2024
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES

SAN JOSE, CA, USA
MARCH 4-7, 2024

Functional Safety Workflow of Internal IP (NPU) Within Large Automotive IC Through Analysis and Emulation Usage

Likhopoy Andrey
SAMSUNG

Kim Inhwan
SIEMENS

accellera
SYSTEMS INITIATIVE

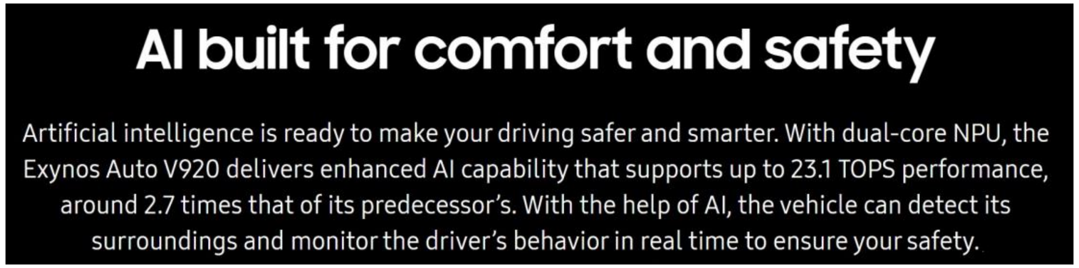
Agenda

Purpose

- Reduce Time to Safety using a unified end to end integrated toolset composing of analysis, simulation and emulation
  - Target
    - Verify DC to achieve ASIL B with fault injection test
  - Tasks
    - Use emulation flow with analysis and optimizations
    - From functional verification to functional safety verification
      - Environment migration
      - Vector analysis and prioritizing

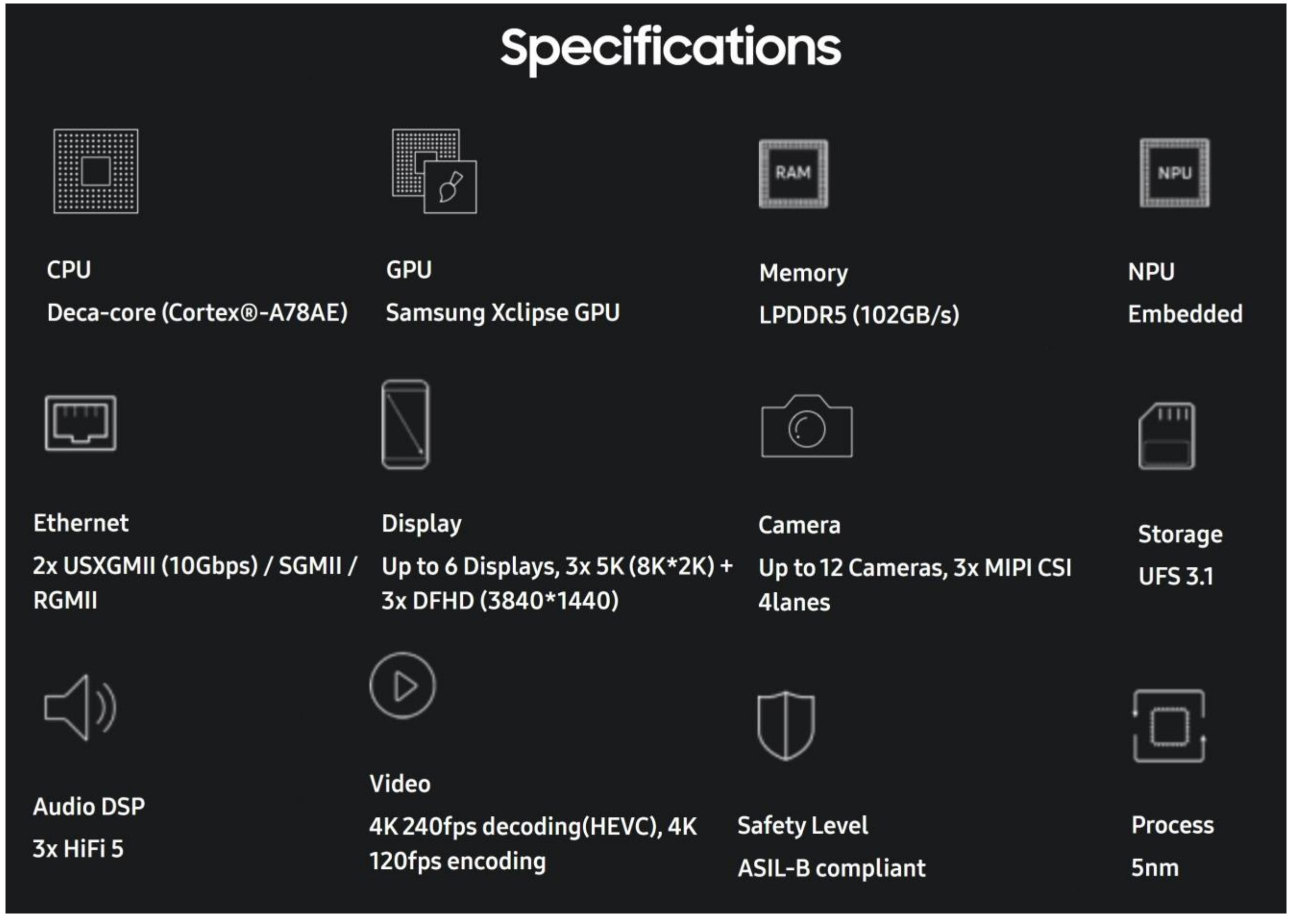
Introduction – Automotive SoC

https://semiconductor.samsung.com/us/processor/automotive-processor/exynos-auto-v920/



AI built for comfort and safety

Artificial intelligence is ready to make your driving safer and smarter. With dual-core NPU, the Exynos Auto V920 delivers enhanced AI capability that supports up to 23.1 TOPS performance, around 2.7 times that of its predecessor's. With the help of AI, the vehicle can detect its surroundings and monitor the driver's behavior in real time to ensure your safety.



Specifications

CPU	Deca-core (Cortex®-A78AE)
GPU	Samsung Xclipse GPU
Memory	LPDDR5 (102GB/s)
NPU	Embedded
Ethernet	2x USXGMII (10Gbps) / SGMII / RGMII
Display	Up to 6 Displays, 3x 5K (8K*2K) + 3x DFHD (3840*1440)
Camera	Up to 12 Cameras, 3x MIPI CSI 4lanes
Storage	UFS 3.1
Audio DSP	3x HiFi 5
Video	4K 240fps decoding(HEVC), 4K 120fps encoding
Safety Level	ASIL-B compliant
Process	5nm

Introduction – Functional Safety

- Functional safety – absence of unreasonable risk due to hazards caused by malfunctioning behavior of electronic systems (ISO 26262)
- HARA: Hazard Event (HE) assigned with ASIL based on Severity (S), Exposure(E), Controllability (C)
- HE -> SG -> SR -> SM

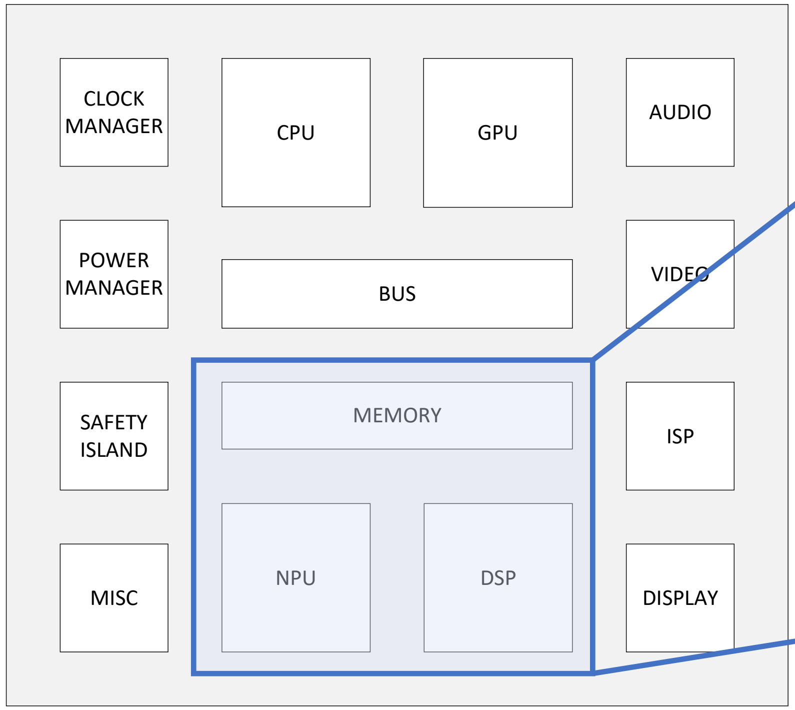
SG Safety Goal

SR Safety Requirement

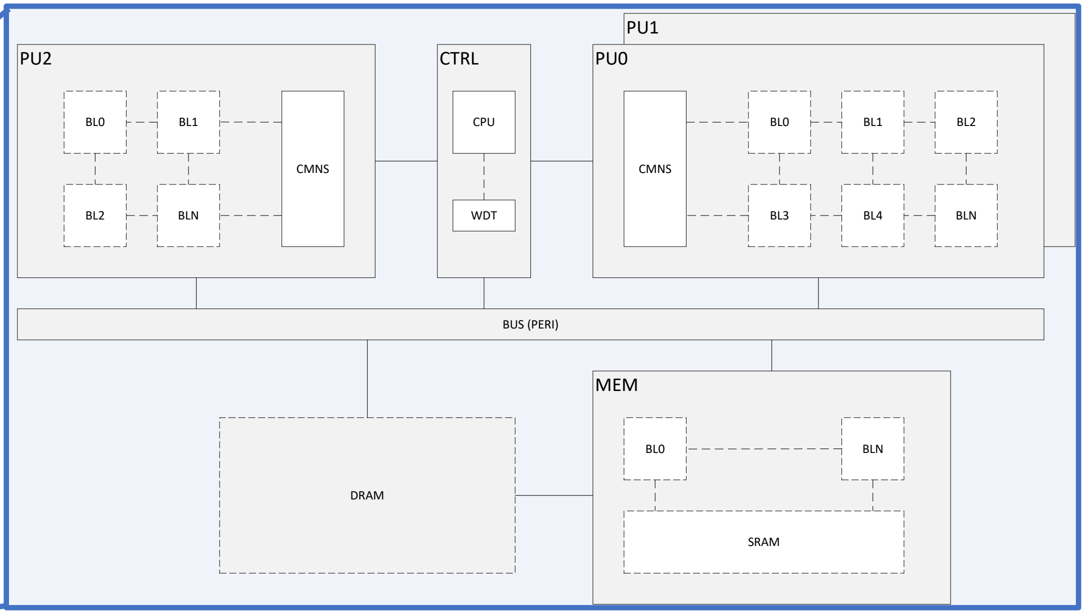
SM Safety Mechanism

ASIL	SPFM	LFM	PMHF
A	Not relevant	Not relevant	< 1000 FIT
B	≥ 90 %	≥ 60 %	< 100 FIT
C	≥ 97 %	≥ 80 %	< 100 FIT
D	≥ 99 %	≥ 90 %	< 10 FIT

NPU Description – 1



Typical automotive SoC



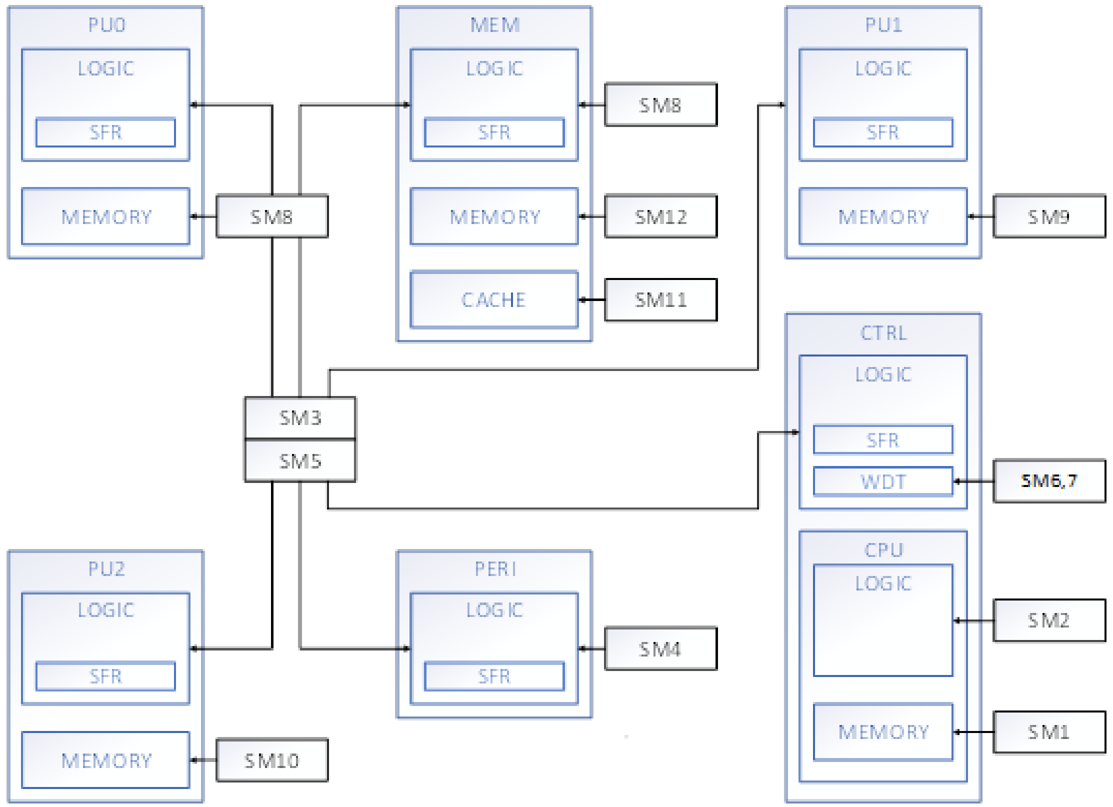
NPU block diagram

NPU Description – 2

- Part of an automotive SoC
- Contributes to safe video sensing
- NPU provides fast and energy-efficient hardware accelerations for AI and ML applications in computer vision, image processing and other domains
- SoC includes Safety Island (and FMU)
- Target DC ≥ 91.21% for ASIL B

SM Description

Type	SM	Part	Description
emulation	SM1	CPU memory	ECC
	SM2	CPU logic	STL
	SM3	PU0/1/2, MEM, CTRL, PERI	Self-test
	SM4	CTRL, PERI (bus)	Ext mem access
	SM5	SFR of all parts	SFR access
	SM6	CTRL	WDT
	SM7	CTRL	WDT SFR
	SM8	MEM	CRC
simulation	SM9	PU0 memory	Parity
	SM10	PU1 memory	Parity
	SM11	PU2 memory	Parity
	SM12	MEM cache memory	Parity
	SM13	MEM memory	Parity



Expected DC Based on FIT Rate Distribution

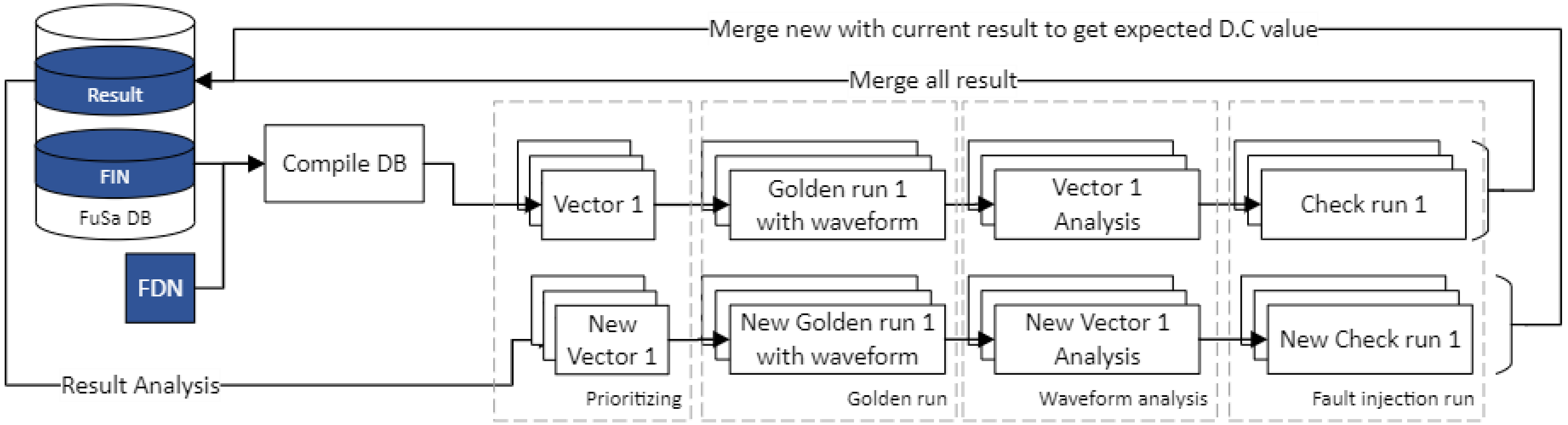
Block	FIT rate distribution (%)	Expected DC (%) (≥91.21%)
PU0	31.35	96
PU1	31.35	96
MEM	17.9	90
PU2	10.86	80
CTRL	2.04	60
PERI	6.48	80
Total	100	91.41

SM3 (Self-test)

Part	FIT rate distribution (%)	Expected DC (%) (≥60%)
CPU memory	1.02	90
CPU logic	98.98	60
Total	100	60.31

SM1, SM2 (CPU)

Fault Campaign Emulation Flow



1. Fault list (FIN) generation with SafetyScope
2. Define FDN
3. Veloce Compilation with FDN & FIN (from FuSaDB)
4. Select vectors based on priority
5. Golden run
6. Test vector analysis with golden waveform
7. Fault injection run
8. Result analysis (DO, DU, UU, UO)

Effective Platform for Complexity

- Fast run for complex design with safety mechanism
  - Fast (depends on test scenario, up to ~500x compared to simulation)
  - According to the complexity and runtime, verification platforms are partitioned. FC simulation covers some subparts in NPU while Emulation FC covers NPU according to bottom-up methods

Effective & Optimized FC Flow

- Fast FC = reduced fault injection run
  - Compile (dead logic based on synthesis)
    - Dead logic which are not directly read or doesn’t have any reader
  - Vector Quality Check (Waveform analysis)
    - Testing vector with waveforms to see toggle info for meaningful stuck at fault

Effective Analysis Methodology

- Easy to debug
  - Propagation path analysis (Cone-of-influence)
    - To see only the path between fault injection net and fault detection net

  - The analysis methodology
    - To find out the debug point from golden run & fault injection run

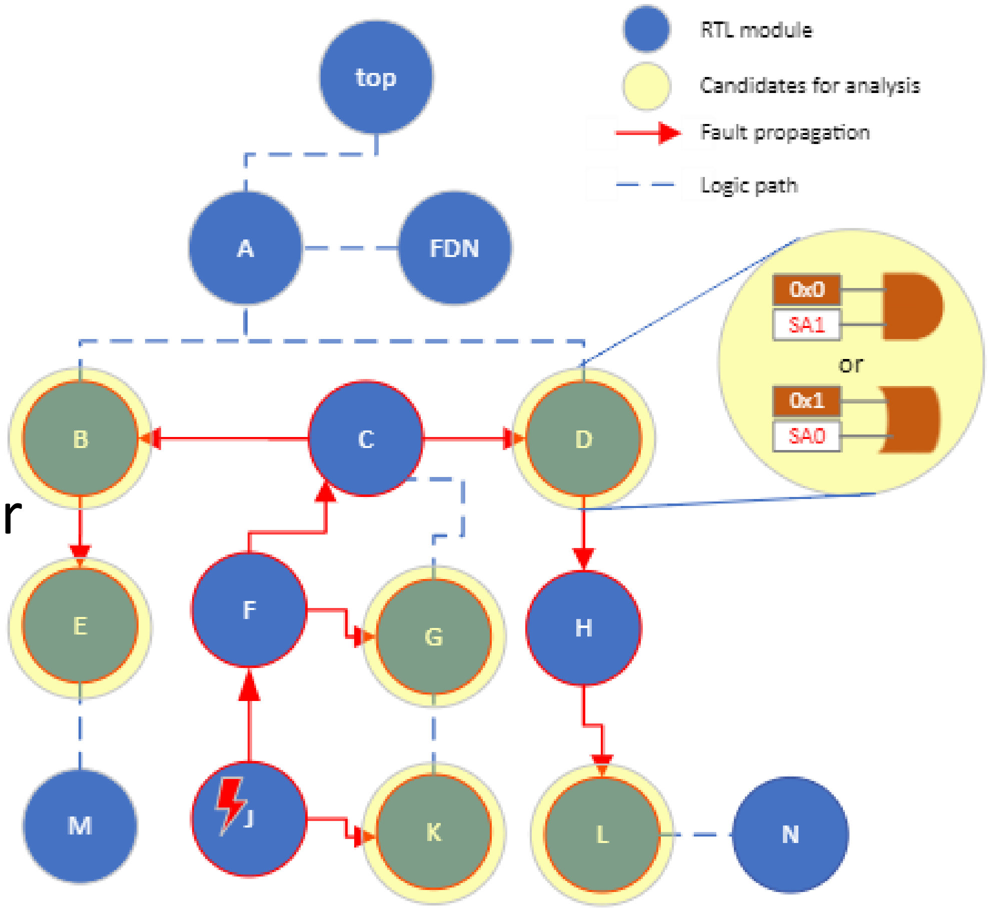
Difficult Point in Fault Campaign

- Safe fault classification based on analysis
  - Lower quality of vector can have UU
  - Specific logic or SW may block the propagation by injected fault

- Engineer judgment is needed to define UU as Safe fault with evidences

UU Analysis – Fast Triage

- Assumption
  - The deviation by fault injection can be found from waveform compare (golden vs check run)
  - FDN is designed to receive all the propagation from FIN
  - The propagation can be gated out due to SW or HW
  - The latest deviation point in simulation time will be analyzed
  - The nearest deviation point to the FDN will be analyzed



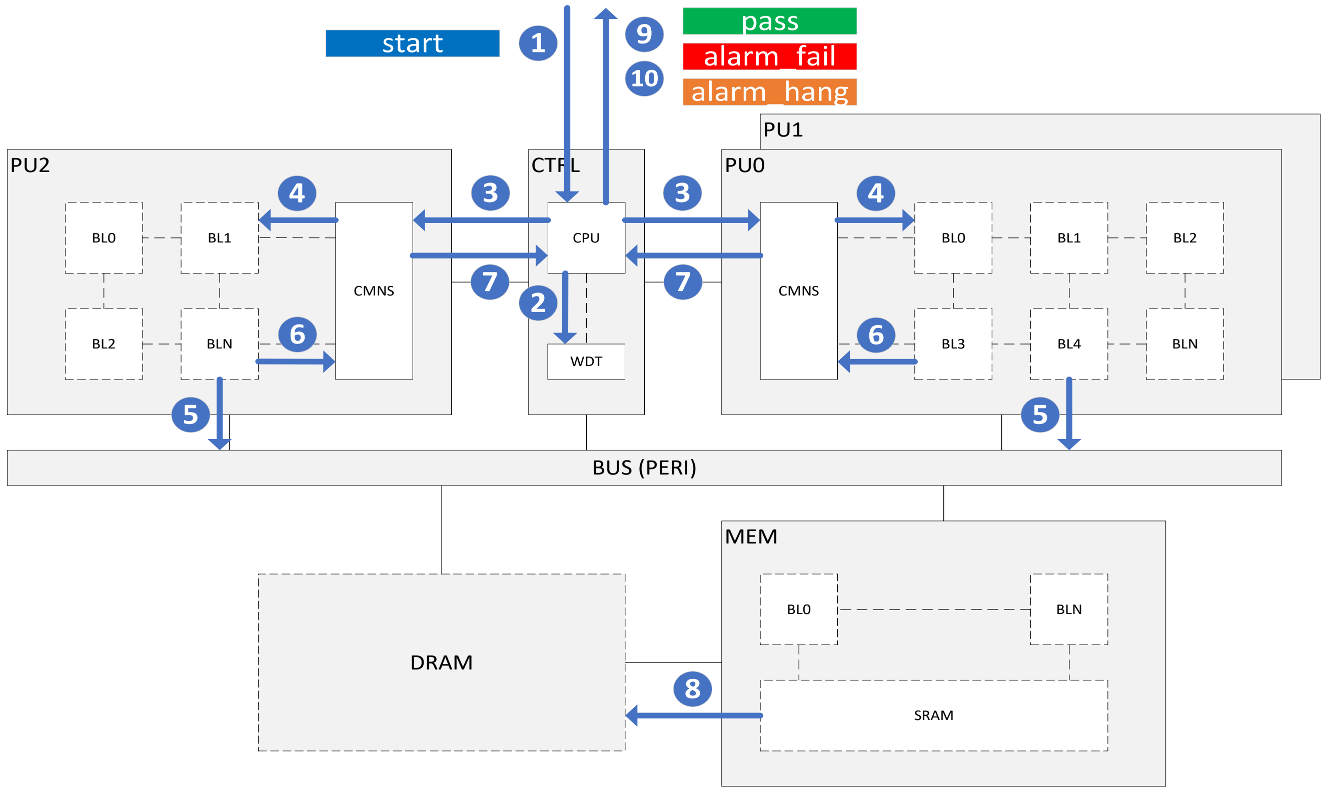
UU Analysis – Others

- Configuration check
  - Configuration status with STL (Test vector for ARM core) can be analyzed with HW – SW co-debugger (Codelink)
- Memory read check
  - Read memory has an effect on fault propagation if the fault is injected on the memory cell

Test Scenarios Generation – Conditions

- Test scenarios for FC must meet two conditions, both high quality and short runtime
  - High quality – above the sign-off level of test scenarios for functional verification (coverage: functional != diagnostic) to achieve DC for ASIL B
  - Short runtime – faster than test scenarios for functional verification (redundancy) as to be used for actual field tests
- No simple method to make an ideal test scenarios

Test Scenarios Generation – Sequence



1. Host boots CPU
2. CPU activates WDT
3. CPU triggers CMNS
4. CMNS configures internal blocks
5. Internal blocks read, process, write data
6. CMNS waits internal blocks end
7. CPU gets interrupts from CMNS
8. CPU triggers MEM to generate CRC/reads internal CRC
9. CPU compares CRC with ref and signals results (pass/fail)
10. CPU can get interrupt from WDT and signal (hang) or stops WDT at all operation end

Test Scenarios Generation – 1

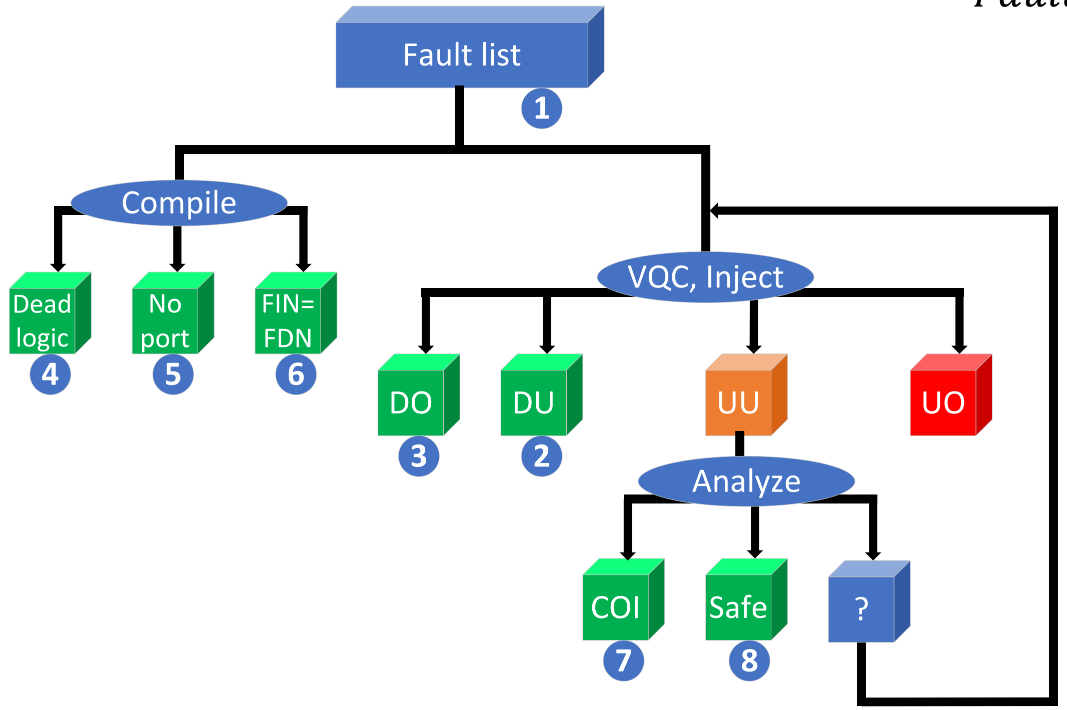
- Test scenarios used in real SW not useful
- Test scenarios for functional verification as a starting point
  - Rank by toggle coverage to remove duplicated
  - Estimate if execution time is met
- Newly generated scenario detects both
  - data mismatch (a)
  - system hang (b)

Test Scenarios Generation – 2

- Data mismatch (a)
  - Sensitive configuration of each sub-IP
    - One – enabled, others – disabled (e.g. LUT chain)
  - Special data patterns
    - Intermediate values to prevent saturation due to clipping (e.g. MAC)
    - MSB, LSB “0”/“1”, others “1”/”0”
    - SFRs all “0”/all “1”
- System hang (b)
  - Adjust WDT

DC Calculation – 1

- $DC \textcircled{9} = \dfrac{DU\textcircled{2} + DO\textcircled{3} + Compile\left(Dead\ Logic\textcircled{4} + No\ port\ in\ mem\textcircled{5} + FIN{=}FDN\textcircled{6}\right) + COI\textcircled{7} + Safe\textcircled{8}}{Fault\ list\ \textcircled{1}}$



- After each iteration one should analyze UU
  - Move to DO by generating new test scenario where the fault propagates to alarm net
  - Move to Safe by confirming the fault has no effect on observe net

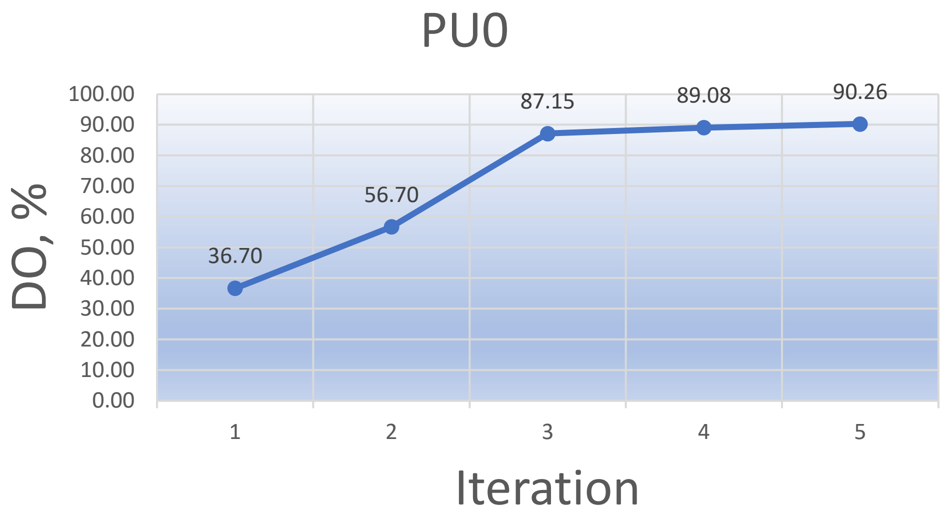
DC Calculation – 2

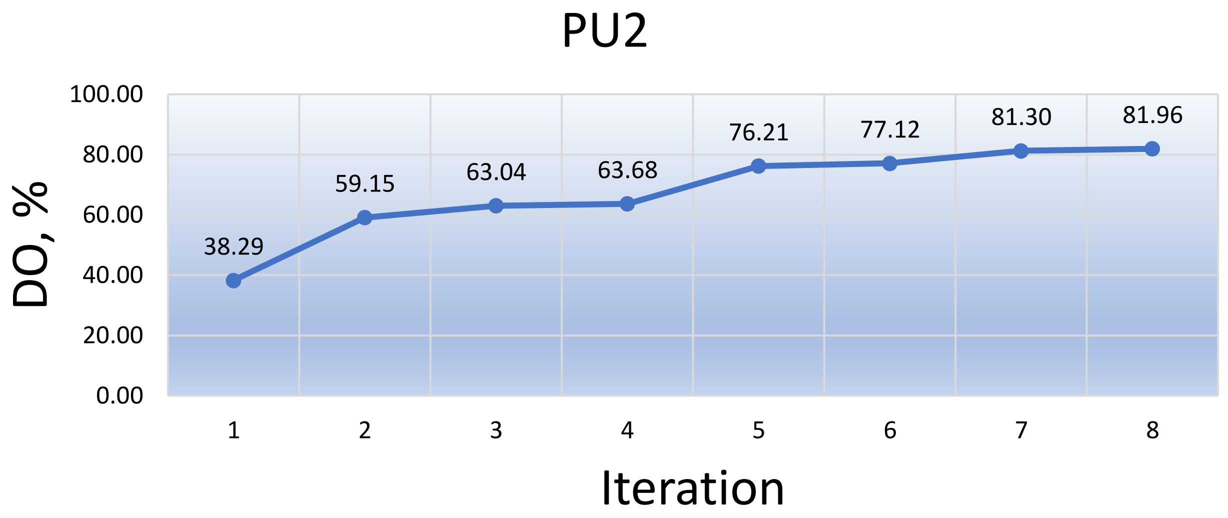
- Analysis of NPU architecture, data flow and algorithms
- Move to DO if fault
  - propagates
- Move to Safe if fault in
  - Spec-out feature
  - Debug feature
  - Clock-gating disable feature
  - Invalid range
  - Performance feature

Results – Iterations

		DO for iteration, %				
Block	Valid net, %	1	2	3	4	5
PU0	100	36.70	56.70	87.15	89.08	90.26

		DO for iteration, %							
Block	Valid net, %	1	2	3	4	5	6	7	8
PU2	100	38.29	59.15	63.04	63.68	76.21	77.12	81.30	81.96





Results – SM3 (Self-test)

	FC steps						Result analysis					Result		
	FuSa DB	Compile	FI run				Compile (optimized)			Formal	Expert	Expected		Final
Block	Fault list ①	Valid net (after Compile)	DU ②	DO ③	UU (incl. VQC)	UO	Dead Logic ④	No port in mem ⑤	FIN = FDN ⑥	COI ⑦	Safe ⑧	FIT rate distr. (%)	Exp. DC (%)	DC (%) ⑨
PU0	4694	4220	0	3827	281	132	438	16	0	19	227	31.35	96	96.44
PU1	4694	4220	0	3827	282	131	438	16	0	19	227	31.35	96	96.44
MEM	4688	3472	0	2392	1022	58	1148	0	0	78	316	17.9	90	83.92
PU2	4658	4064	0	3331	711	22	522	72	0	21	80	10.86	80	86.43
CTRL	4582	2050	6	235	1716	93	1430	1102	0	131	0	2.04	60	63.38
PERI	4576	3058	0	2151	895	12	1518	0	0	75	0	6.48	80	81.82
Total												100	91.41	91.49

Results – SM1, SM2 (CPU)

	FC steps						Result analysis					Result		
	FuSa DB	Compile	FI run				Compile (optimized)			Formal	Expert	Expected		Final
Part	Fault list ①	Valid net (after Compile)	DU ②	DO ③	UU (incl. VQC)	UO	Dead Logic ④	No port in mem ⑤	FIN = FDN ⑥	COI ⑦	Safe ⑧	FIT rate distr. (%)	Exp. DC (%)	DC (%) ⑨
CPU memory	4696	4642	320	3835	486	1	54	0	0	0	0	1.02	90	89.63
CPU logic	4554	4174	0	2303	1818	53	364	0	16	94	0	98.98	60	60.98
Total												100	60.31	61.13

Conclusion

- Functional safety workflow of NPU IP with SW-based SMs
- Internal blocks and related SMs selection based on FIT rate distribution
- FC process including fault injection run by emulation
- Debug methodology for efficient UU analysis
  - DO (test scenarios), Safe (expert judgement)
- Target DC score of 91.21% necessary to meet ASIL B metrics
- Future work – further optimization of test scenario runtime for SW safety verification

Questions