2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

SDV Aware Verification : Verification Challenges, Opportunities and Evolution around Software Defined Vehicles and Zonal Architectures

Neha Srivastava
NXP Semiconductors India Pvt Ltd
neha.srivastava@nxp.com

***Abstract*- Software-Defined-Vehicles (SDV) with their inherent domain or zone-based automotive architectures [1][2] have certain tenets which are leading to an evolution in the industry trends and requirements. The major ones are Hardware-Software(HW-SW) co-design, provision of on-road Over-The-Air (OTA) patching by SW, and freedom-from-interference between the SW cohorts. Associated with it are also revolutionary shifts in how verification should be planned, executed and signed off. This paper describes the fundamental, underlying changes to the verification environments and associated methodologies. Notable among these are verification checks to ensure alignment between HW-development stages and SW-development stages; ensuring equivalence of SW prototype model to HW; functional coverage and signoffs for SW controllability of HW; and ensuring freedom-from-interference through granular controls over each HW element and isolations between them. Significant efficiency gains and earlier enablement of SDV customers, in the order of 12-24 months, are achieved through confidence from SDV aware verification signoffs. Around 500 elements comprising cores, memories, peripherals across 315 SW resettable domains were verified with a combination of formal, structural and dynamic techniques. As an added benefit, the confidence from pre-silicon SW configurability sign-offs also saved silicon testing time, from days to minutes, for certain characterization tests.**

I. Introduction

Software(SW) has a much bigger relevance on how customers configure and use our automotive solutions [3], including over-the-air (OTA) updates to extend the vehicle use even after years of usage. There is an increasing interleaving of roles of Hardware(HW) features and SW configurability and any limitation found on an on-road vehicle does not demarcate between HW and SW any longer. And hence it is essential for SDV development to ensure that SW delivery is early and high quality with equivalence and synchronization established between HW and SW at each development stage. The traditional HW-centric verification approach does not suffice for SDV architectures. Hence there is a need to incorporate perfect interlocking between HW and SW perspectives of planning and executing verification for SDV aware signoffs. These verification methodologies have been deployed in our last SDV product [5] and have helped intercept the new scopes and challenges associated with SDV. With the learnings from failure of few of these deployments, there are also opportunities to improve in the future.

II. SDV tenets and Industry evolution trends

A. HW-SW Co-Design : Digital twins

With increasing chip demand and an intent to productize solutions faster, industry can no longer afford delays due to SW-enablement after HW-development cycle is complete. The SW development and testing needs to proceed in parallel with HW development and verification. This also allows early feedback to architecture and design (IP, SoC, MCU) community during earlier cycles of HW development to rectify any disconnects on requirement expectation vs design implementation [6].

As shown in “Fig. 1”, there are multiple advantages to chip makers as well as customer through this approach.

1. **Early prototyping** : This ensures customer requirements and design behaviors are modelled into a prototype SW library package which can be delivered to customer much before chip delivery (even with start of architecture definitions and HW development). The chip at customer delivery stage would be much better quality and meeting customer requirements to more precision compared to traditional approaches.

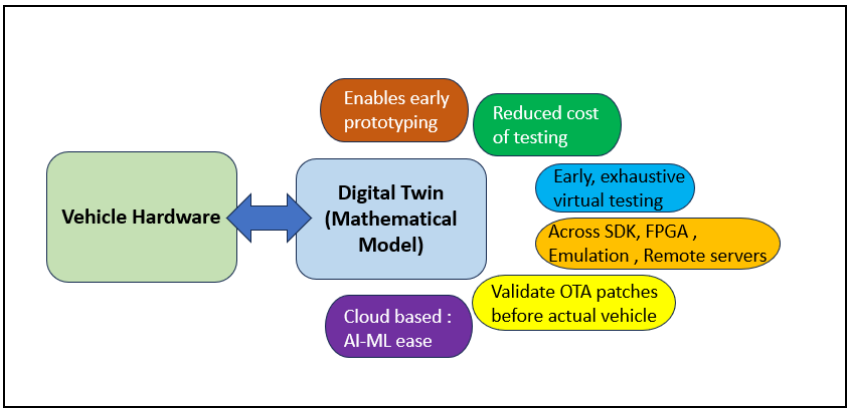


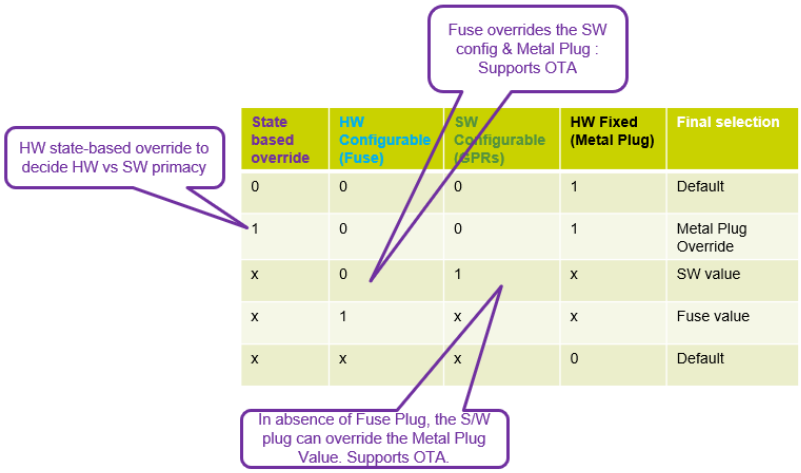
Figure 1. HW -SW Co-Design : Advantages

2. **SW-centric exhaustive testing** : With designs themselves becoming more SW-centric, if we leave the SW development to later cycles after HW development completion, we can't really model and focus on testing how SW controls the HW behaviors or how the overall functionality is a synergy between HW and SW, both of which HW architects and HW developers/testers can easily oversee or bypass).
3. **Reduced cost of testing** : APIs and common learnings can be leveraged in parallel between HW and SW developers and testing teams as well as with customer SW development teams. There can be complete reuse across Pre-Silicon IP/SOC verification, Accelerated Verification, FPGA testing, Pre-Silicon Validation, Applications and SW teams, OEM SW development teams, and customer VDK (Virtual Development Kit) .

B. SW Patch , OTA provision : Flexible , Configurable HW

A large aspect of SDV is the ability to have SW updates Over-the-Air(OTA) while HW remains constant [4]. True extension of life of a vehicle can only happen if we are not limited by HW ageing but rather if features can be added or enhanced purely through SW updates, even years into on-road operation of vehicle.

A lot of such product commitments to customers can be verified on the digital twin to ensure HW is configurable enough to accept OTA SW updates. Also, SW should have control and observability over most of the HW features. As shown in "Fig. 2", all critical and extendable features of the SoC can be developed such that they always have a SW configurability in addition to HW fixed (e.g., hardcoded feature or a metal plug which can only be changed on a tape-out) or HW configurable (e.g., programmable fuses) mechanisms. This ensures that through the SW general purpose controls and by application / Firmware (FW) controls, functionality of a device can be enhanced.



State based override	HW Configurable (Fuse)	SW Configurable (GPRs)	HW Fixed (Metal Plug)	Final selection
0	0	0	1	Default
1	0	0	1	Metal Plug Override
x	0	1	x	SW value
x	1	x	x	Fuse value
x	x	x	0	Default

Figure 2. HW -SW Inter-configurability allowing SW OTA updates

C. *Freedom from Interference : Isolations between cohorts and applications*

The different zones or virtual cohorts which are operating in parallel in a SDV need to be completely free from any interference from the functionality or failures of other cohorts/zones. Such isolation controls should be mappable by SW to dynamically varying cohort definition within an application.

This requires a list of all HW resources in the system which can be assumed to the lowest granularity building blocks for the varying, SW-defined virtual cohorts in different SW applications. An example is each Linflex module instance can be allocated to different cohorts and even the allotment can vary across applications. Hence these need to be individually isolatable and its resets and clocks controlled individually such that if any fault or error occurs in these individual HW components, they can be quiesced, isolated, its clock stopped and as a final local recovery, its reset asserted with no impact to rest blocks of that cohort and as an extension, no impact on any applications running across different cohorts in the system. As an example, in "Fig. 3", the left part of table shows the HW view where, as an example, we have individual HW resources, like different cores, different input/output peripherals, multiple DMA engines and DMA channels as well as multiple system accessible memories. Each one is identified with a unique index so as to be able to allocate it individually to cohorts / virtual machines (VMs). The right-hand side of the table shows the SW application / cohort view.

So, in one application (0), cohort 3, highlighted in light blue, is composed of one real time core, DRAM0 , I2C_1 instance , and EDMA0 . The cohort could simply be involved in fetching data through I2C input path, transferring the data from I2C embedded FIFOs to DRAM , processing it by real time core and communicating out the processed data through I2C peripheral. In another configuration, highlighted in green , secondary M7 core , UART_0, eDMA_3 and a secondary SRAM are grouped as resources needed to perform SW tasks of cohort 1 of application 1 .

HW view (Fixed mapping in RTL)				Virtualization view (SW configurable, dynamic)	
	Lowest Granularity HW resource	HW unique index		Cohort / VM Mapping (Application 0)	Cohort / VM Mapping (Application 1)
Core	Main boot core	0		0	0
Core	Security core	1		1	0
Core	Secondary core	2		2	1
Core	Real time0 Core 1	3		3	2
Core	Real time0 Core 2	4		4	3
Core	……				
Core	Real time3 Core 2	7	⟷ HW to SW mapping	7	6
Peripherals	I2C_0	18		0	3
Peripherals	I2C_1	19		3	2
Peripherals	UART_0	25		7	1
Peripherals	UART_2	26		2	6
Peripherals	DSPI_1	33		4	2
Peripherals	DSPI_3	35		6	3
Peripherals	eDMA_0	47		3	6
Peripherals	eDMA_3	49		2	1
Peripherals	MRU3	64		4	6
Peripherals	……				
Mem	Secondary-SRAM	112		7	1
Mem	Real Time DRAM0	114		3	2
Mem	……				

Figure 3. HW resource indexing and dynamic mapping to cohorts across varying SW applications

III. VERIFICATION OPPORTUNITIES AND INNOVATIONS AROUND ARCHITECTURE TENETS

A. Verification enhancements to enable high quality VDK /Digital Twin for customer

A simple example of verification enhancement done around this was to ensure that all register description and basic protocol (e.g., CAN Tx, Rx) verification was prioritized to complete much earlier. This involves exercising the various register fields of a HW module for write, read back as well as negative checks around reserved or invalid spaces of registers or different types of accessibility (write-only, read-only etc).
Extension of this is ensuring reuse and establishing equivalence of the developed module register and functionality APIs across platforms – RTL IP Verification, RTL SOC Verification, FPGA, Emulation /Accelerated Verification, Pre-Silicon Validation, Post-Silicon Validation, Applications and SW teams, OEM SW development teams, and customer VDK (Virtual Development Kit). "Fig. 4" describes this comprehensive approach. The virtual prototype for our latest SDV Automotive chip has been delivered to a leading OEM and its Tier1's supply chain about 2 years before the production sample delivery dates to customer, as shown in "Fig. 5". This enabled the OEM to shift-left their software development and accelerate the development of SDV capabilities. There are working demonstrations available for the efficiency of these VDKs. High confidence is available from SW development cycles over test vehicles samples of the Automotive SOC OEMs and customers.

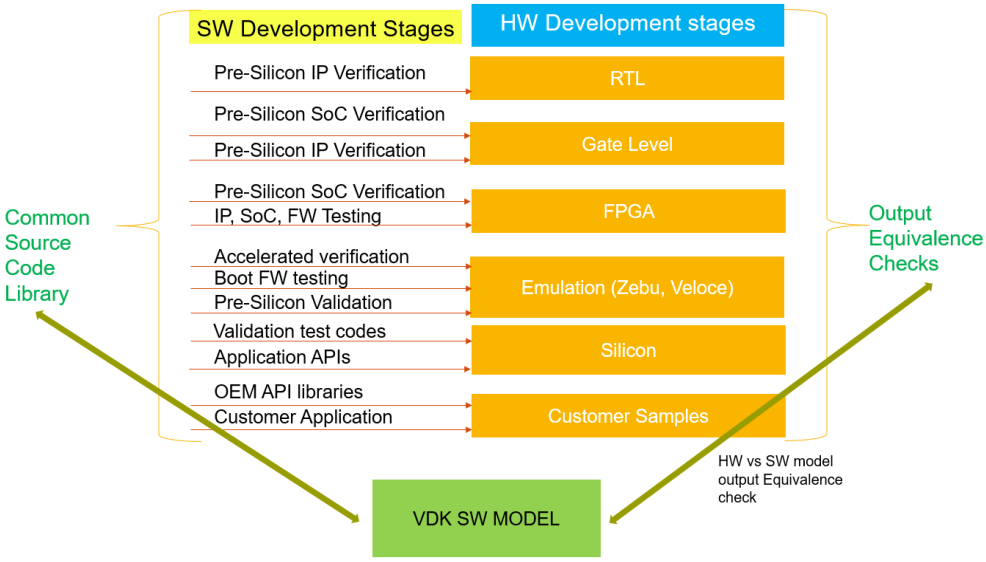


Figure 4. HW and SW development stages: Equivalence checks across HW stages and vs SW model

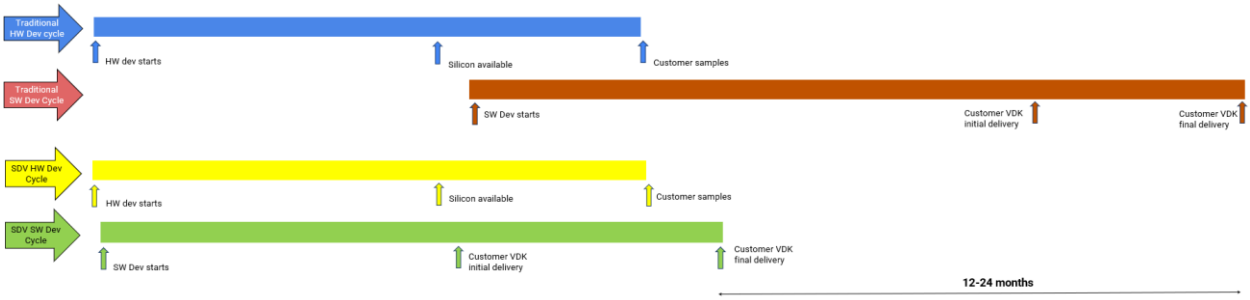


Figure 5. HW and SW development stages: Traditional vs. SDV development cycles

B. *HW-SW mixed control use-cases*

An example of verification enhancement is handling of fault recovery in a graded approach with SW handling being the default for most of the faults in system. In SDV approach, with dynamic virtualization, intent is to diagnose and resolve faults by SW as the default handling. This is delegated to SW safety APIs as we want to maximize the availability and applications running on rest of the HW resources. Only in case a localized, SW only fault recovery is not sufficient, do we move to next grade which could be a module reset. In case even that doesn't suffice, and fault may have propagated to beyond a module boundary, we can escalate to a cohort reset based recovery and finally a system reset. The SW control and diagnostics decrease with each of these grade escalations and the system availability too decreases. This is shown as a simplified representation in "Fig. 6". Verification enhancement accounts for all such HW-SW recovery possibilities with fault injection combinations targeting faults on single module, multiple in module, multiple in cohort , multiple across cohorts etc.

SW- HW handling of fault recovery

Example			
Single bit ECC (restricted to a module)	Grade 0	Interrupt handler SW based fault /error clearing	SW
Multi bit ECC (restricted to a module)	Grade 1	SW triggered HW reset (SRD)	SW + HW
Multi bit ECC (across modules of a cohort)	Grade 2	Escalation to cohort reset	SW + HW
Multi bit ECC (in system shared resource like DMA)	Grade 3	System reset	HW

Figure 6. Example of mixed handling of faults through SW and HW increasing verification combinations

As a quantification of the fault coverage , close to 256 fault sources were verified on the latest SDV SoC with safety recovery across all 4 possible reactions. Creation of automated, modular APIs from machine readable requirements and reuse of safety sequence checks helped make the flow efficient. Feedbacks to architecture and design made overall solution more SW friendly , robust and standardized .

Another example is a feature of analog clock source trimming. Traditionally the trim values have been implemented as a purely HW feature , being burned into metal plugs or fuses and updated only on a reset cycling. But the increasing need to have SW controllability and from the desire to be able to tune and calibrate the devices as the HW ageing happens after years of on-field usage, arose the requirement of SW trims for clocks (and similar analog circuitry) even in automotive field to allow OTA to upgrade the vehicle functionality or even at times to extend the life by tuning some degrading functionality. Verification stressing now involves the HW-SW combined verification with iterations of HW trim interleaved with SW trim controls, values and triggering mechanism. This also includes negative checks with random reset assertions at different cycles of execution to see retention of trimming, transient frequencies, settling behaviors and prioritization between the interoperating schemes.

An added benefit from this pre-silicon verification sign-off confidence is also critical tester and validation time reduction on Silicon through faster SW configured characterization across possible trim values without going through reset cycles. This saves testing time from days to few minutes.

C. *Verification around Freedom from Interference*

As shown in "Fig. 7", the components inside the dotted line boundary represent elements outside the cohort definition and hence should have freedom-from-interference from the elements outside the boundary (i.e., the cohort elements). Hence there needs to be isolation between the within-cohort and outside-cohort elements, even if they are part of same HW physical partition. Secondly there needs to be isolation capability even within elements of same cohort in case fault occurs in one granular HW resource of a cohort as we want to restrict the propagation of fault in one HW resource of the cohort to other resources of the same cohort too to ensure high availability.

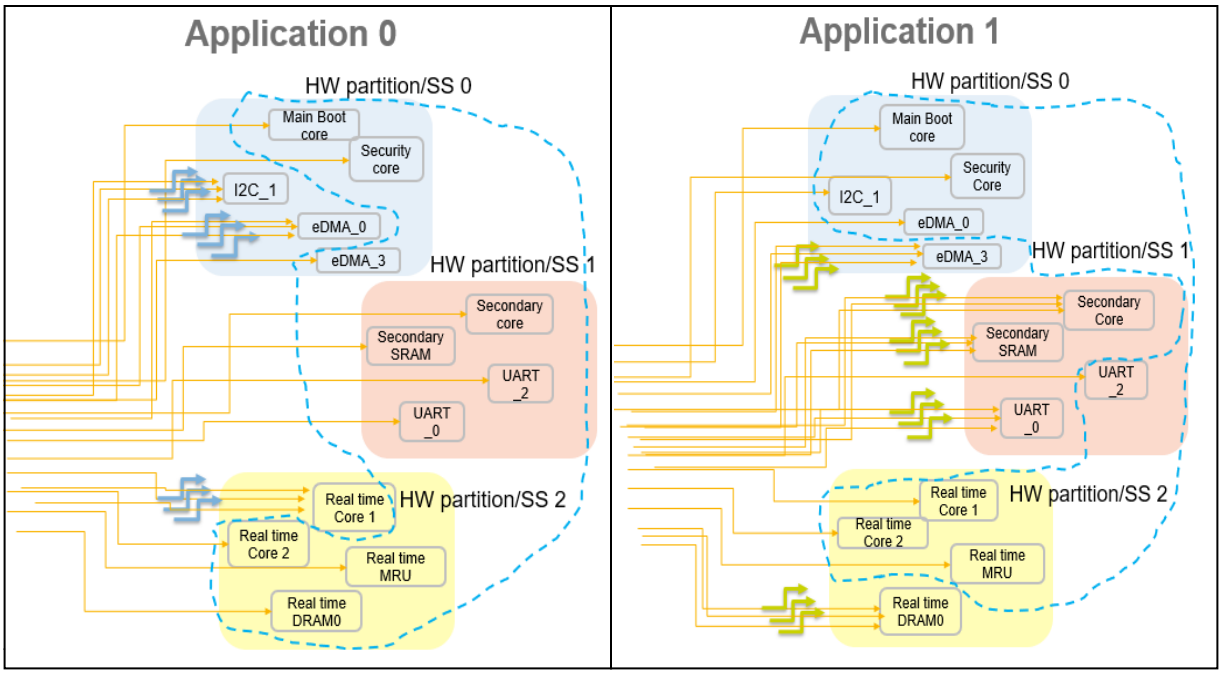


Figure 7. Grouping of HW resources in a particular cohort (Applications 0 and 1)

Lastly, we should be able to control configuration bits and read back the status for all components of a cohort in a grouped way (even if these components are spread across separate physical partition of the chip). This aids SW efficiency and API latency reduction. The arrows can be envisioned as the isolation, clock , reset controls .

Verification enhancement involves the following :

1) Ensuring modular, granular tasks for each of these individual HW resources for basic enablement and functional protocol configuration (e.g., clearing the freeze bit of Flexcan and configuring the FIFOs) .
2) Verification APIs for ensuring granular controls for isolation enable/disable, clock start/stop and SW controlled reset assertion/de-assertion along with the diagnostic status reads for all these controls.
3) Dynamically variable tasks for clubbing individual APIs up to create cohort definition. This grouping and cohort mapping needs to be dynamic according to application being executed.
4) Fault injection tasks for generating faults in cascaded way – starting with individual HW resource, then multiple resources of a cohort, across cohorts of application , across applications of a system and so on.
5) Fault reaction and recovery tasks and checks to ensure that the localization and recovery of fault can happen at the lowest granularity and purely by SW. Interrupt based fault handling, SW reconfiguration of module, etc are possible ways to recover by SW.
6) Ensuring escalated recovery tasks can be executed in case local SW recovery doesn't work, like resetting individual HW resource, or cohort reset, or application reset, or multiple application reset, or final system reset which can again be functional , destructive, power on reset grades.
7) Dynamic, use-case aligned sign-off : Initiate data and control traffic across combinations of dynamic cohorts with fault injection and functional coverage monitoring to ensure excitation of each granular HW resource, interconnects and sidebands between modules and cohorts. This is crucial to rule out hangs, spurious design status capturing and unexpected interferences across cohorts or even applications.

To quantify the verification sign-off scope, the SDV SoC which was targeted for sign off comprised of 20 CPUs, 25 unique memories and 445 peripherals. SDV aligned tuning of verification strategy and methodologies, and reuse of

the modular and automated APIs and checks helped conquer this seemingly vast and challenging scope faster and with high efficiency and quality.

Additionally, there are structural (formal tool based) checks to verify that:

1. There are isolation blocks sitting at the boundary of each individual HW resource which can be dynamically allotted to a cohort. Any missing isolation at any signal is reported as a failure .
2. Correctness of value of each of these isolation blocks. As a rule of thumb, the isolation or safe stated value should be same as the reset value of signals, barring known exceptions like reset signals, transfer error, transfer wait, parity type signals. Violations against expected golden value is reported as an error .
3. These isolation presence and isolation value checks are also done at each SW resettable domain (SRD) boundary to ensure that any fault does not propagate to other cohorts and applications.

Example data from formal runs during early design development (allowing robust design through early fix):

- Total SRD in a particular HW partition – **28** (check extended eventually to all **315** SRDs)
- SRD Analyzed through script- **26**. 2 SRDs were clubbed by architecture with system reset (flow feedback).
- Count of signal reported from script –
    a. 1433 incorrect isolation values were reported from 26 SRD domains.
    b. 599 Missing isolations were reported from 26 SRD domains.
- Signal reported from script – analyzed [dispositioned / ticket added on design]
    a. Incorrect Isolation values [Total 1433]
        i. 622 incorrect isolations waived off (isolation value intentionally different from reset) .
        ii. 811 incorrect isolation values need design correction.
    b. Missing Isolation [Total 599]
        i. 312 safe stating handled differently (as part of Logic BIST safe stating).
        ii. 287 missing isolation blocks need design correction.

A summarization of sample output from flow is shown in “Fig. 8”.

Reset Domain	Isolation present (Y/N)	Safe Stating Check
POR_RESET[main]	N	N
POR_RESET[aon]	N	N
DEST_RESET[main]	N	N
DEST_RESET[aon]	N	N
FUNC_RESET[main]	N	N
FUNC_RESET[aon]	N	N
SRD_SSI	Y	Pass
SRD_DEBUG	N	N
SRD_REST	N	N
SRD_DMA	Y	Pass
SRD_AUXMODULE	Y	Pass
SRD_QSPI	Y	Fail
SRD_uSDHC	Y	Pass
SRD_RTC	No Module	N
SRD_SDB	No Module	N
SRD_MAIN	No Module	N
SRD_DSPI_0	Y	Pass
SRD_DSPI_1	Y	Pass
SRD_LIN_0	Y	Pass
SRD_LIN_1	Y	Fail
SRD_LIN_2	Y	Pass
SRD_FLEXCAN_0	Y	Pass
SRD_FLEXCAN_1	Y	Pass
SRD_FLEXCAN_2	Y	Pass
SRD_UART_0	Y	Pass
SRD_UART_1	Y	Pass

Figure 8. Formal Verification Output of Flow to check isolation completeness and correctness

IV. RESULTS

Major successes achieved with the SDV-Aware Verification are as follows:

1. VDK (Virtual development kit) based sampling of digital twin models of HW and shift-left of register programming models, basic protocol API creation and stitching of granular APIs into SDV use-cases.
    a. OEM/Customer SW development on VDK could start parallel to HW development and **1-2 years** faster compared to traditional automotive SoCs.
2. Ensuring sufficient SW options over HW functionality and SW-HW interleaved controls.
    a. High confidence signoff of **256** fault sources with **4** possible reactions each (mix of SW and HW).
    b. SW trim confidence (**minutes**) over HW reset cycling-based testing and validation (**days**).
3. Signing-off freedom from interference across multiple , dynamic , application cohorts utilizing the services of HW resources of chip.
    a. Verification APIs and checks on SoC comprised of **20** CPUs, **25** memories and **445** peripherals .
    b. Isolation checks across **315** Software Defined Reset Domains for presence and value.

V. CONCLUSIONS

This paper presents the SDV tenets and the next generation of automotive industry requirements due to zonal architectures around SDV. It discusses new verification challenges which were converted into opportunities and also deployed into our latest product solutions. The complex , vast scope associated with HW-SW interoperability has been quantified and conquered through modular, reusable APIs, verification tasks and a combination of dynamic, structural and formal methods helping in exhaustive sign-offs.

The advantages gained out of these ideas in action are encouraging and helped us maximize the potential of HW-SW interoperability. Through early sampling of the SW configurability over HW , the confidence in the deliveries to customers was vastly improved and OEM/customers could be enabled with solid VDK along with silicon samples.

Future work on the SDV aware verification will explore replacing representative bus traffic or use-case profile modelling during fault injection or isolation checks with actual customer SDV APIs around their use-case scenarios. This will be done in collaboration with OEMs and customers to make the sign-off more realistic and robust. Additionally, further stressing will be planned over the HW-SW combinations of chip functionality, including any conflict or primacy resolution between the two. Additional confidence would also target by sweeping asynchronous reset, error, fault or interrupt events while system is executing multiple applications across multiple cohorts to mimic actual vehicle behavior for HW-SW interoperability confidence.

ACKNOWLEDGMENT

I would like to acknowledge the various team members at NXP, who contributed to the successful deployment of the SDV-Aware verification on the state-of-the-art automotive processor.

REFERENCES

[1] Victor Bandur, Gehan Selim, Vera Pantelic, and Mark Lawford, "Making the Case for Centralized Automotive E/E Architectures", IEEE Transactions On Vehicular Technology, Vol. 70, No. 2, February 2021

[2] Alessandro Frigerio, Bart Vermeulen, Kees G. W. Goossens, "Automotive Architecture Topologies: Analysis for Safety-Critical Autonomous Vehicle Applications", IEEE Access, Volume: 9, April 2021

[3] Manfred Broy, Ingolf H. Kruger, Alexander Pretschner, Christian Salzmann, "Engineering Automotive Software", Proceedings of the IEEE, Volume: 95, Issue: 2, February 2007

[4] Houssem Guissouma, Carl Philipp Hohl, Fabian Lesniak, Marc Schindewolf, Jürgen Becker, and Eric Sax, "Lifecycle Management of Automotive Safety-Critical Over the Air Updates: A Systems Approach", IEEE Access, Volume: 10, May 2022

[5] https://www.nxp.com/products/processors-and-microcontrollers/s32-automotive-platform:S32

[6] https://www.synopsys.com/verification/solutions/automotive/automotive-ecosystem-collaboration.html