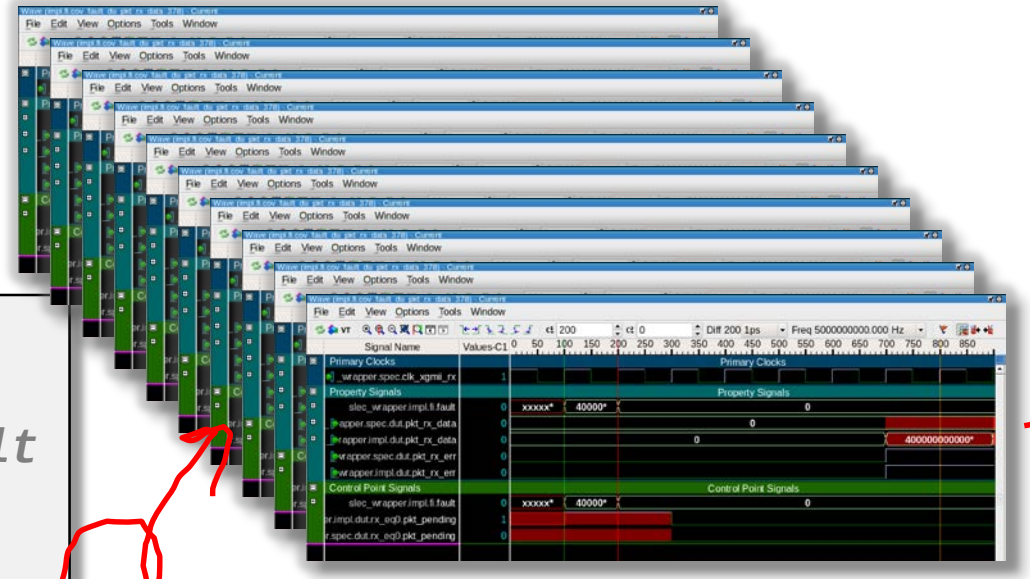
Fault lists

```
dut.top.shift.en
dut.abp1.psel
...
```

Fault injection logic

```
module fault_injector;
 initial begin
   // Read list + pick random fault
   ...
   uvm_hdl_force( signal, 1'b0 );
endmodule
bind testbench fault_injector fi();
```

Regression simulations



```
vsim +FAULT_LIST=… \
     +FAILURE_MODE=… \
     +FAULT_ID=… \
     ...
```

Testbench checks results

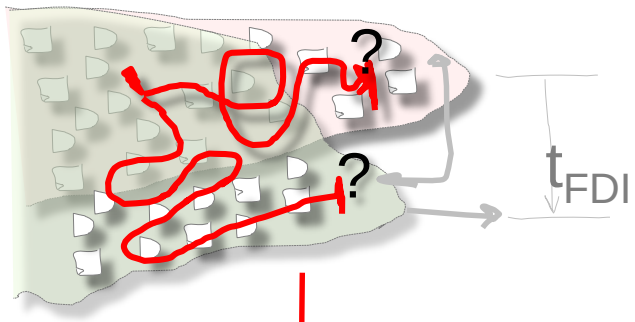
Fault campaign from the top down

FMEDA

Part Analysis (Use Blocks from Elementary Level)													Calculated or from Elementry Level			
Part (Reference)	Sub-Part (Reference)	Elementary sub-parts (Block Name) (Reference)	Elementary sub-parts (Instance name)	Safety Related Block (Y/N)?	Permanent / Transient	Memory / Logic	Failure Mode	λ (FIT) (May be from Elementry Level)	From Elementry Level	Safe Faults (%)	Safety Mechanism(s) allowing the system to prevent the failure mode from violating the safety goals (e.g. None, SM1, SM2)	Diagnostic Coverage (%)	Diagnostic Coverage from Fault Campaign (Comparison)	Permanent Logic $\lambda_{SPF} + \lambda_{RF}$	Transient Logic $\lambda_{SPF} + \lambda_{RF}$	Memory Permenant $\lambda_{SPF} + \lambda_{RF}$	Memory Transient $\lambda_{SPF} + \lambda_{RF}$
IC	example_top	spi_top	p_wbspi	Y	P	L	Permanent Fault in Block	4.476	Y		See Elementary Level			0.036	0.002	0.000	0.000
IC	example_top	spi_top	p_wbspi	Y	T	L	Transient Fault in Block X	4.476	Y		See Elementary Level			0.036	0.002	0.000	0.000
IC	example_top	spi_top	s_wbspi	Y	P	L	Permanent Fault in Block	4.476	N	100%	None	0%		0.000	0.000	0.000	0.000
IC	example_top	spi_compare	p_vs_s_spi_co	Y	P	L	Permanent Fault in Block	0.265	Y		See Elementary Level			0.000	0.000	0.000	0.000
IC	example_top	wb_comapre	p_vs_s_wb_co	Y	P	L	Permanent Fault in Block	0.468	Y		See Elementary Level			0.000	0.000	0.000	0.000
IC	example_top	reg_compare	p_vs_s_reg_co	Y	P	L	Permanent Fault in Block	0.272	Y		See Elementary Level			0.000	0.000	0.000	0.000
IC	example_top	error_handler	err_h0	Y	P	L	Permanent Fault in Block	0.034	Y		See Elementary Level			0.000	0.000	0.000	0.000
IC	example_top	error_handler	err_h1	Y	P	L	Permanent Fault in Block	0.034	Y		See Elementary Level			0.000	0.000	0.000	0.000
IC	example_top	bist_handler	bist_h	Y	P	L	Permanent Fault in Block	0.407	Y		See Elementary Level			0.000	0.000	0.000	0.000
								0.000						0.000	0.000	0.000	0.000
								0.000						0.000	0.000	0.000	0.000
								0.000						0.000	0.000	0.000	0.000
								0.000						0.000	0.000	0.000	0.000
								0.000						0.000	0.000	0.000	0.000
								0.000						0.000	0.000	0.000	0.000

Safety info

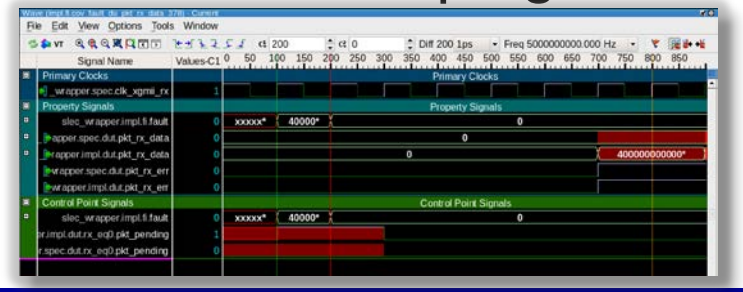
Formal Fault Campaign



Fault list
Reports

Testbenches
UCDB

Simulation/Emulation Fault Campaign



Diagnostic coverage

UCDB

Back-annotate

Tracking

Verification Management Tracker

Sec#	Testplan Section / Coverage Link	Type	Coverage	Goal	% of Goal	Status	Weight
0	testplan	Testplan	50%	-	50%		1
1	SR-1	Testplan	50%	-	50%		1
1.1	SR-1.1	Testplan	100%	100%	100%		1
1.2	SR-1.2	Testplan	0%	100%	0%		1

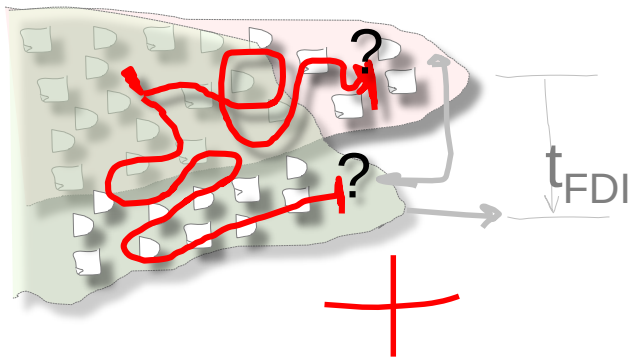
Fault campaign from the bottom up

Safety Definitions

Section	Title	Description	Safety Path Expression	Primary Safey Mechanism Expression	Secondary Safey Mechanism Expression	Fault Detection Time	Multi-Point Fault Detection	Link
1	SR-1	top_module						
1.1	SR-1.1	Permanent Fault leading to wrong results in Register.	dat_o & (stb && cyc)	|p_error	|s_error	5	5	TEST_ATTRIBU
1.2	SR-1.2	Permanent Fault leading to wrong results in Register.	ack & (stb && cyc)	|p_error	|s_error	5	5	TEST_ATTRIBU
1.3	SR-1.3	Permanent Fault leading to wrong results in Register.	interrupt	|p_error	|s_error	5	5	TEST_ATTRIBU
1.4	SR-1.4	Permanent Fault leading to wrong results in Register.	ss	|p_error	|s_error	5	5	TEST_ATTRIBU
1.5	SR-1.5	Permanent Fault leading to wrong results in Register.	sclk	|p_error	|s_error	5	5	TEST_ATTRIBU
1.6	SR-1.6	Permanent Fault leading to wrong results in Register.	mosi	|p_error	|s_error	5	5	TEST_ATTRIBU

Safety info

Fault Campaign



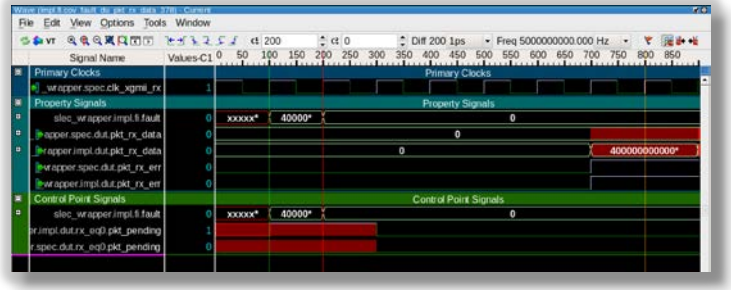
Tracking

Verification Management Tracker

Sec#	Testplan Section / Coverage Link	Type	Coverage	Goal	% of Goal	Status	Weight
0	testplan	Testplan	50%	-	50%		1
1	SR-1	Testplan	50%	-	50%		1
1.1	SR-1.1	Testplan	100%	100%	100%		1
1.2	SR-1.2	Testplan	0%	100%	0%		1

Diagnostic coverage

UCDB



FMEDA

Part Analysis (Use Blocks from Elementary Level) — Calculated or from Elementry Level

Part (Reference)	Sub-Part (Reference)	Elementary sub-parts (Block Name) (Reference)	Elementary sub-parts (instance name)	Safety Related Block (Y/N)?	Permanent / Transient	Memory / Logic	Failure Mode	λ (FIT) (May be from Elementry Level)	From Elementry Level	Safe Faults (%)	Safety Mechanism(s) allowing the system to prevent the failure mode from violating the safety goals (e.g. None, SM1, SM2)	Diagnostic Coverage (%)	Diagnostic Coverage from Fault Campaign (Comparison)	Permanent Logic $\lambda_{SPF} + \lambda_{RF}$	Transient Logic $\lambda_{SPF} + \lambda_{RF}$	Memory Permenant $\lambda_{SPF} + \lambda_{RF}$	Memory Transient $\lambda_{SPF} + \lambda_{RF}$
IC	example_top	spi_top	p_wbspi	Y	P	L	Permanent Fault in Block	4.476	Y		See Elementary Level			0.036	0.002	0.000	0.000
IC	example_top	spi_top	p_wbspi	Y	T	L	Transient Fault in Block X	4.476	Y		See Elementary Level			0.036	0.002	0.000	0.000
IC	example_top	spi_top	s_wbspi	Y	P	L	Permanent Fault in Block	4.476	N	100%	None	0%		0.000	0.000	0.000	0.000
IC	example_top	spi_compare	p_vs_s_spi_co	Y	P	L	Permanent Fault in Block	0.265	Y		See Elementary Level			0.000	0.000	0.000	0.000
IC	example_top	wb_comapre	p_vs_s_wb_co	Y	P	L	Permanent Fault in Block	0.468	Y		See Elementary Level			0.000	0.000	0.000	0.000
IC	example_top	reg_compare	p_vs_s_reg_co	Y	P	L	Permanent Fault in Block	0.272	Y		See Elementary Level			0.000	0.000	0.000	0.000
IC	example_top	error_handler	err_h0	Y	P	L	Permanent Fault in Block	0.034	Y		See Elementary Level			0.000	0.000	0.000	0.000

Template + back-annotate

Summary

- Formal provides …
  - Quick and easy fault categorization for worst-best case DC
  - No environment setup required – no testcases
  - High-level of confidence in results – can’t beat a proof!
  - Ties in with simulation and emulation
  - A great front-end for the entire fault campaign process

2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

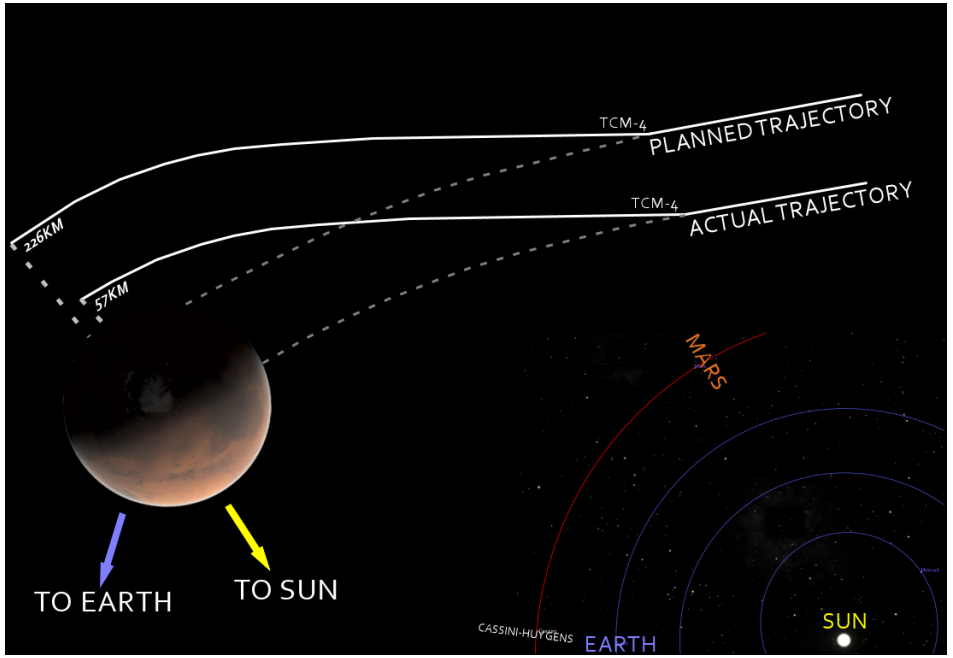
Requirement Tracing in the ISO26262 World

Charles Battikha (chuck_battikha@mentor.com)

Mentor®
A Siemens Business

Why Requirements matter…

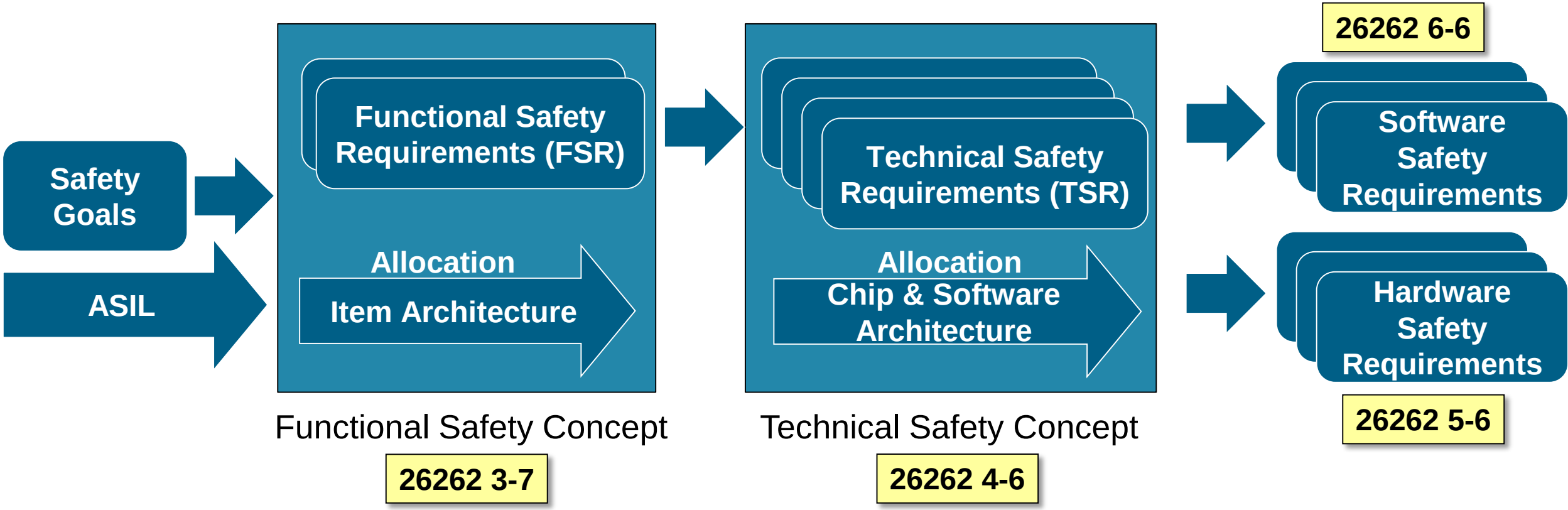
- Example: NASA's Mars Climate Orbiter
  - Sent crashing into Mars by NASA
  - The Orbiter spoke to NASA in metric… But the engineers on the ground were replying in non-metric English



"What is being designed, built, and verified is based on requirements and thus ***intended***"

Safety Requirements

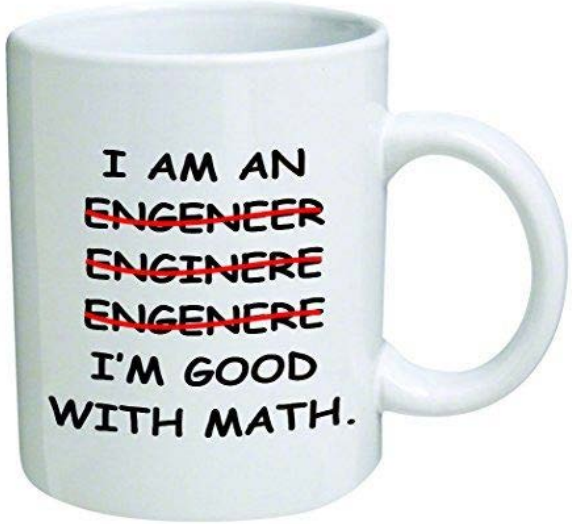
Safety Goal met? = Complete List + Each True



Achieving ASIL rating means meeting all requirements

Why are requirements hard to write?

- Human Language is inherently vague and imprecise
- Relying on engineer's writing skills….
- Trouble separating WHATs from HOWs
  - Desire to jump into the details…
- Believe spending time writing requirement will cause delays
  - Good enough…
  - Let's get on with it…



Requirements: Common Problems

- Errors of Omission
  - What was intended, was not actually stated; Important information left out
- Errors of Commission
  - Information is wrong; Information is contradictory
- Errors of Clarity
  - Requirements stated in ways that lead to confusion, misunderstanding
  - Creation of assumptions
- Errors of Understanding
  - Ambiguous, words get in the way
  - Each person internalizes and applies their own definitions

Writing Safety Requirements

26262 8-6

ASIL A/B

- Natural language
- Informal notation

ASIL C/D

- Semi-formal notation (syntax defined)
- Formal notations (syntax & semantics defined)

Increase Effort
Increase Rigor

Reduced Risk

Defining Requirements

26262 8-6

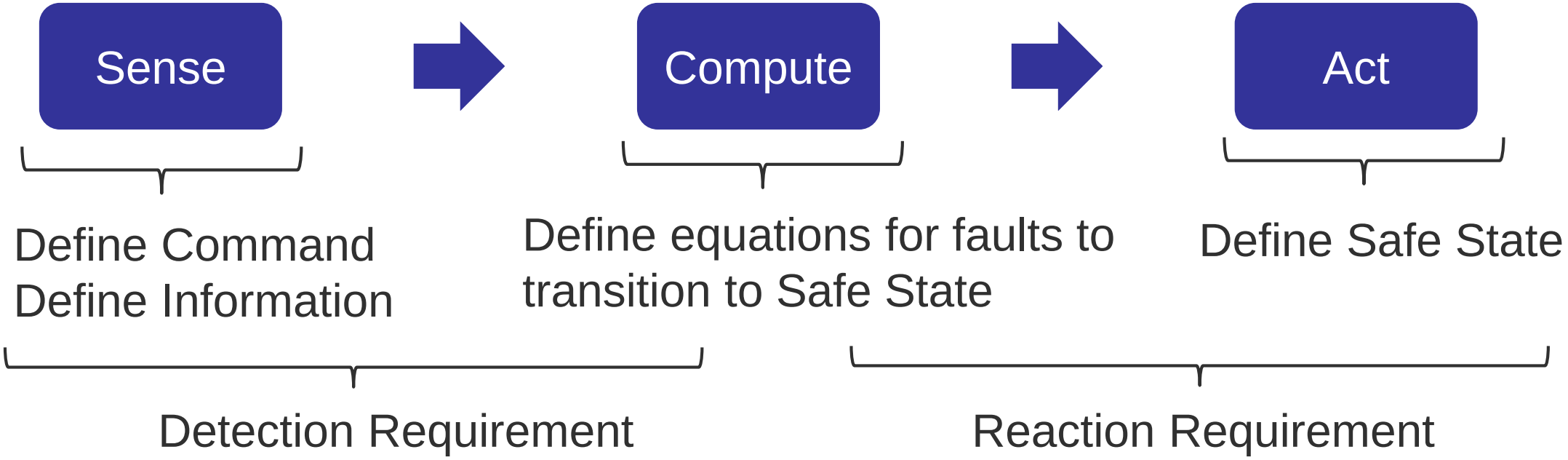
- Define WHAT not HOW
- Should be:
  - Complete / Atomic
  - Consistent
  - Comprehensible
  - Realistic / Feasible
  - Verifiable
  - Valid / Correct
  - Necessary

Requirements Writing

Desired format of requirements:
<**Function/Object**> shall <**Action**><**Condition**> <**Testable Result**> <**Reaction Time**>

- Each requirement should be written with a standard style and contain the following components:
  - **Action**: Operation design will perform. Atomic and unambiguous.
  - **Condition**: Under what conditions is the action performed.
  - **Testable Result:** What will occur. Should be specific.
  - **Reaction Time**: A bounding time. For instance, a ‘within’ time frame.
    - Time should in the proper context. Stay away from implementation details.

Creating Safety Requirements



Requirements should be testable - can be viewed as preliminary test cases.

Value of Tracing

Top Down Trace – Find **unallocated** and/or **unimplemented** requirements

System Requirements

Hardware Design Requirements

Hardware Design Documentation

Hardware Design Implementation

Bottom Up Trace – Find **unnecessary**, **unneeded**, **unwanted** functions or features

Top Down Trace – Find **untested** requirements, **failing** requirements, **unimplemented** tests

System Requirements

Hardware Design Requirements

Verification Plan

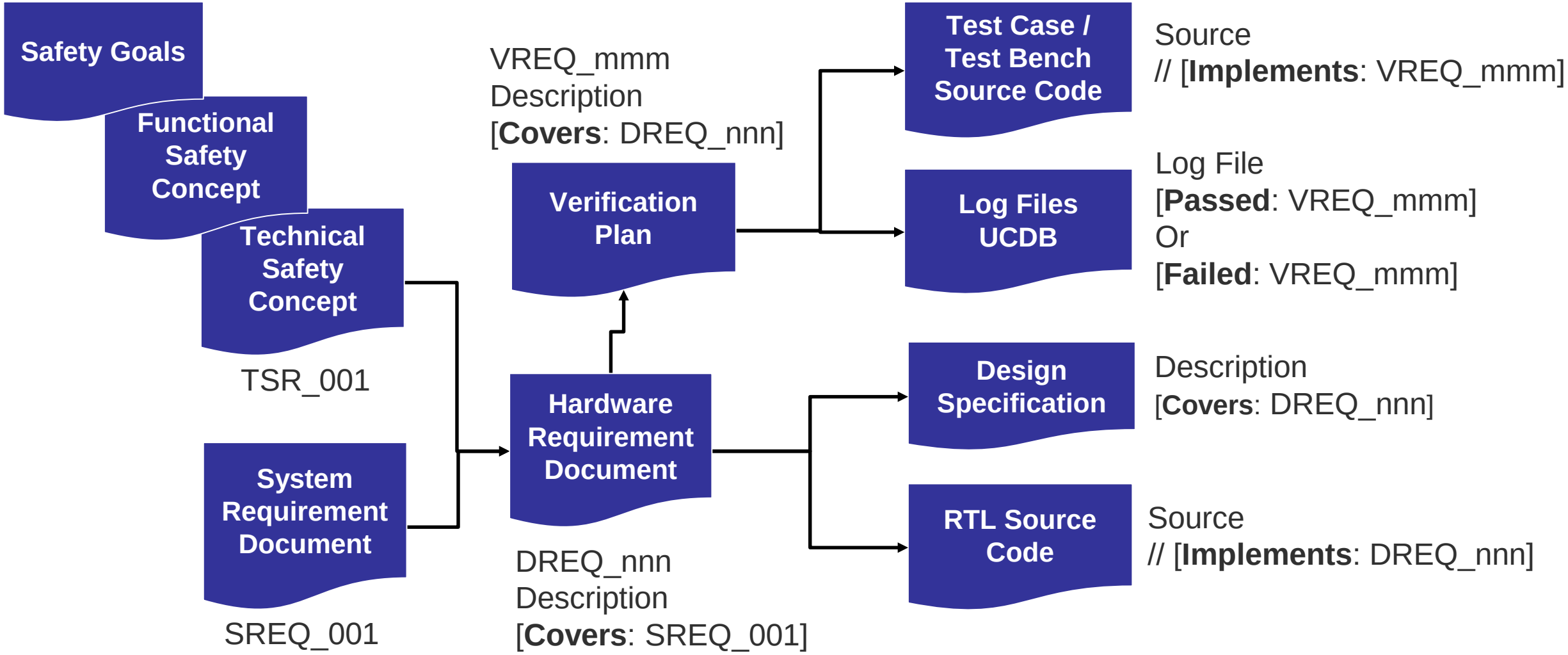
Test Bench Implementation

Testing Artifacts

Bottom Up Trace – Find **undocumented** testing

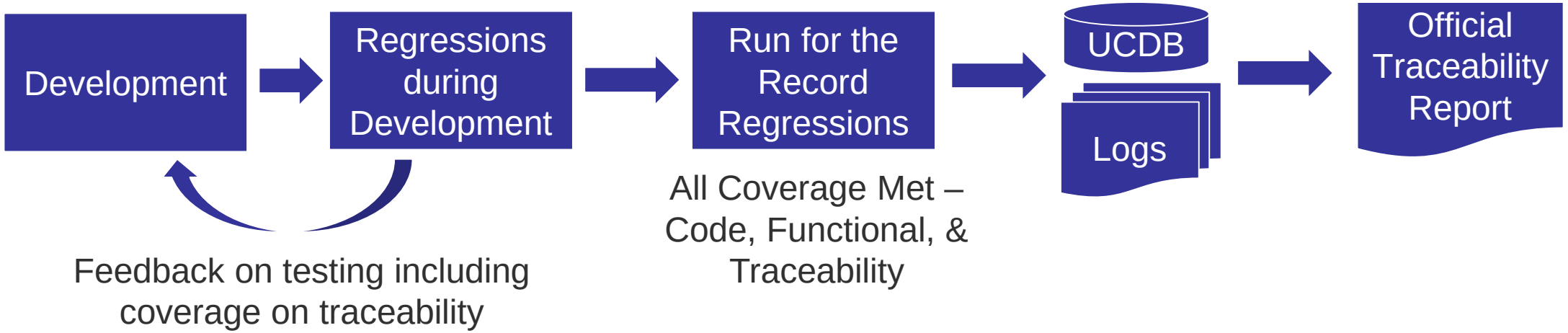
Tracing Requirements

26262 8-6



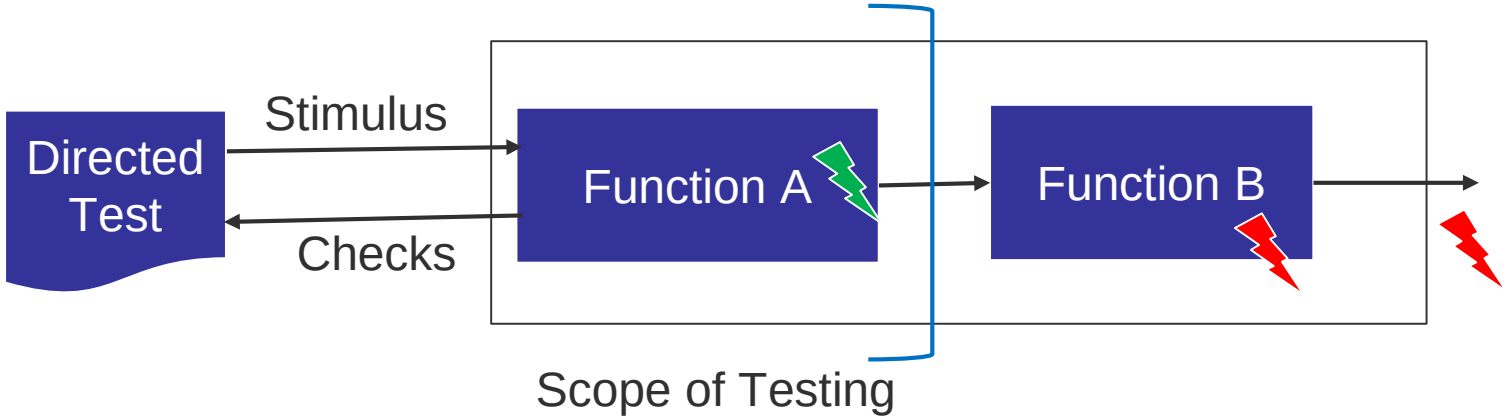
Testing Artifacts

- Requirements must trace into the testing artifacts
  - Shown to have been actively tested and shown to pass
  - For simulations, typically UCDB and/or Test Log Files
  - Artifacts from a “Run for the Record” regressions used for final reports.



Directed Testing & Requirements

- Directed Tests are often used for 1-1 match of requirement to test
- However, typical Directed Tests driven to satisfy requirements tend to have shortcomings:
  - Not complete in testing across the full design
  - Down stream errors are not checked
  - Are limited to specific times, situations



Random Testing & Requirements

- Random Testing and UVM Test Benches allow a smaller set of Test Cases to address multiple requirements concurrently.
- Random testing of requirements & checking for passing is the AND of:
  - Test Case Passing
  - Appropriate Stimulus Generated
  - Appropriate Prediction Generated
  - Results are checked and match
- Checks distributed work across test cases, predictors, and scoreboards
  - AND function can be addressed by Functional Coverage

Tracking in a Test Bench

- Logging for traceability occurs where testing of a requirement is done
- Typically is an ‘else’ in an error check
- Simple Functional Coverage is okay IF run for record must achieve 100% passing test cases

```
if (expected_crc != actual_crc)
    `uvm_error("DUT generated bad CRC")
```

becomes

```
// [Implements: VREQ_nnn]
if (expected_crc != actual_crc)
  `uvm_error("DUT generated bad CRC")
else begin
  `uvm_info("DUT generated good CRC")
  Add to specific VREQ to covergroup
end
```

[DES_REQ_nnn] When sending a message out on the channel, the design shall calculate a CRC in accordance with …

[VREQ_nnn] The Test Bench shall have a checker on DUT channel output that ensures all messages generated by the design have a correct CRC….

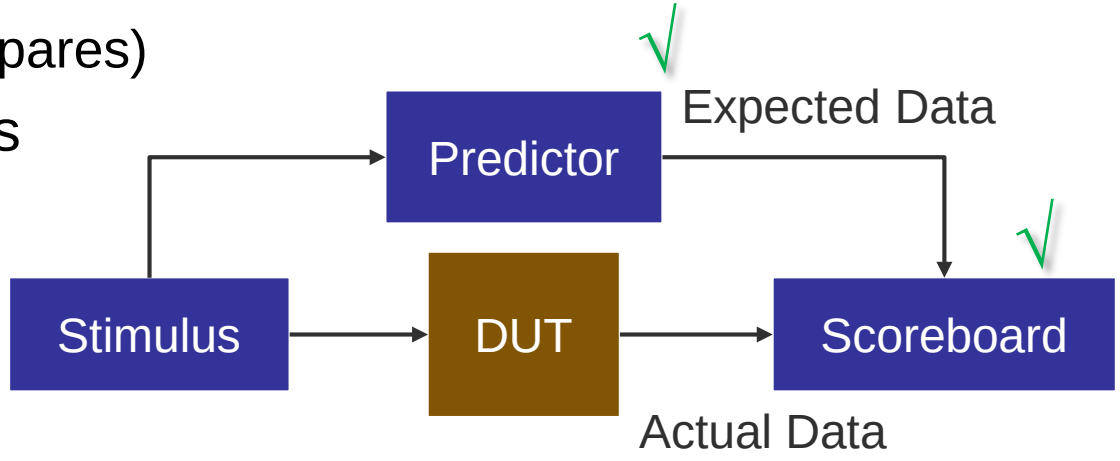
Tracking in a Test Bench

- If failing test cases can occur in run for the record,
  - Passing test cases may have set functional coverage
  - Create passing / failing covergroups
  - Coverage of failing conditions trumps good covergroup.
  - Parsing log files can accomplish similar tracking

```
// [Implements: VREQ_nnn]
if (expected_crc != actual_crc) begin
  `uvm_error("DUT generated bad CRC")
   Add specific VREQ to bad covergroup
end else begin
  `uvm_info("DUT generated good CRC")
  Add specific VREQ to good covergroup
end
```

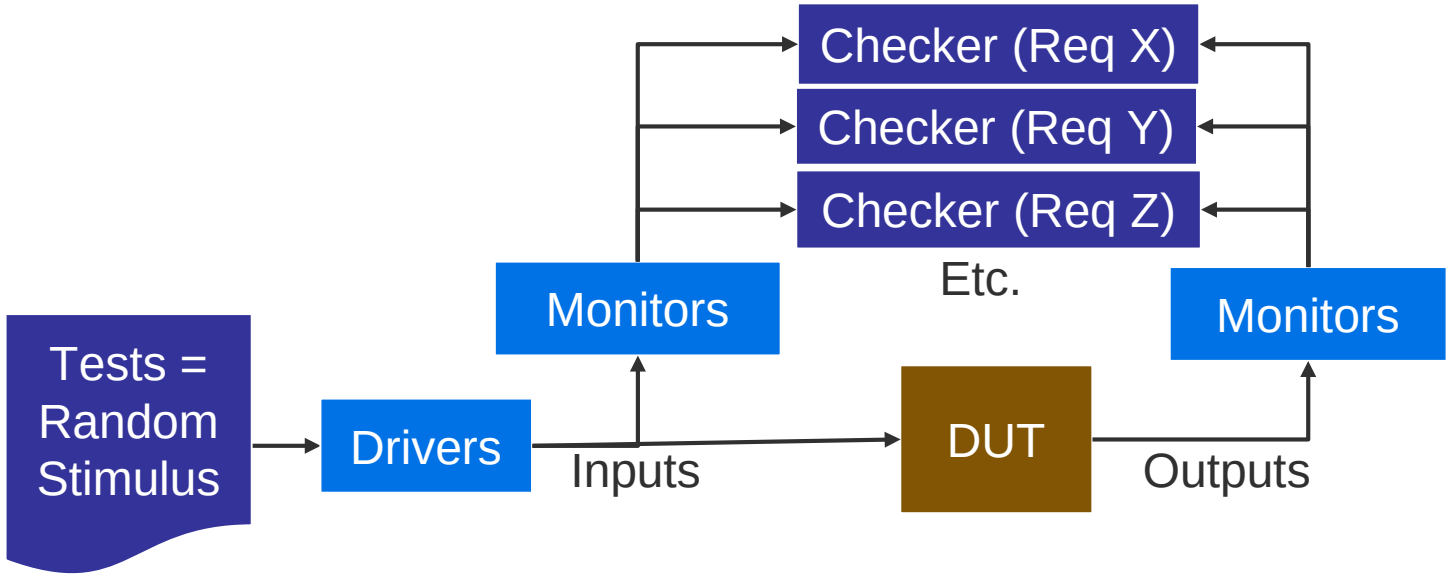
Tracking in a Test Bench

- UVM Test Benches distribute work so checking may be too simplistic for tracing
  - Scoreboards may simply compare expected data against actual data
  - May not be possible to isolate checks to a specific requirement
- Usually the ‘predictor’ can be associated with a requirement
  - A requirement would then be considered passing if:
    - The Predictor made appropriate prediction
    - Test Case has passed (no scoreboard miscompares)
  - Coverpoints created to AND these conditions



Predictor to Requirement Mapping

- For some designs, it may be possible to create a more direct predictor/checker mapping to requirements
  - Tradeoff of complexity in checkers versus complexity in tracking



Assertions -> Requirements

Assertions can also be assigned per requirement.

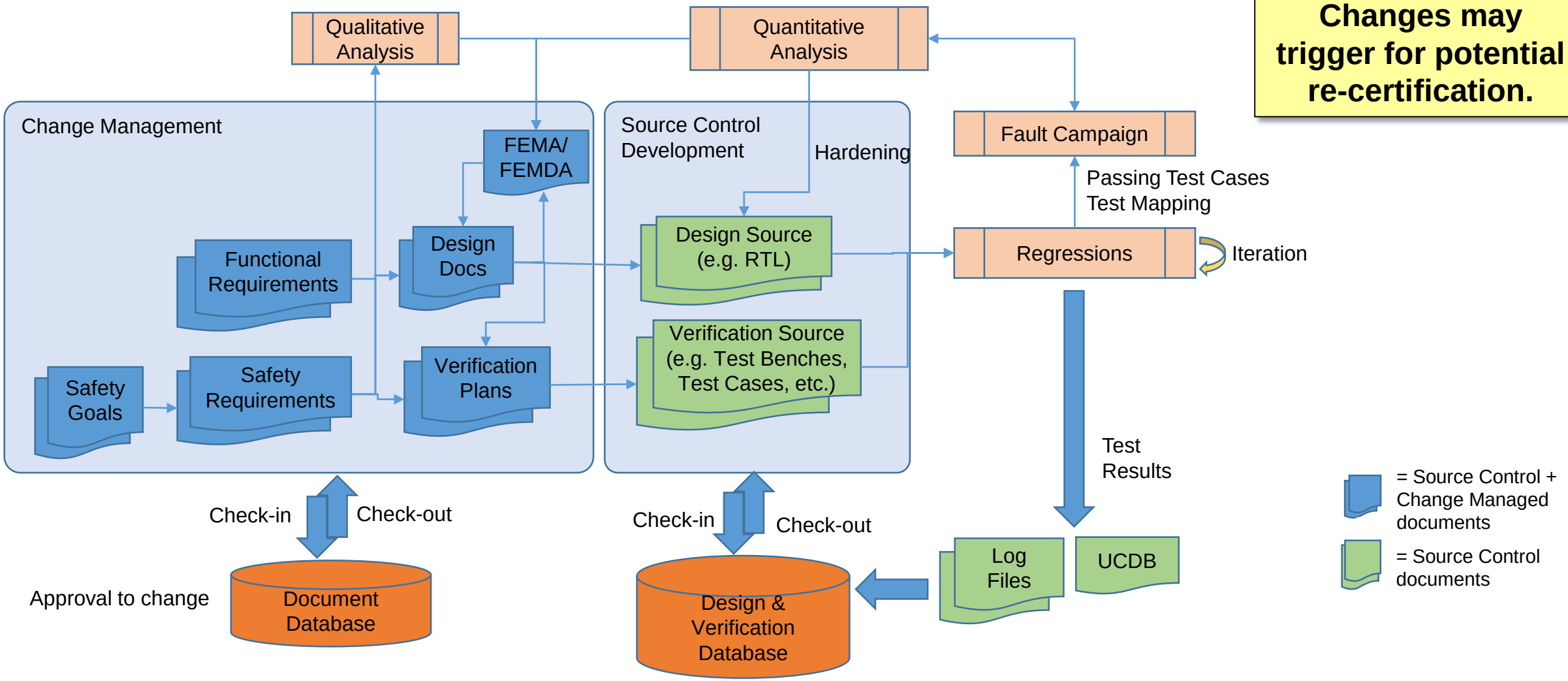
A proved assertion is positive coverage.

```
// formal randomly picks a bit(s) to flip.
asm_mask:  assume property ( $countones(one_error_mask) == 1 );
// Check ECC repair is correct
req_nnnn:  assert property ( fixed_data[7:0] == data );

-- XOR mark to flip 1 bit
one_error_data <= one_error_mask XOR encoded_data;
fixed_data <= ecc_correction_function(one_error_data);
```

```
function [12:0] ecc_calc( data, … );
 wire logic p1 = 1 ^ data[0] ^ data[1] ^ ...
 wire logic p2 = …
 …
 return ({data[7],data[6],data[5],data[4],p8,data[3], … } );
endfunction

// Check ECC calculation
req_xyz: assert property ( encoded_data == ecc_calc(data, … ));
```

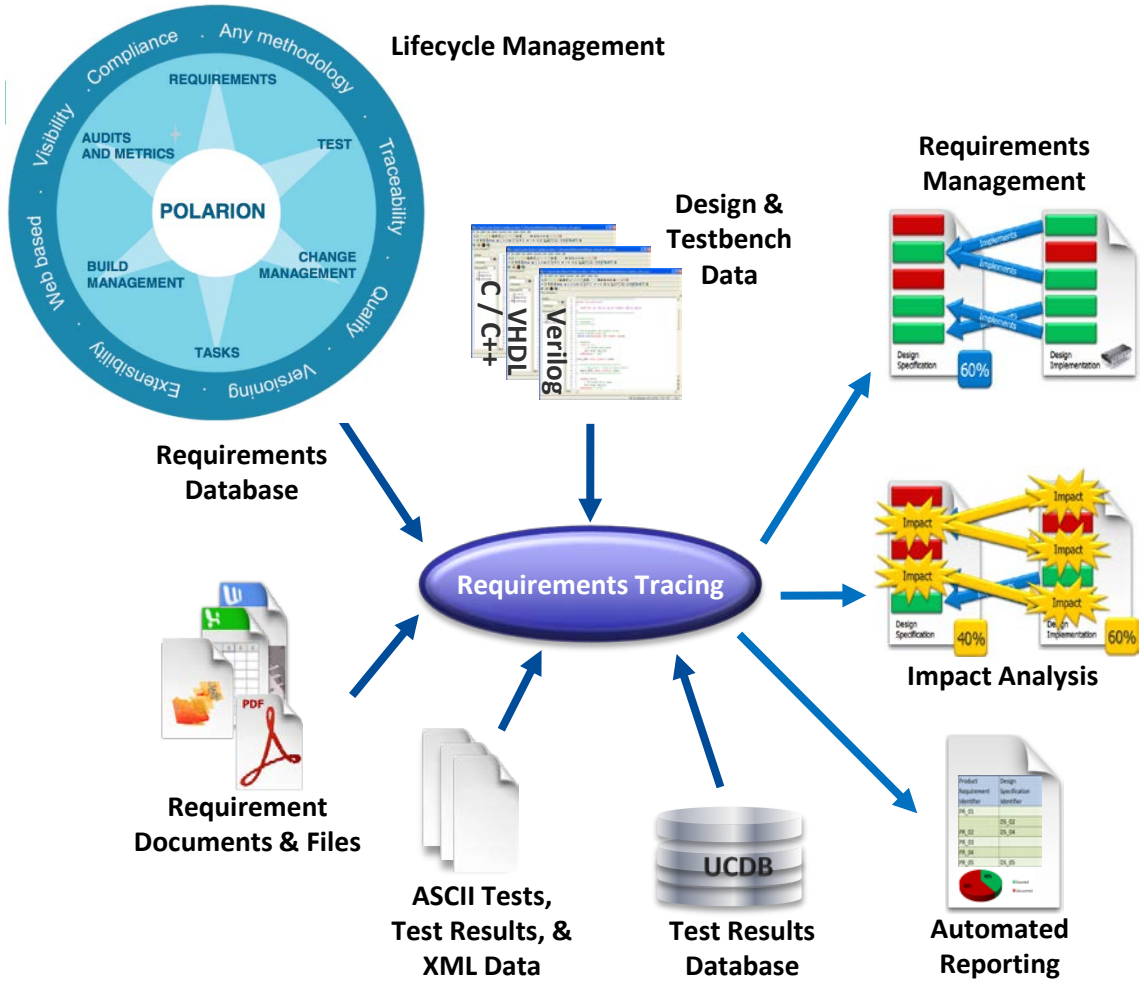
Requirements Management

After release to production. Changes may trigger for potential re-certification.



Requirements Tracing Tools

- A centralized view that connects the development process and results
- Traceability at all stages of development
- Quickly understand the impact of a change across the project
- Reflects the current status of the project using live data



Summary

- ISO26262 defines:
  - Top down flow of safety requirements
  - Requires precise language for requirement definition
  - Traceability
  - Change Management and Source Control
- Poor requirements creates an unstable base to build on
- Tracing should be done into verification artifacts

Questions?

Contact Information

Charles Battikha (chuck_battikha@mentor.com)
Doug Smith (doug_smith@mentor.com)

https://www.mentor.com/mentor-automotive

White paper - “How Formal Reduces Fault Analysis for ISO 26262”
http://go.mentor.com/4QQrY