2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

Who watches the watchman? FuSa Verification of DCLS Configuration through Formal and Static Checks

Avinash Kumar Pandey, Srobona Mitra, Sathish Kumar Manickam, Arunava Dutta, Deepak Baranwal
{avinpand, srobmitr, satishku, arundutt, dbaranwa}@qti.qualcomm.com
Qualcomm India Private Limited

***Abstract*- Dual Core Lock Step (DCLS) is one of the most common system configurations that is followed to ensure defect-free operation for Functional Safety (FuSa) of automotive systems. However, each implementation flavours of DCLS configuration need to be verified to ensure it can detect and report the defects in the field. In this paper we propose an automated DCLS verification package consisting of a set formal and static checks to perform a comprehensive verification of the DCLS configuration.**

We also present the results on applying the proposed verification package on several in-house designs to prove its effectiveness.

I. Introduction

While delivering a product for safety-critical application, manufacturers must abide by several safety-standards, like EN50129 for railways, ASIL-A/B/C/D [1], and ISO 26262 [2][4] for automobiles. Automotive Safety Integrity Level **(**ASIL) implementation requires a systemic process for design and verification to help safety coverage. Dual Core LockStep (DCLS) [3] provides 100% coverage of the intended module in safety analysis therefore requires processes in place to verify it. There are safety error injection techniques using random injection to measure efficiency of safety mechanisms and require a large framework. With growing design complexity it's needed to have early feedback to designer to avoid late ECO implementation.

In DCLS configuration, a replica of the central computational module is used (shown in Figure 1 as Main_Module and Redundant_Module). The two instances of the module are fed with the same inputs and their corresponding outputs are compared to detect any discrepancy. Delay F/F (as shown in Figure 1 by D block) are often added at specific places to mitigate effect of glitches in input signal and hence avoiding common cause failure. During fabrication if any defect gets injected into any of the two instances and thereby deviates its functionality, the XORing of the outputs of the instances detects the defect. It is subsequently notified through the FuSa F/F. Since the two modules are placed spatially apart from each other, the probability of same defect appearing at both the instance is very less. Thus, it provides a strong assurance that only good outputs are propagated to the down-stream logic.

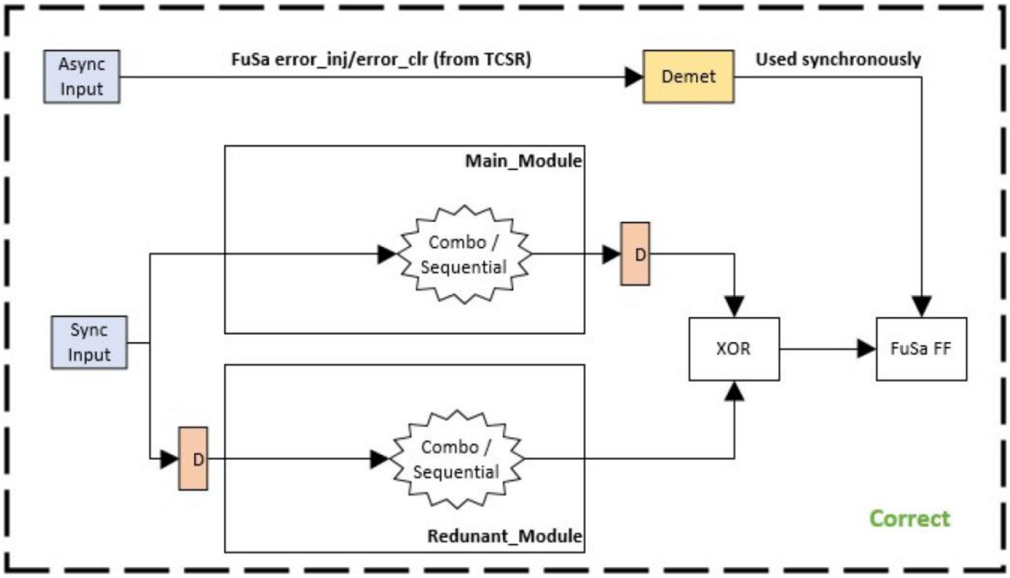
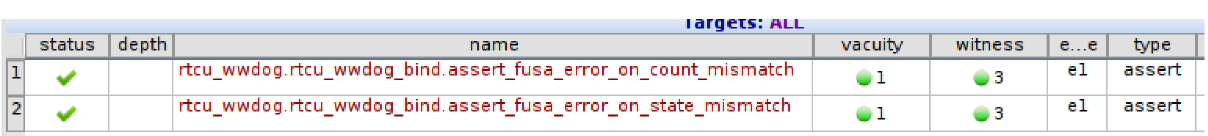


Figure 1: DCL*S* Configuration

II. IMPLEMENTATION

<u>DCLS Verification Package</u>: Though a simple output-comparing configuration, the verification challenges are manyfold. Achieving a 100% coverage of all the ports through input pattern generation and subsequent simulation is quite a challenge. Additionally, the lockstep instance can be deeper in design hierarchy providing less control for these ports. Hence, we are proposing a focused verification package capable of carrying out a complete verification of the DCLS configuration to flush-out bugs in the RTL stage. Following is a list of verification checks that are carried out on every DCLS configuration to ensure its proper functioning.

- <u>FuSa check</u>: With the comparator inputs made free, end-to-end formal property-based checks verifies the functioning of the XOR block as well as the FuSa FF (shown in Figure 2)



Targets: ALL

	status	depth	name	vacuity	witness	e...e	type
1	✔		rtcu_wwdog.rtcu_wwdog_bind.assert_fusa_error_on_count_mismatch	1	3	el	assert
2	✔		rtcu_wwdog.rtcu_wwdog_bind.assert_fusa_error_on_state_mismatch	1	3	el	assert

Figure 2: FPV-based checks for XOR block and FuSa F/F

- <u>Delay checks:</u> Formal connectivity checks are carried out to ensure that the amount of delay in the following paths is same as the delay of the DCLS
  - DCLS input to the redundant module input
  - The main module output to the XOR input

- <u>Demet checks:</u> Multiple CDC checks are deployed to check whether the asynchronous inputs coming to the XOR gate from main module and redundant module are synchronized with proper synchronizer cells.

- Reset checks: A combination of formal properties and connectivity properties are used to verify the reset configuration.
    - Formal properties
        - The FuSa reset should not be de-asserted until the resets of its backend blocks have been de-asserted
        - The delay between the FuSa reset de-assertion and resets of its backend blocks de-assertion, should not exceed the DCLS delay amount
    - Connectivity properties (shown in Figure 3)
        - Async_reset → delay F/Fs at the output of main and input of redundant module
        - Async_reset → main and redundant modules
        - Async_reset → FuSa F/F
        - Delayed reset to the redundant module
        - Delayed reset to the FUSA F/F

Targets: ALL

	status	name	vacuity	source_expr	enable_expr	destination_expr	structural	path_delay
1	✓	...rr_comparator_count_11	1	u_rtcu_wwdog_core.o_count	1'b1	u_rtcu_wwdog_err_comparator.count	YES	0
2	✓	...err_comparator_state_8	1	u_rtcu_wwdog_core.o_state	1'b1	u_rtcu_wwdog_err_comparator.state	YES	0
3	✓	...tor_count_redundant_10	1	u_rtcu_wwdog_core_redundant.o_count	1'b1	u_rtcu_wwdog_err_comparator.count_redundant	YES	0
4	✓	...rator_state_redundant_9	1	u_rtcu_wwdog_core_redundant.o_state	1'b1	u_rtcu_wwdog_err_comparator.state_redundant	YES	0

Figure 3: Connectivity Checks in DCLS Verification package

Figure 4 gives an overview of the proposed formal and static checks and their application site.

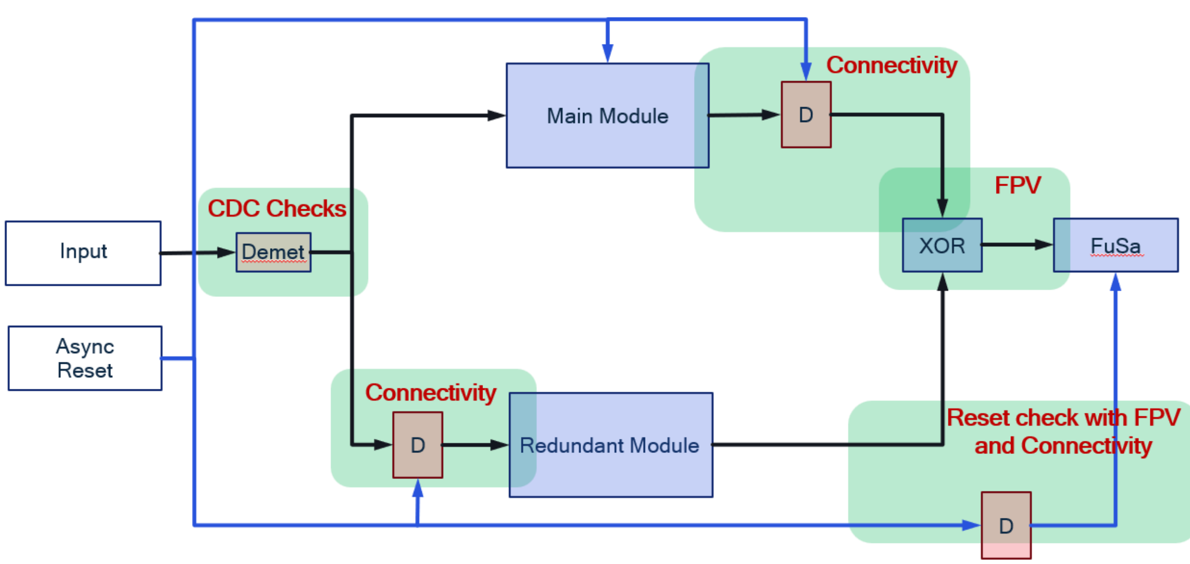


Figure 4: DCLS Verification checks

III. RESULT

We have successfully deployed the proposed verification package on multiple designs including a watchdog implementation and designs handling safety critical tasks in the flagship automotive processors. Our proposed package has also shown its capabilities in catching some known bugs present in some older designs.

.

IV. CONCLUSION

We are proposing a verification package for comprehensive verification of DCLS configuration through formal and static checks. We have successfully deployed our proposed verification package in several in-house designs. It has brought down the effort of verification of DCLS configuration, while increasing the quality of the same. As a future work will be further integrating this automated verification package it into our verification environment for seamless application and effective release qualification.

REFERENCES

[1] Synopsys: “What is ASIL (Automotive Safety Integrity Level)?” URL: https://www.synopsys.com/glossary/what-is-asil.html.
[2] ISO: “ISO 26262-1:2018 Road vehicles — Functional safety” URL: https://www.iso.org/standard/68383.html.
[3] Codasip: “Dual-core lockstep” URL: https://codasip.com/glossary/dual-core-lockstep/.
[4] embitel: “[Vlog – ISO 26262 Standard for Functional Safety] How to Evaluate Hardware Architecture Metrics Using FMEDA Method” URL: https://www.embitel.com/blog/embedded-blog/iso-26262-standard-for-functional-safety-using-fmeda-method.