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

Reduce, Reuse, Reverify: An efficient approach to transition formal verification environments from PCIe Gen6 to Gen7

Md Zahid Fazal, Cadence Design Systems, Bengaluru, India (zahidf@cadence.com)

Moola Jeevan Chaitanya Goud, Cadence Design Systems, Bengaluru, India (jeevanm@cadence.com)

Hamish Hendry, Cadence Design Systems, Edinburgh, UK (hhendry@cadence.com)

Sakthivel Ramaiah, Cadence Design Systems, Bengaluru, India (sramaiah@cadence.com)

***Abstract*—As PCIe (Peripheral Component Interconnect Express) technology advances, maintaining backward compatibility while optimizing performance for next-generation interfaces has posed a significant challenge. This paper explores the reuse of two existing PCIe Gen6 formal verification environments for PCIe Gen7 by strategically using gearbox technique. The gearbox is a critical component in managing protocol timing to accommodate Gen7's increased bandwidth. Through systematic refinement of the formal verification setup —including protocol constraints, timing assumptions, and data path configurations—we demonstrate an efficient transition from Gen6 to Gen7 without the need for a complete environment overhaul. This approach significantly reduces verification effort and accelerates the validation process for next-generation PCIe designs. The proven results confirm that with targeted modifications, the Gen6 verification infrastructure can effectively validate Gen7 implementations, ensuring compliance with new specifications while leveraging prior investments in formal verification.**

Keywords—PCIe Gen7, Formal Verification, Property Reuse

I. INTRODUCTION

The rapid progression of PCI Express (PCIe) standards has brought significant improvements in data rate and bandwidth capabilities, with PCIe Gen7 doubling the data rate to 128 GT/s compared to its predecessor, Gen6 at 64 GT/s. While this evolution enables higher performance and lower latency, it also introduces substantial complexity in design verification, particularly when adopting formal verification methodologies. Formal environments, which are known for their exhaustive and property-driven checking, require significant setup effort and design-specific tuning. As a result, reusing an existing, well-established formal verification environment becomes a valuable objective to reduce verification time, effort, and risk.

In this work, we present a methodology to extend and reuse two example PCIe Gen6 formal verification environments for Gen7 designs by introducing a gearbox mechanism that effectively bridges the operational differences between the two generations. At the heart of this challenge lies the disparity in data rates — Gen6 operates at 64 GT/s while Gen7 pushes the boundaries to 128 GT/s. This difference not only affects the raw data throughput but also impacts timing, sampling, and synchronization of control and data signals within the formal verification setup.

Direct reuse of Gen6 test environments for Gen7 without accounting for the rate mismatch would result in inaccurate formal behavior, false failures and invalid property triggering due to misaligned sampling and inconsistent handshakes between the formal model and the DUT. To overcome this, we designed a formal-aware gearbox, a synchronization and translation layer that takes 64 GT/s data from the Gen6 environment and reformats it appropriately for Gen7 operation at 128 GT/s. This gearbox ensures correct timing alignment, preserves data integrity, and properly maps control signals, thereby preserving the semantics of Gen6 assertions in the context of Gen7 execution.

The proposed solution not only safeguards the investment made in Gen6 verification infrastructure but also provides a scalable methodology to future-proof formal environments for upcoming PCIe standards. Additionally, the gearbox approach facilitates a smoother left-shift in the verification timeline, enabling early bug detection without having to wait for full-blown Gen7 environment readiness.

We also followed a formal-first approach to deploy formal as the mainstream block level verification method to catch early design bugs. This helped avoid time-consuming testbench development for issues that could be efficiently caught by formal methods. In the sections that follow, we detail the design and integration of the gearbox, its impact on formal property accuracy, and the results of reusing Gen6 properties in Gen7 verification flows, supported by case studies and observed benefits formal convergence.

II. METHODOLOGY

In this setup, we have a fast clock domain and a slow clock domain (divide by 2 of fast). These two domains communicate through a gearbox, which ensures proper synchronization between them. This is critical in PCIe verification when transitioning from Gen6 to Gen7, as the data rate changes, but the verification environment needs to be reused efficiently.

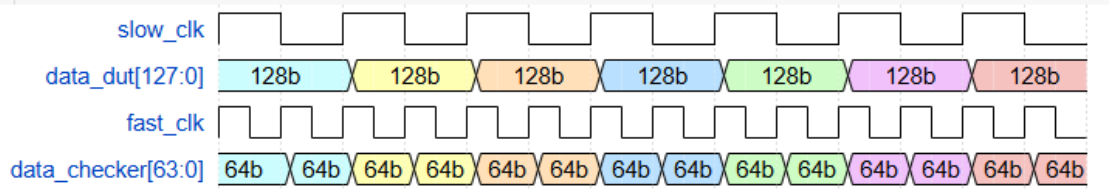


Figure 1: Data sampling (DUT vs Checker)

Gearbox Logic and Synchronization

The gearbox acts as a synchronization mechanism between the fast and slow clock domains. It ensures that:

- The incoming data in the formal checker should be continuously sampled on fast_clk and the DUT works on slow_clk.
- There is no data corruption due to the difference in clock rates.

Challenges and need for the Gearbox Technique

In transitioning from Gen6 to Gen7, we deliberately opted for the gearbox method over a simplistic parameterization approach to reuse the formal verification environment. While on the surface, changing parameters such as data widths or bus sizes might appear to be a straightforward solution, the reality within our environment is significantly more intricate.

The formal environment, particularly the deskew logic, comprises over 9,000 lines of highly interdependent code. This includes not just the core verification logic, but a large volume of auxiliary code—covering aspects such as:

- Detection logic for specific input and output data patterns
- Complex environmental FSMs
- Sequences and temporal constraints aligned to Gen6 timing assumptions
- A wide array of assertions fine-tuned for Gen6 protocol behavior

All these components are tightly coupled with Gen6 architectural assumptions, signal timing, and data width expectations. Attempting to refactor this entire environment through parameterization would introduce significant risk, require deep revalidation, and require extensive engineering effort. Each parameterized change could cascade into multiple submodules, leading to bugs that are hard to trace in a formal context.

Furthermore, during this transition, the Gen7 test chip was approaching code freeze, and time was critical. There wasn't sufficient bandwidth to onboard new engineers into the complex Gen6 formal environment and simultaneously adapt it with parameterized hooks.

In this context, the gearbox method served as a highly efficient and elegant workaround. By inserting a translation layer—a gearbox wrapper—between the Gen7 DUT and the existing Gen6 formal environment, we preserved the core structure of the verification environment and avoided invasive rewrites. The gearbox allowed us to bridge data width and protocol mismatches between the two generations without altering the core logic of the formal testbench.

This approach ensured:

- Minimal code changes to the 9k-line environment
- Quicker onboarding for engineers unfamiliar with the deep internals of Gen6 verification
- Faster verification turnaround, which was critical given project timelines
- Retention of assertion quality and maturity already validated in Gen6

Note: There was no compromise on the verification quality even during the stringent timeline and test chip release. We indeed added extra assertions for Gen7 formal environment to bridge the verification gap.

Case Study- I

PCIe Gen7 PCS – Aligner 128b Module

The Aligner 128b Module is a foundational component in the PCI Express (PCIe) Physical Coding Sublayer (PCS). It is strategically positioned as the initial receiver of raw, potentially unaligned 128-bit data streams from the Physical Media Attachment (PMA). Its primary responsibility is to re-establish proper data block boundaries and ensure protocol compliance before data moves upstream to higher layers of the link and transaction systems. Due to its frontline location in the physical layer stack, the aligner must also be resilient to channel-induced disturbances, providing robust handling of noise and signal integrity challenges.

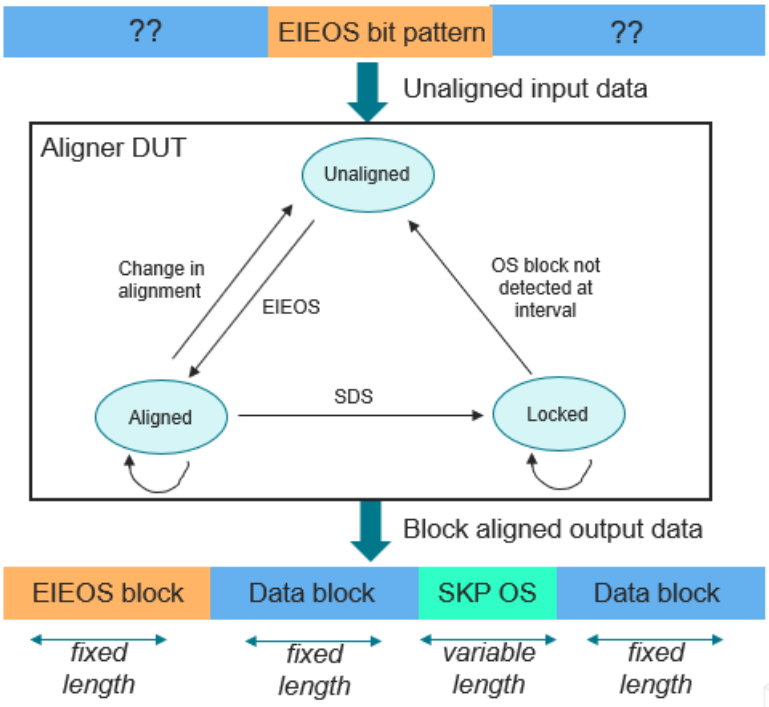


Figure 2: Aligner 128b block

Key Functionalities:

- **Data Alignment and boundary recovery:**
  - Continuously monitor the incoming bitstream for special alignment markers, such as the EIEOS ordered set, as defined by the PCIe Gen7 specification.
  - Realigns the 128-bit parallel data to ensure they are byte-synchronous and aligned with expected word boundaries.
  - Maintains a dynamic alignment state machine to adapt to changes in data flow or loss of lock events.
- **Skip Detection:** Detects SKP ordered sets inserted into the stream by the transmitter for clock compensation during link operation. It also detects its length.
- **Error Correction:** Detects anomalies, bit errors, or mismatches within ordered sets and applies correction schemes defined in PCIE spec where applicable.

Formal Env Re-use: The Gen6 aligner module has already been formally verified and includes a checker module. The aim is to reuse the existing formal environment.

Challenges and Solutions:

1. **Input Data Synchronization:** Since the Gen7 design operates on a 128-bit data width while the reused Gen6 checker is configured for a 64-bit data width, a gearbox technique was utilized alongside assumptions to achieve synchronization between the DUT and checker.
2. **Formal Checker Updates:** As in the Gen7 design data width increased from 64 to 128 bits, all the symbols of the ordered set are received in single clock cycle. Hence there is a sampling change and modifications in the design FSMs. The auxiliary logic in the existing checker had the potential to diverge from the expected behavior. To address this, the auxiliary logic was carefully tuned to ensure alignment with the updated design architecture. Additionally, a set of new assertions were introduced directly at the DUT interface. These assertions helped confirm the integrity of gearbox conversions and were instrumental in quickly identifying several initial bugs during the early stages of verification.

Implementation:

The formal environment setup was intentionally kept simple and leveraged just two gearbox components to enable reuse of the existing Gen6 checker. The input gearbox converts the 64-bit input data generated by the formal environment into 128-bit words, which are then fed into the Design Under Test (DUT). On the output side, a corresponding gearbox converts the DUT's 128-bit output back into 64-bit data. This down-converted output is then compared against the 64-bit model-generated reference output within the checker to validate correctness. Some cover properties were written on the DUT 's 128bit Interface to ensure it does not omit valid input scenarios due to the chosen method. The received data stream has very relaxed constraints to allow reception of Ordered Sets with bit errors that are both inside and outside the correctable levels described in the PCIE spec.

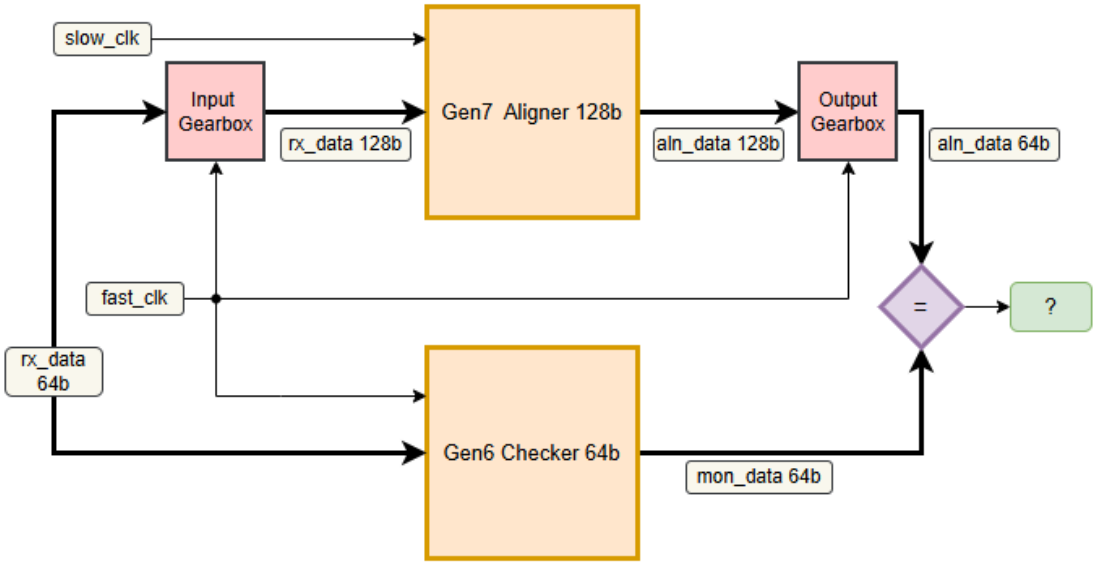


Figure 3: Formal Env for Aligner

Bug Example 1:

The design is performing an unexpected EIEOS correction. As per PCIe Gen6/Gen7 spec, EIEOS is valid if at least consecutively 5 received stream symbols matches with their respective EIEOS pattern, with either symbol0 or symbol8 being a valid EIEOS symbol. In the captured scenario, the DUT is correcting it even though neither symbol0 nor symbol1 matches their corresponding EIEOS symbol.

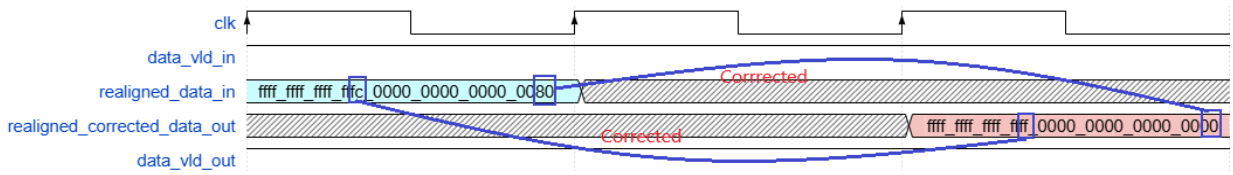


Figure 4: Bug example 1

Bug Example 2:
As per PCIe Gen6/Gen7 spec, EIOS is valid if at least 5 received stream symbols matches with their respective EIOS pattern, with either symbol0 or symbol8 being a valid EIOS symbol. In the captured scenario, the DUT fails to correct the first half of the block, even though the received stream meets the spec-defined correctable limits.

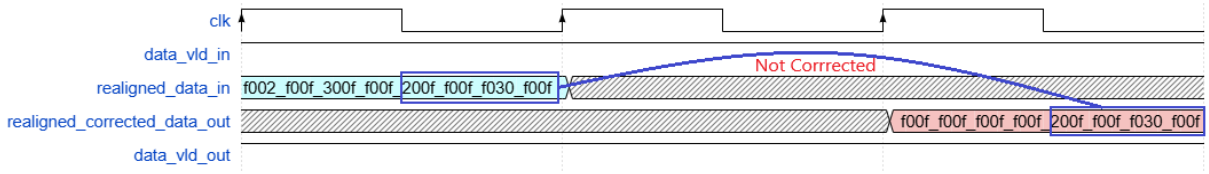


Figure 5: Bug example 2

Both example bugs relate to bit error corner case scenarios that are quick and easy to catch using formal verification but could easily be missed in simulation. Numerous further bugs were found relating to design behavior around non ideal input traffic containing bit errors.

Case Study - II

PCIe Gen7 Lane to Lane Deskew
The deskew function is a fundamental component in the Link Training and Initialization (LTSSM) phases of PCI Express (PCIe) communication. Its primary purpose is to ensure the reliable alignment of data across multiple lanes, which is critical for maintaining the integrity and performance of high-speed serial links.

Key responsibilities of the deskew block include:

- Data Stream Alignment
  - During multi-lane communication, timing discrepancies can occur due to variations in signal propagation across lanes. The deskew function compensates for these differences by realigning the data streams, ensuring that all lanes are synchronized before the data is forwarded for further processing.
- Utilization of Elastic Buffers and FIFO Queues
  - To facilitate alignment, the deskew mechanism employs elastic buffers and First-In-First-Out (FIFO) queues. These temporary storage elements allow early arriving data to be held until delayed signals catch up, thereby enabling precise synchronization across all lanes.
- Ensuring Data Integrity
  - Misalignment of data across lanes can lead to corruption and protocol violations. By maintaining proper alignment, the deskew function plays a critical role in preserving data integrity and ensuring the robustness of the PCIe link.

Reusing the pipe deskew module of PCIe Gen6 for Gen7 with the addition of a gearbox involves several technical challenges due to speed, width, and protocol changes. Below are the key challenges during this reuse:

1. Data Width and Rate Mismatch
   - Gen6 operates at 64 GT/s, while Gen7 doubles that to 128 GT/s. A gearbox was used to convert between these data rates.
2. Synchronization
   - Formal verification may report false failures due to improper data sampling if synchronization is not handled correctly.
   - All control signals going into the design were constrained to be synchronous to the slower clock used by the design.
3. SKP interval adjustment
   - Adapting the existing deskew auxiliary code to handle new SKP intervals required some environment changes.

Example assertion

The following assertion checks whether the data output on each lane is correctly aligned after the initial deskewing process has completed. It has a special set of constraints (unique to this assertion) that ensure at least 1 lane has "zero skew" compared to the other lanes. For this setup, all lanes receive the same data pattern and just have varying degrees of skew inserted by the formal environment. If the deskew block is working correctly, it will remove the skew on the delayed lanes so that the data output on each lane matches the other active lanes. The exhaustive nature of formal verification ensures that all combinations of skew between the lanes are evaluated.
This concept is illustrated with the following example showing some simple incrementing data values. SDS is the Start of Data Stream ordered set that the deskew block uses to perform deskew and align to.
Input data with skew between lanes:

- Lane0 (zero skew) input data> SDS 1 2 3 4 5 6 7 8 9 10 11 12 13 15 16 17 18..
- Lane1 input data < variable delay to model lane skew> SDS 1 2 3 4 5 6 7 8 9 10 11 12 13 15 16 17 18..

Expected output after deskew complete

- Lane0 SDS 1 2 3 4 5 6 7 8 9 10 11 12 13 15 16 17 18..
- Lane1 SDS 1 2 3 4 5 6 7 8 9 10 11 12 13 15 16 17 18..

Simple assertions generated for each lane in the PCIE link check the output data matches between lanes indicating lanes successfully deskewed. Lane 0 always exists for any configuration so every lane's data_out is compared with lane 0 data_out.

```
generate for (ln_idx=0; ln_idx< LINK_WIDTH; ln_idx=ln_idx+1)
  begin : deskew_lane_gen

    assert_data_aligned_after_deskew_done_gen7: assert property (
      data_valid_out && lane_map[ln_idx] |->
      data_out[LANE_WIDTH*ln_idx+63:LANE_WIDTH*ln_idx] == data_out[63:0]);

  end // deskew_lane_gen
endgenerate
```

Note that end to end data integrity of each lane is checked with separate formal scoreboard assertions to ensure no data loss, no data creation or unexpected manipulation.

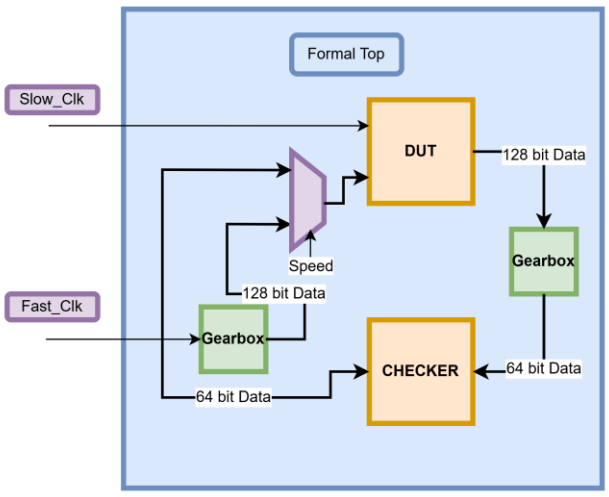


Figure 6: Formal Env for Deskew

The deskew block supports deskewing at all speeds from Gen1 to Gen7. The gearbox between the checker and DUT is only required for the Gen7 speed.

III. RESULTS

Aligner Results: For the Gen7 Aligner, formal verification was used as the primary strategy with minimal simulation effort. A total of 45 RTL bugs were identified solely through formal techniques. Most of these were uncovered using existing legacy assertions originally developed for the Gen6 Aligner. Additional issues were captured by new assertions targeting the Gen7 output interface.
In contrast, Gen6 Aligner verification began with simulation, where 18 bugs were found. Formal verification was then applied, and 25 additional bugs were detected, out of which many are corner case bugs related to noise. After formal sign-off in Gen6, no new bugs were discovered in simulation.

Deskew Results: No bugs were found in the Gen7 deskew module using formal verification although it gave high confidence in the block's quality. All the reused assertions from Gen6 passed and were used by the formal tool to generate 100% code coverage. No issues were found in the block during later top level simulation. In comparison, 19 design bugs were found in the Gen6 version of the deskew design using formal verification.

General observations: The formal verification approach proved especially effective at detecting design bugs which are related to noise and complex logic functionalities. This exhaustive approach to verifying the blocks significantly reduced the burden on downstream verification teams, ensuring increased confidence in the overall design stability.

The gearbox approach did not appear to have any negative effect on environment performance. The number of cycles to exhaustively prove each assertion was generally unchanged between Gen6 and Gen7 as the formal checks were still running at the 64b lane width even at Gen7 setup.

Table 1: Bugs count for Aligner

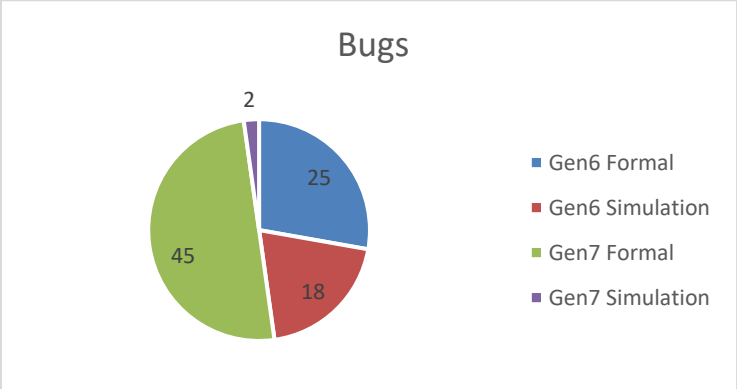
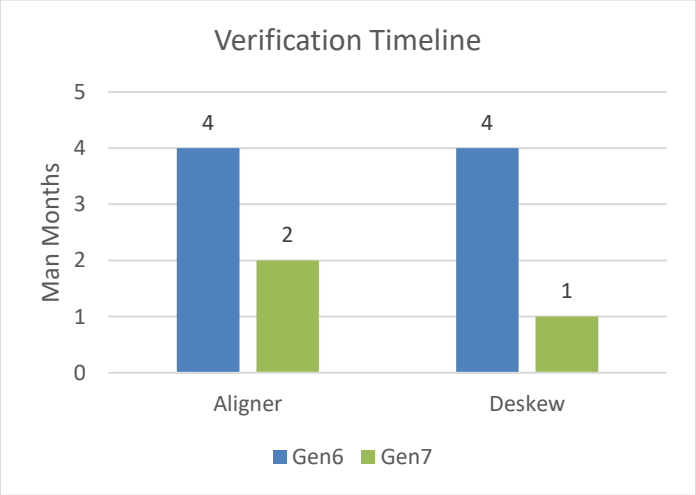


Table 2: Verification timeline for Aligner and Deskew



IV. CONCLUSION

This work effectively bridges the gap between PCIe Gen6 and Gen7 verification by leveraging the existing Gen6 formal checker to validate the Gen7 Aligner design. By reusing the formal infrastructure, we achieved a significant reduction in verification effort without compromising coverage or quality. The approach proved both scalable and efficient, uncovering a large number of RTL bugs, many of which would have been difficult to detect through simulation alone.

The transition to a formal-first methodology also streamlined verification cycles and enabled earlier closure, which in turn helped left-shift the project timeline. This reuse-centric strategy not only accelerated Gen7 verification but also promoted a sustainable, maintainable, and consistent environment for future developments.

V. REFERENCES

- https://dvcon-proceedings.org/wp-content/uploads/one-testbench-to-rule-them-all-paper.pdf