2025 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE
MUNICH, GERMANY
OCTOBER 14-15, 2025

Exploring the Limits of Vertical Reuse Automation in PSS-Driven SoC Verification

Petr Bardonek, Brno University of Technology, Brno, Czech Republic (ibardonek@fit.vut.cz)
Alessandra Dolmeta, Politecnico di Torino, Torino, Italy (alessandra.dolmeta@polito.it)
Marcela Zachariášová, Brno University of Technology, Brno, Czech Republic (zachariasova@fit.vut.cz)
Guido Masera, Politecnico di Torino, Torino, Italy (guido.masera@polito.it)

***Abstract*—As System-on-Chip (SoC) designs integrate increasingly diverse and software-driven components, verification reuse becomes critical for managing complexity across block, subsystem, and system levels. Despite advances in automation, the theoretical and practical boundaries of vertical reuse in complex SoC integrations remain underexplored. This paper investigates the limits of vertical reuse in Portable Stimulus Standard (PSS) workflows by applying a static analysis–based toolchain to a real-world SoC design. In PSS, reusable verification intent is captured as Portable Models (PMs), which combine abstract scenario definitions with realization-layer bindings to design interfaces. While prior work demonstrated that static analysis enables vertical reuse on designs of various sizes and complexities, this study examines how far such reuse can be extended as integration progresses toward full SoC levels. Using a Keccak-based cryptographic accelerator integrated through two distinct architectures — loosely coupled and tightly coupled — the feasibility of vertical reuse across six integration contexts is evaluated. The results show that static connectivity analysis supports reuse across hierarchy levels but that SoC-level reuse increasingly intersects with software-driven control, requiring additional modeling effort. This case study highlights both the reach and the limits of structural automation in enabling PM reuse, providing insight into when automation suffices and where additional modeling effort is required.**

***Keywords*—PSS, Digital Design, Static Analysis, Data Flow, Control Flow, SMT, SystemVerilog, Simulation-based Verification**

I. Introduction

As modern System-on-Chip (SoC) architectures integrate a growing variety of configurable Intellectual Property (IP) blocks, the complexity of functional verification continues to increase. With accelerators and subsystems frequently reused across multiple designs and variants, verification reuse becomes a key concern for engineering teams. Verification strategies must increasingly support layered approaches, spanning block-level verification, subsystem integration, and full SoC verification. In this context, enabling the reuse of verification intent and infrastructure across hierarchy levels becomes critical to managing verification efforts.

The Portable Stimulus Standard (PSS) [1] was introduced to provide a portable, model-based approach for expressing verification intent in a declarative form as Portable Models (PMs), enabling the generation of constrained stimuli derived from modeled scenarios. PSS aims to enable reuse across platforms (technique reuse), design variants (horizontal reuse), and design hierarchy levels (vertical reuse). PMs separate verification intent from implementation through a logical division into modeling and realization layers. The modeling layer expresses intent as abstract scenarios, while the realization layer provides the infrastructure for their execution on the Design Under Verification (DUV).

While existing tools [2-6] and research have focused heavily on modeling layer features [7], the realization layer—particularly in vertical reuse scenarios—still requires significant manual effort [8]. Block-level PMs can often be reused at higher levels of the design hierarchy, with their modeling layer remaining valid. However, their realization layer must be adapted to reflect new integration contexts, a process that depends on understanding how structural and control connectivity change across hierarchy levels. This adaptation forms a practical bottleneck for engineers and limits the scalability of vertical reuse in large designs.

To address this challenge, the authors developed VerTrace [9], a static analysis–based toolchain to support vertical reuse within PSS. The toolchain analyzes SystemVerilog RTL designs to trace connectivity across hierarchy levels and generates realization-layer artifacts. While its applicability has been tested on various designs [10], it has not yet been applied at the full SoC scale. This paper presents a case study that demonstrates

the application of the approach and examines the boundaries of automated vertical reuse at the SoC level. It focuses on a cryptographic hardware accelerator based on the Keccak algorithm, which is security-relevant and reused across multiple SoC variants. The Keccak accelerator implementation used in this study is part of an SoC architecture developed within the European ISOLDE project [11]. Results from applying the toolchain to two different integration approaches—loosely coupled and tightly coupled—are presented, highlighting examined reuse boundaries, modeling implications, and practical lessons.

II. Theoretical PSS Background

PSS defines a declarative domain-specific language for modeling scenario spaces and generating test cases across verification platforms. A PSS model is logically separated into a modeling layer and a realization layer.

The **modeling layer** captures verification intent at a high level of abstraction from which verification tests and stimuli are derived. Actions represent a unit of behavior, while passive entities such as data items, resources, and states provide the objects that actions operate on. Constraints and scheduling constructs refine the scenario space, defining legal combinations and orderings. This declarative style enables engineers to express what must be verified without manually coding all possible sequences.

The **realization layer** provides the link between the modeling elements and concrete implementations, enabling the execution of modeled scenarios on the DUV. Abstract actions are mapped to implementation artifacts through *exec body*, which may call SystemVerilog tasks, Universal Verification Methodology (UVM) sequences, or software routines. For example, a write action may be realized by a SystemVerilog task performing a register write or by a UVM sequence accessing an Advanced Peripheral Bus (APB, Figure 1).

This separation preserves the modeling intent independently of execution details, enabling the same scenario description to be reused across platforms and integration contexts.

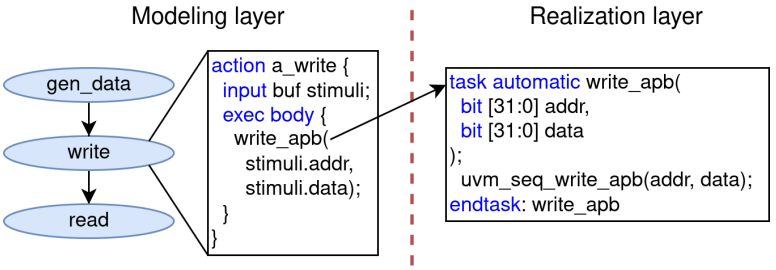

Figure 1 - Example of linking a modeling element to a realization.

III. VerTrace Toolchain

The VerTrace toolchain [9] was developed by the authors as an open-source framework to support vertical reuse within the PSS. It integrates both existing open-source and custom-developed tools in a staged flow and automates key aspects of realization-layer adaptation. The toolchain performs static analysis of RTL connectivity across hierarchy levels to assist verification engineers in reusing existing PMs as integration progresses from block to subsystem and potentially full SoC contexts. By automating signal tracing and condition analysis, the toolchain improves scalability and reduces the manual engineering effort required for reuse in structurally complex designs.

The toolchain (Figure 2) operates in two main parts: static analysis and PSS structure generation. During static analysis, SystemVerilog RTL source files are parsed using the Verible [12] open-source parser, which produces a syntax tree that is then transformed with a custom preprocessor into an internal representation. The data flow analyzer (DFA) traces signal propagation, constructing a data flow graph (DFG) for each signal, capturing paths to its potential drivers. Then, for each instantiated submodule, the control flow analyzer (CFA) analyzes the DFGs of signals that influence its inputs. This includes both signals that are drivers and those that appear in conditions along the propagation paths. If such condition signals are themselves conditionally driven, the CFA recursively expands its analysis to include their DFGs. The results are used to build a connectivity map within the selected integration context. For conditional paths, the CFA collects branching conditions and invokes

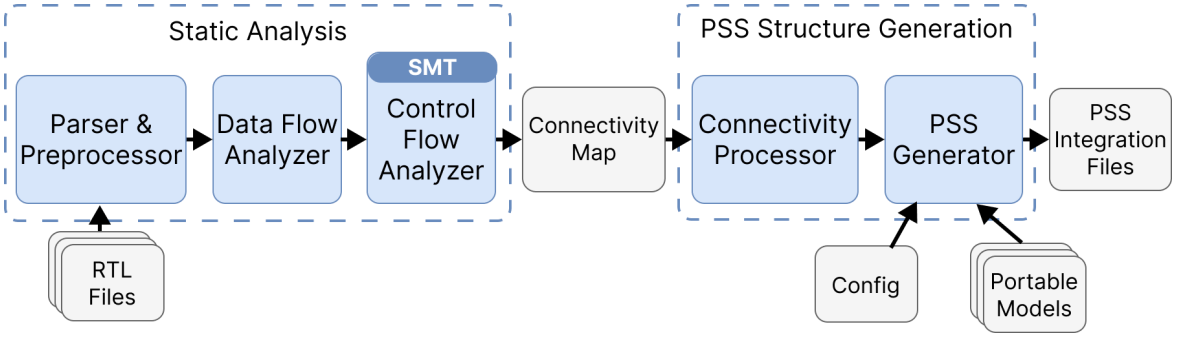

Figure 2 - Overview of the VerTrace toolchain flow for PSS vertical reuse.

the Z3 SMT solver [13] to generate assignment models, which capture the input combinations under which propagation occurs. The resulting connectivity map expresses both unconditional and conditionally enabled paths of signals related to the driving of inputs of submodules.

In the PSS structure generation phase, the connectivity information is processed by the connectivity processor, which derives interconnections as PSS constraints that mirror the RTL-level signal relationships. These constraints enable the reuse of existing PMs in new integration contexts, mapping the signals they reference to those available at the higher level. Alongside these constraints, the toolchain also takes configuration settings and existing portable models as inputs, generating additional realization-layer artifacts that serve as the structural backbone for assembling a new PM. This supports vertical reuse by enabling composition without re-modeling existing PMs' internals.

IV. Case Study: Keccak Accelerator

Among modern cryptographic primitives, the Keccak family is notable for its unconventional design and strong resilience against known attacks. At the heart of Keccak lies the Keccak-*f* permutation, which processes a 1600-bit state arranged as a 5 × 5 grid of 64-bit words. This state results from combining the bit-rate *r* and capacity *c*, parameters that define the level of security and throughput. The algorithm was selected by NIST as the foundation of the SHA-3 standard, succeeding SHA-2 after an international competition [14].

The Keccak-*f* transformation operates through 24 consecutive rounds, each composed of five fundamental functions: θ (theta), ρ (rho), π (pi), χ (chi), and ι (*iota*). Together, these steps provide diffusion, introduce non-linear operations, and guarantee resistance against both differential and linear attacks [15]. Because this permutation underpins not only SHA-3 but also a wide set of NIST-selected Post-Quantum Cryptography (PQC) proposals, it represents a relevant benchmark for accelerator design. Efficient hardware realizations of Keccak are therefore essential to meet the stringent performance and security demands of next-generation PQC systems.

For this work, we adopted a simplified Keccak unit that was integrated into X-HEEP (eXtendable Heterogeneous Energy-Efficient Platform), a SystemVerilog-based open-source RISC-V microcontroller [16]. Two integration paths were explored: a loosely coupled variant (memory-mapped) and a tightly coupled variant relying on the Core-V eXtension (CV-X-IF) interface, as shown in Figure 3.

In the loosely coupled configuration, the accelerator behaves as a standard peripheral accessible via the system bus. This solution maximizes design flexibility, allowing the connection of domain-specific hardware modules with custom memory, interrupt, and power management ports. Conversely, the tightly coupled design

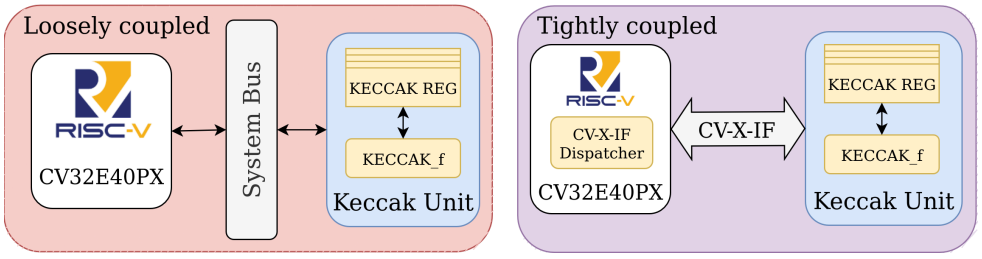

Figure 3 - Schematic of Keccak's loosely coupled and tightly coupled integration methodologies.

leverages CV-X-IF, a recently introduced RISC-V extension interface that connects accelerators directly to the processor pipeline. CV-X-IF enables low-latency execution of custom instructions and efficient data handshaking between the CPU and the accelerator. Initially developed for the CV32E40X core [17] and later extended to the CV32E40PX variant [18], the interface introduces a dispatcher that forwards control and data signals to the accelerator. A more detailed description of its integration can be found in [19, 20].

V. Toolchain-Based Reuse Analysis

This section presents the application of the VerTrace toolchain to the two Keccak integration architectures introduced in the previous section. The toolchain was applied step by step, starting from the integration context where the Keccak Unit was instantiated and progressing outward through successive hierarchy levels to evaluate how far vertical reuse of Keccak's PM can be carried.

The results of the static analysis are summarized in Table 1. **Integration contexts** are shown with both their descriptive names and shorthand (e.g., Loosely Coupled – LC1, Tightly Coupled – TC2). The **Signals Traced** column reports the number of signals identified by the DFA, while **Signals Requiring CFA** indicates how many of those signals were further processed by the CFA to resolve connectivity for inputs of submodules.

Verification at the block level was performed using a UVM-based environment for the Keccak Unit. As integration advanced through the design hierarchy levels, new UVM environments were created. In each integration context, VerTrace provided interconnection constraints and structural artifacts that supported the reuse of the original Keccak PM within the new UVM verification environments.

Table I. Static analysis results of integration methodologies

Integration Context	*Signals Traced*	*Signals Requiring Control Flow Analysis*
Loosely Coupled - LC 1	31	30
Loosely Coupled - LC 2	252	148
Tightly Coupled - TC 1	38	12
Tightly Coupled - TC 2	44	36
Tightly Coupled - TC 3	2	32
Tightly Coupled - TC 4	319	142

A. Loosely Coupled Integration

At the first integration context (LC1, Table 1), shown in Figure 4, where the Keccak Unit is instantiated, the toolchain showed that data signals are routed through the Open Bus Interface (OBI) and control signals through APB (Figure 4). DFA traced 31 signals, with 30 requiring CFA to resolve connectivity for driving the inputs of submodules. This high ratio indicates that LC1 acts as the wrapping layer for the accelerator, where protocol logic and routing dominate before the design connects to the top-level SoC.

From a reuse perspective, LC1 highlights an **interface protocol challenge**: accelerator signals are routed through OBI and APB protocol signals, the dependencies of which must be respected. Instead of tracing through the internals of these buses, VerTrace treats them as natural reuse boundaries, where existing bus PMs should be connected. While the constraints generated by the toolchain are able to map the signals referenced in the Keccak

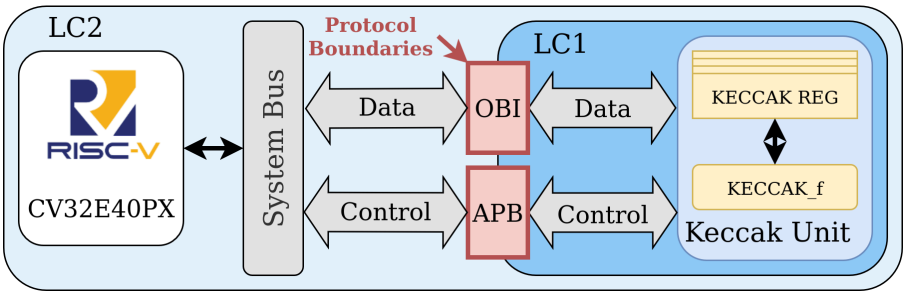

Figure 4 - Analyzed structure of Keccak's loosely coupled integration.

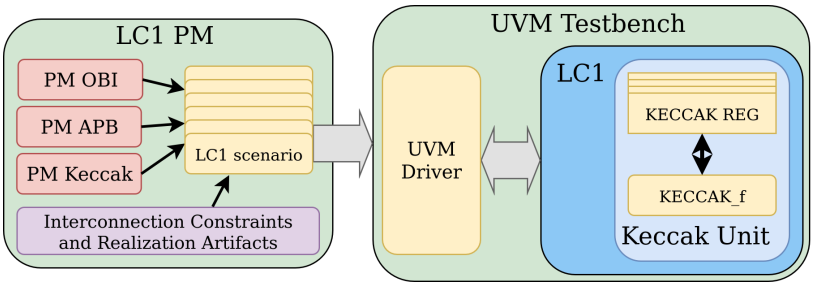

Figure 5 - LC1 PM drives stimuli through the UVM testbench for LC1, reusing existing PMs (Keccak, APB, OBI) integrated via toolchain-generated interconnection constraints and realization artifacts.

PM to LC1 interface signals, they also restrict the set of legal stimuli that can be produced by connected bus PMs. For the LC1 PM (Figure 5), the realization layer is implemented within the UVM testbench, and bus PMs (OBI, APB) drive stimuli generation for LC1 signals based on the generated constraints.

To make this explicit, the toolchain not only outputs structural constraints but also generates PSS actions to match the new interface. For example, action *send* (highlighted by yellow in Figure 6) is generated to call *send_seq()* function with the newly identified top-level interface signals and action *check_done* (highlighted by green in Figure 6) to call the *check_done()* function, both linked to functions implemented in the UVM testbench. To outline the complete workflow in Figure 6, the generated realization layer artifacts bridge the existing Keccak PM with the UVM infrastructure. These artifacts enable the same verification scenarios to execute in the LC1 context without modifying the original PM source, demonstrating practical vertical reuse.

The second integration context (LC2, Table 1), shown in Figure 4, corresponds to the SoC level, where the Keccak Unit is connected to the CPU core alongside other system components. DFA traced 252 signals, of which 148 required CFA to resolve connectivity for submodule inputs. Compared to LC1, the lower ratio reflects additional SoC-level logic that does not directly contribute to driving submodules.

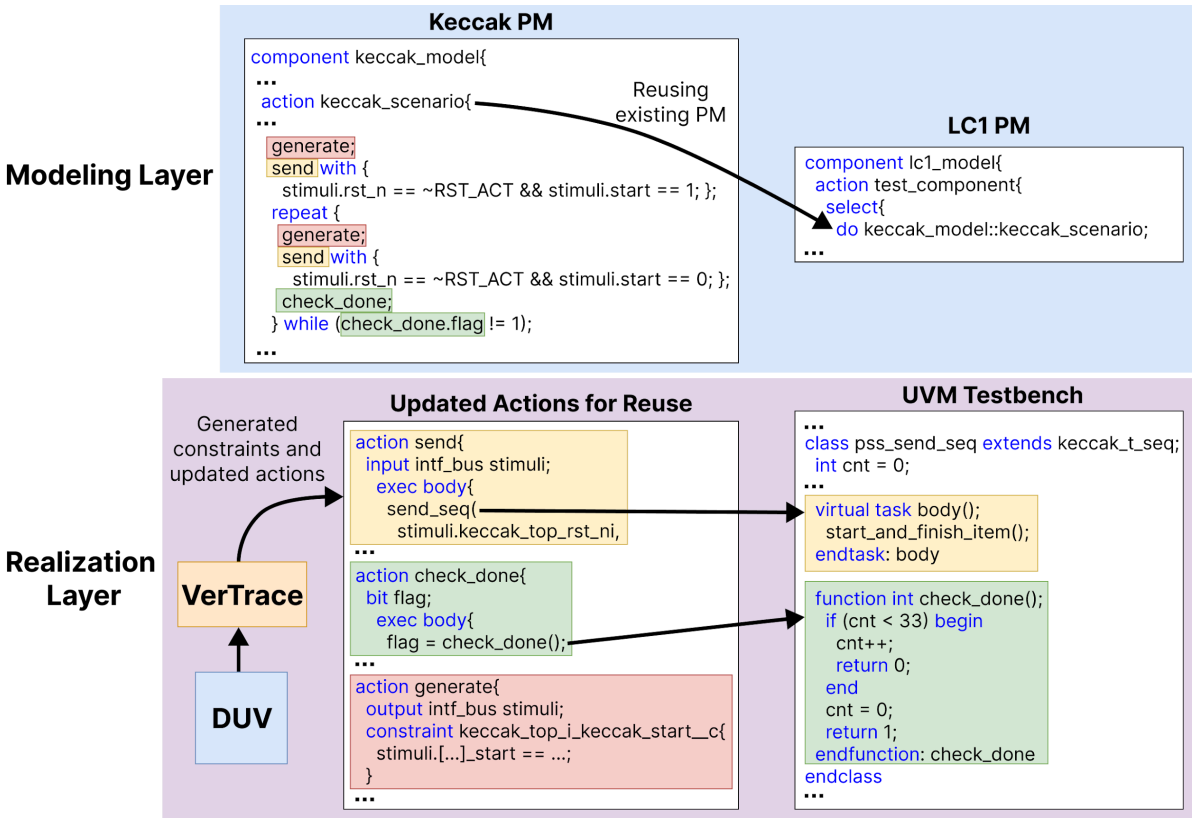

Figure 6 - Vertical reuse workflow showing how VerTrace generates realization layer artifacts that enable existing PMs to be reused in new integration contexts without modification to the original modeling layer.

From a reuse perspective, LC2 is the first point at which a practical limitation appears. While the toolchain still generates interconnection constraints, the way stimuli are applied changes: instead of being driven directly through signal interfaces, many values now come from registers configured by software. The toolchain can identify which of these registers influence inputs at this level and generate constraints for the values they must hold for reuse, but it cannot determine how or when software writes them, nor expose their memory map. The UVM testbench remains usable at LC2, however, with stimuli now driven by software. This transition highlights a **software-driven control challenge**, as structural analysis alone cannot support modeling once register values are set by software.

B. Tightly Coupled Integration

At the first integration context (TC1, Table 1), shown in Figure 7, where the Keccak Unit is instantiated, the toolchain showed that direct instruction information is propagated into TC1 through its proprietary instruction interface and used to determine, depending on the instruction, where to load and store inside the Keccak register file (KECCAK REG of Figure 7) or when to trigger the Keccak permutation. DFA traced 38 signals, with 12 requiring CFA to resolve submodule connectivity. The low CFA ratio reflects that TC1 relies on extracted instruction information rather than interface signals. VerTrace generated interconnection constraints linking TC1 interface signals to those used by the Keccak PM, enabling reuse for stimuli generation without modification, with the UVM environment applying them at this integration context.

At the second integration context (TC2, Table 1), shown in Figure 7, the toolchain identified a controller submodule that orchestrates control and data signals toward TC1. DFA traced 44 signals, with 36 requiring CFA. This high CFA ratio reflects the controller's role in managing accelerator operation. This controller serves as the main integration point for the Keccak accelerator to the SoC through the CV-X-IF interface. From a reuse perspective, this introduces a **module boundary challenge**: the control logic is encapsulated in a dedicated submodule, which is a candidate for separate verification with its own PM. The toolchain then supports the reuse of both the controller and Keccak PMs in the PM intended for TC2.

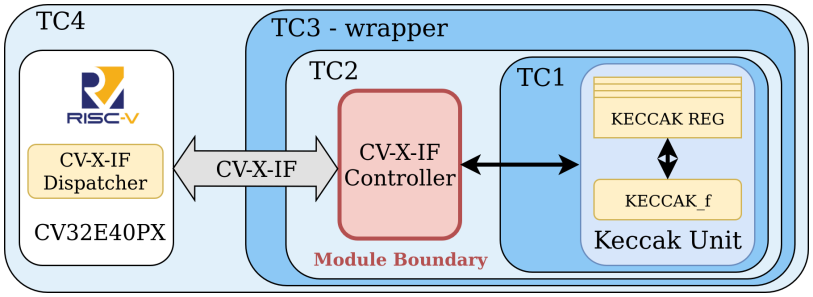

Figure 7 - Analyzed structure of Keccak's tightly coupled integration.

The third integration context (TC3, Table 1), shown in Figure 7, is a wrapper around TC2 that adapts the interface for SoC connectivity. While the DFA counted only two top-level signals (clock and reset), the CFA expanded the SystemVerilog interface bundles on the wrapper ports, which resulted in CFA analyzing 32 signals in total. This illustrates an **abstraction challenge**: structured types can conceal significant activity that still must be resolved. VerTrace's interconnection constraints preserve visibility across this boundary and keep PM connectivity intact.

At the fourth integration context (TC4, Table 1), shown in Figure 7, the Keccak integration connects to the CV-X-IF dispatcher, which offloads instructions from the CPU. DFA traced 319 signals, with 142 requiring CFA. As expected from the tightly coupled architecture, no additional accelerator-specific logic was present. However, TC4 marks the transition to software-driven control: VerTrace can still identify which registers must be configured and generate constraints for the values they must hold, but it cannot determine how or when software writes them. This highlights a **software-driven control challenge**, where structural analysis alone is not enough for vertical reuse. Even so, VerTrace reduces manual analysis effort by systematically pointing out the relevant configuration points, helping engineers adapt the realization layer as verification shifts to software-driven execution.

VI. Comparative Summary and Practical Evaluation

Despite architectural differences, both integration methods offered meaningful opportunities for vertical reuse of the Keccak PM. In the loosely coupled case, protocol boundaries provided natural insertion points for reusing existing models of standard communication protocols. In the tightly coupled variant, a dedicated controller was identified as a key integration point and is a natural candidate for a separate PM, composable with the Keccak PM. The analysis also showed that across both integration methods, none of the traced paths driving submodule inputs were conditional, so no SMT resolution was required. This reflects the SoC practice of encapsulating control logic in dedicated modules, such as the controller in TC2.

Across all contexts, the toolchain provided the interconnection information necessary to support vertical reuse. At earlier integration contexts (LC1, TC1–TC3), it generated the full infrastructure required for reuse, automatically resolving the mappings needed to connect existing PMs into new environments. These results demonstrate that structural analysis can support effective vertical reuse through multiple hierarchy levels. At SoC-level contexts (LC2, TC4), reuse is still feasible, but stimulus driving shifts to software. Here, the toolchain identifies which registers influence inputs, but additional modeling is required to capture software behavior.

The case study identified several challenges that have to be accounted for and addressed when vertical reuse takes place at the SoC scale:

- **Interface protocol challenge**, which comes with indirect signal propagation through shared interconnects such as buses with decoders and arbitration logic, and introduces a layer of abstraction. In such cases, bus PMs can be used for generating stimuli while generated constraints align them with supported functionality.
- **Module boundary challenge**, where submodule inputs are influenced by outputs of other submodules. In such cases, outputs cannot be set directly, but the submodules can be verified separately using their own PMs and then composed with others in higher-level PMs.
- **Software-driven control challenge**, where stimuli are no longer driven through signal interfaces but are applied through running software. While manual modeling is necessary to capture how software sets the values, the analysis pinpoints configuration points and provides constraints for values that must be set for reuse.
- **Abstraction challenge**, where structured types can conceal significant signal activity. The analysis expands these types to resolve connectivity, but this adds overhead and requires explicit handling to preserve PM visibility across module boundaries.

In practice, VerTrace serves as a planning and composition aid for engineers, mapping signals referenced by existing PMs to those available in higher-level contexts, providing artifacts for immediate reuse when possible, and identifying where additional modeling is required.

VII. Conclusion

This paper presented the application of VerTrace, a static-analysis toolchain, to explore vertical reuse in SoC verification using a Keccak accelerator integrated in loosely coupled and tightly coupled architectures. The study demonstrated that structural static analysis can provide the interconnection information needed to reuse existing PMs across multiple hierarchy levels, significantly reducing the manual effort required to trace and map signals.

In practice, VerTrace supported reuse up to the SoC level by generating constraints and structural artifacts that allowed existing PMs to be applied without modification in early contexts and to be composed with new models, such as the controller PM in the tightly coupled case. At higher levels, where stimulus driving shifts to software, the toolchain continued to identify relevant configuration points, providing engineers with a clear picture of where reuse remains straightforward and where additional modeling is required.

The case study also demonstrated that interface protocols, module boundaries, and software-driven control represent recurring challenges that must be accounted for in SoC-scale verification. VerTrace reduces the burden of handling these challenges by automating connectivity analysis and producing artifacts for vertical reuse.

Future work will focus on extending the toolchain with stronger guidance features, making it easier for verification engineers to plan reuse strategies and adapt realization layers efficiently when software or protocol behavior comes into play.

Acknowledgment

This work was supported in part by the project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union – NextGenerationEU, and in part by Brno University of Technology under project number FIT-S-23-8141.

References

[1] Accellera Portable Stimulus Working Group, "Portable Test and Stimulus Standard," 2019. [Online]. Available: https://www.accellera.org/downloads/standards/portable-stimulus

[2] Cadence Design Systems, "Cadence Perspec System Verifier," https://www.cadence.com/en_US/home/tools/system-design-and-verification/software-driven-verification/perspec-system-verifier.html , Accessed: September 2025.

[3] Siemens EDA, "Portable Stimulus with Questa Verification Platform," https://verificationacademy.com/topics/portable-stimulus/, 2025, Accessed: 2025-09-17.

[4] Breker Verification Systems, "Trek Suite for Portable Stimulus," https://brekersystems.com/products/trek-suite/, 2025, Accessed: 2025-09-17.

[5] Synopsys Inc., "VC Portable Stimulus," https://www.synopsys.com/verification/soc-verification-automation/vc-portable-stimulus.html, 2025, Accessed: 2025-09-17.

[6] Agnisys Inc., "Portable Stimulus Compiler," https://www.agnisys.com/solutions/service/pss/, 2025, Accessed: 2025-09-17.

[7] J. Nagar, T. Dworzak, S. Simon, U. Heinkel, and D. Lettnin, "MetaPSS: An Automation Framework for Generation of Portable Stimulus Model," in DVCon Europe 2023; Design and Verification Conference and Exhibition Europe, 2023, pp. 79–85.

[8] P. Bardonek and M. Zachariášová, "Control Flow Analysis for Bottom-up Portable Models Creation," in DVCon Europe 2023; Design and Verification Conference and Exhibition Europe, 2023, pp. 65–70.

[9] P. Bardonek, "VerTrace: Static Analysis Toolchain for Vertical Reuse in PSS Verification," https://gitlab.com/pbardonek/data-control-flow-analyzer, Accessed: September 2025.

[10] P. Bardonek and M. Zachariášová, "VerTrace: an Open-Source Toolchain for Portable Stimulus Vertical Reuse," in 2025 32nd IEEE International Conference on Electronics, Circuits and Systems (ICECS), 2025.

[11] "ISOLDE Project," https://www.isolde-project.eu/, Accessed: 2025-09-17.

[12] CHIPS Alliance, "Verible: SystemVerilog Analysis and Linting Tool," version: verible-v0.0-2789-gbf0553eb. [Online]. Available: https://github.com/chipsalliance/verible

[13] Microsoft Research, "Z3 Theorem Prover." [Online]. Available: https://github.com/Z3Prover/z3,Accessed: 2025-09-17.

[14] National Institute of Standards and Technology, "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions," https://doi.org/10.6028/NIST.FIPS.202, 2015, FIPS PUB 202.

[15] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, and R. Van Keer, "FIPS PUB 202 - SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions," 2015. [Online]. Available: https://keccak.team/hardware.html

[16] S. Machetti, P. D. Schiavone, T. C. Müller, M. Peón-Quirós, and D. Atienza, "X-HEEP: An Open-Source, Configurable and Extendible RISC-V Microcontroller for the Exploration of Ultra-Low-Power Edge Accelerators," 2024.

[17] Open-HW Group, CV32E40X User Manual, 2024, Accessed: 2024-06-11. [Online]. Available: https://docs.openhwgroup.org/projects/cv32e40x-user-manual/en/latest/intro.html

[18] ESL-EPFL, "cv32e40px: RISC-V Core with Extensions for Cryptography and Machine Learning," https://github.com/esl-epfl/cv32e40px, 2024, Accessed: 2024-10-11.

[19] V. Piscopo, A. Dolmeta, M. Mirigaldi, M. Martina, and G. Masera, "A Deep Dive into Integration Methodologies in RISC-V," in Proceedings of the 22nd ACM International Conference on Computing Frontiers: Workshops and Special Sessions, ser. CF '25 Companion. New York, NY, USA: Association for Computing Machinery, 2025, p. 30–33. [Online]. Available: https://doi.org/10.1145/3706594.3726969

[20] A. Dolmeta, M. Mirigaldi, M. Martina, and G. Masera, "Implementation and integration of Keccak accelerator on RISC-V for CRYSTALS-Kyber," in Proceedings of the 20th ACM International Conference on Computing Frontiers, ser. CF '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 381–382. [Online]. Available: https://doi.org/10.1145/3587