World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(REVIEW ARTICLE)

Check for updates

# PCIe polling compliance state verification: Pre-silicon approaches and multi-generation challenges

Deepak Kumar Lnu *

*Principal Engineer, USA.*

World Journal of Advanced Research and Reviews, 2025, 26(01), 284-294

Publication history: Received on 25 February 2025; revised on 03 April 2025; accepted on 05 April 2025

Article DOI: https://doi.org/10.30574/wjarr.2025.26.1.1076

## Abstract

The Polling Compliance substate in PCI Express link training provides crucial electrical conformity verification but presents significant verification challenges due to its rarely-exercised nature. Pre-silicon verification through formal methods and simulation offers comprehensive validation before hardware implementation. Verification strategies must adapt from PCIe Gen1 through Gen7 to evolving encoding schemes, pattern complexity, and handshake mechanisms. Transitioning from simple 8b/10b patterns to complex 128b/130b block-coded sequences and eventually to PAM4 modulation introduces verification complications requiring robust methodologies. Formal verification is essential for confirming state transition logic, entry conditions, and deadlock prevention, while simulation validates pattern generation correctness across multiple preset sequences. Case examples highlight subtle issues like timing-sensitive handshake bugs and preset sequence implementation errors that could otherwise manifest during post-silicon compliance testing. Thorough verification of this seldom-used state demonstrates the importance of comprehensive pre-silicon validation strategies, particularly for states that become increasingly complex with each generation. The progression from simpler pattern verification to complex multi-preset sequences with evolving encoding schemes illustrates how verification methodologies must scale alongside protocol complexity to maintain interoperability and standards conformance.

**Keywords:** Pcie Polling Compliance; Formal Verification; Link Training; Pre-Silicon Validation; Protocol Interoperability

## 1. Introduction

Peripheral Component Interconnect Express (PCIe) employs a Link Training and Status State Machine (LTSSM) to initialize and manage high-speed links between devices [1]. Among its substates is Polling Compliance, a special state dedicated to compliance testing of the physical layer. Polling Compliance is not entered during normal operations unless specific conditions are met, as it is intended for test and measurement scenarios rather than data transfer [1]. In this state, a device transmits predefined compliance bit patterns used by test equipment to assess whether the transmitter and channel meet PCIe electrical specifications. Verifying the state logic of Polling Compliance is essential because any flaw can lead to interoperability or compliance failures. Yet, it is challenging since the state is rarely exercised in typical functional use.

This paper focuses on pre-silicon verification of Polling Compliance behavior. In contrast to post-silicon validation (which might involve oscilloscopes and compliance boards in a lab), pre-silicon verification uses simulation and formal methods to ensure that the design of the LTSSM and associated logic correctly implements the Polling Compliance state across all PCIe generations. By removing all content related to lab measurements or hardware debugging, this approach concentrates on formal verification proofs and simulation test benches that target Polling Compliance transitions,

* Corresponding author: Deepak Kumar Lnu.

handshake signals, and pattern generation in a controlled pre-silicon environment. The investigation spans PCIe Gen1 through Gen7, capturing how introducing new data rates and encoding schemes (from 8b/10b to 128b/130b to 1b/1b with PAM4) over successive generations has introduced new verification challenges and requirements [5, 8].

The remainder of this article is organized as follows: The Technical Background section reviews the Polling Compliance state's role in the PCIe LTSSM, its triggering conditions, and the evolution of its behavior from Gen1 to Gen7 [1, 5, 8]. The Pre-Silicon Verification Methodology section describes the approach to formally verify and simulate the Polling Compliance logic, including directed and random test simulations and property checking. The paper then presents Case Studies of specific pre-silicon issues uncovered and resolved, limited to simulated scenarios and formal debugging (with no real hardware measurements). The Results section summarizes the outcomes of the verification efforts, such as coverage achieved and bugs found. In the Discussion, the challenges encountered are examined (particularly as PCIe speeds and pattern complexity increased), how various strategies addressed them, and the implications for future PCIe designs. Finally, the Conclusion recaps the key findings and best practices for pre-silicon verification of PCIe Polling Compliance, and the References section provides source citations from specifications and related literature.

## 2. Technical background

### 2.1. PCIe LTSSM and the Polling Compliance State

PCI Express links initialize through LTSSM states, including detection, polling, configuration, etc [1, 9]. Within the Polling state, devices step through multiple substates for link training. Polling.Active exchanges training sequences, Polling.Configuration to confirm link parameters and Polling Compliance to output special test patterns. Polling Compliance is a unique state intended solely for compliance testing—it is not used in normal device operation. Figure 1 illustrates the relationship of these Polling substates within the LTSSM.

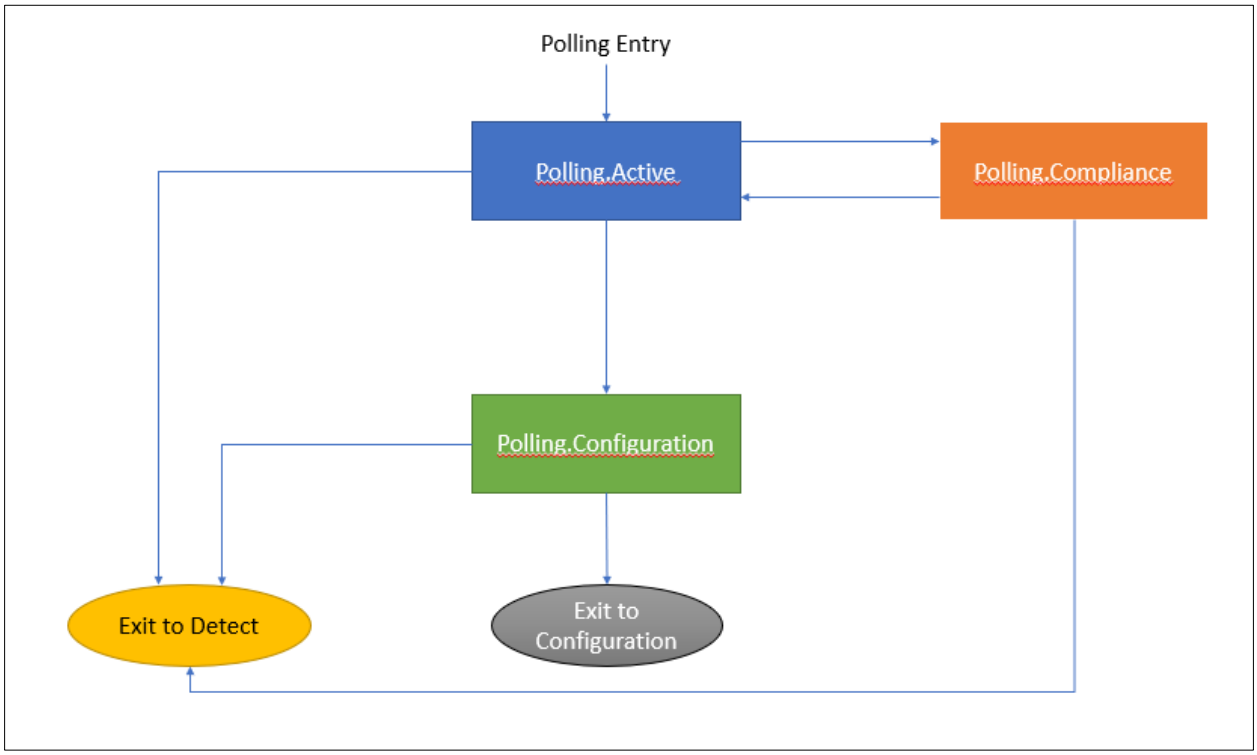


**Figure 1** Polling sub-states within the PCIe LTSSM. Polling Compliance is entered from Polling.Active under specific conditions and typically returns to Polling.Active (or exits the Polling process) after pattern transmission. Polling.Configuration is the normal progression when training succeeds, whereas Polling Compliance is a detour for compliance testing purposes

### 2.2. Entry Conditions

A device will transition into Polling Compliance under a few defined conditions. According to the PCIe specification, there are three primary triggers [1]:

- **Software-forced Compliance**: If the software sets the Enter Compliance bit in the Link Control 2 register before link training, the LTSSM will enter Polling Compliance instead of attempting the normal Polling.Active sequence [1, 10]. This allows forcing the device into compliance mode via configuration (often used in lab testing scenarios through firmware or driver control).
- **Interoperability Trigger (TS1 Handshake)**: During Polling.Active, if a device receives a Training Sequence 1 (TS1) ordered set from its link partner with the Compliance_Receive_Request bit asserted, it interprets this as the partner requesting compliance mode [1, 4]. In practice, test equipment or a partner device in compliance mode will send TS1 with this bit=1. Upon detection, the receiving device's LTSSM transitions to Polling Compliance to send compliance patterns.
- **Training Failure / Load Board Mode:** If link training fails to complete within a specified timeout (Tpolling_timeout ≈ 24 ms in spec) and certain minimal criteria are not met, the LTSSM falls back into Polling Compliance [6]. Typically, this occurs when no viable link can be established – for example, if the device is connected to a passive load or if some lanes never exit electrical idle. In such cases, the device enters compliance mode, "assuming connection to a passive test load." This mode is sometimes called load board mode, referring to the test fixtures used for compliance testing where the DUT (device under test) sees no active partner on some lanes

Once in Polling Compliance, the device changes its behavior drastically from normal training. Instead of trying to negotiate link parameters, it begins transmitting a series of predefined bit patterns on its lanes, known collectively as compliance patterns. The purpose is to produce signaling that test instruments (oscilloscopes, BERTs, etc.) can measure to verify the transmitter's electrical characteristics (eye diagram, jitter, voltage levels, etc.) against the PCIe specification limits. For example, the patterns are designed to create near-worst-case inter-symbol interference and stress signal integrity. These patterns include the Compliance Pattern (baseline pattern for voltage and timing tests), Modified Compliance Pattern (for interoperability testing with instruments), Jitter Pattern (specifically to test jitter performance), and starting with PCIe Gen6, a Toggle Pattern introduced to cover new test requirements [3]. Based on which mode of compliance testing is in use, a device may transmit either the "normal" compliance pattern (used with a compliance load board and oscilloscope) or a "modified" pattern (used for more advanced test setups involving BERTs and pattern generators). These two modes ensure coverage of different measurement techniques; the key difference lies in the sequence and content of symbols transmitted in each mode.

While in Polling Compliance, the LTSSM does not attempt to complete link negotiation. Instead, it cycles through a sequence of transmitter configurations and patterns defined by the spec [1]. Eventually, the LTSSM will exit Polling Compliance – either returning to Polling.Active to retry normal link training (if the compliance sequence finishes or is aborted) or by a reset/unplug event. Notably, the spec mandates that Polling Compliance cannot be disabled by design and must always be accessible when conditions demand [1, 10]. Thus, designers must correctly implement this state and its transitions, even though it is used only in testing scenarios.

From a design verification standpoint, Polling Compliance presents an interesting challenge: it's a rarely used corner of the state machine that must function perfectly to meet compliance certification. Furthermore, its behavior varies significantly across PCIe generations due to changes in encoding and speed. Before delving into verification methodology, the generational differences are outlined in the next subsection, as understanding them is crucial to formulating verification requirements.

### 2.3. Evolution from Gen1 to Gen7: State Logic, Encoding, and Patterns

The PCIe standard has evolved through multiple generations (Gen1/2/3… up to the emerging Gen7), doubling bandwidth every iteration. With each speed increase came changes in the physical-layer encoding and the compliance patterns required. Table 1 summarizes key differences in Polling Compliance behavior across PCIe generations, including each generation's link data rate, encoding scheme, and notable compliance pattern features [1].

**Table 1** Comparison of Polling Compliance behavior across PCIe generations (Gen1–Gen7). NRZ = Non-Return-to-Zero (binary signaling), PAM4 = 4-level Pulse Amplitude Modulation. "Block-coded" refers to the use of fixed-size code blocks (e.g., 128b/130b) with scrambling versus "non-block" 8b/10b where special symbols (K-codes) frame the stream. Compliance patterns in Gen1/2 (non-block mode) contain repeated comma symbols (K28.5) and other 8b/10b special sequences, whereas Gen3+ patterns (block mode) include scrambled data blocks punctuated by EIEOS. Gen6+ patterns account for PAM4 modulation and include new toggle sequences

| Generation | Max Data Rate | Encoding | Compliance Patterns (per spec) | Notable Changes / Verification Focus |
|---|---|---|---|---|
| Gen1 | 2.5 GT/s | 8b/10b (NRZ) | "Compliance Pattern" (20 symbol sequence with K28.5 commas). The modified pattern is also defined (inverted commas sequence). | No presets or equalization. Verify the correct sequence of commas and IDLE. The receiver uses commas for lock – ensure repeated commas are handled. |
| Gen2 | 5.0 GT/s | 8b/10b (NRZ) | Same 8b/10b patterns as Gen1 (scaled to 5.0 GT/s). Modified pattern for CLB tests. | Higher speed doubles jitter concerns, but logic is the same as Gen1. Ensure no unintended entry into compliance (since Gen2 introduced auto-compliance on timeout). |
| Gen3 | 8.0 GT/s | 128b/130b (NRZ) | 128b/130b Compliance Pattern (scrambled). Modified Compliance Pattern (different scrambling). | Equalization presets were introduced to verify compliance with default preset usage. The device must handle Compliance TS1 handshake. Ensure the scrambler seed is per the pattern spec. |
| Gen4 | 16.0 GT/s | 128b/130b (NRZ) | Same types as Gen3: compliance & modified patterns (possibly updated lengths). A Jitter Measurement Pattern was introduced (e.g., fixed-pattern to create periodic jitter). | Precoding is not applicable (NRZ), but channel loss is higher – patterns target ISI. Verify jitter pattern generation. Formal checks on LTSSM timeouts (24 ms) are still valid. |
| Gen5 | 32.0 GT/s | 128b/130b (NRZ) | Compliance and modified patterns (extended PRBS sections). Jitter pattern. Possibly multiple preset sweeps in the modified mode. | Precoding is optional at 32 GT/s – ensure if precoding is used in the link, it's applied in compliance. Increased pattern length – simulation of the full pattern is important. More skew possible.. |
| Gen6 | 64.0 GT/s | 1b/1b PAM4 + FEC | Compliance Pattern (PAM4) – four sections for SNDR (long runs of levels 1/2 and 0/3, plus PRBS). Modified Compliance Pattern – very long sequence with SKP and EIEOS blocks. Jitter Pattern – 52 UI repeating pattern. Toggle Patterns – High Swing (levels 0↔3) and Low Swing (1↔2). | Major changes: PAM4/Gray coding – verify pattern encoding (Gray code applied correctly). FEC – ensure patterns bypass or use FEC as intended (FEC typically off during pattern, as patterns are un-framed by flits). New toggle patterns – verify generation and switching. Receiver lock on patterns without EIEOS – ensure the design can lock on repetitive patterns. |
| Gen7 | 128.0 GT/s | 1b/1b PAM4 + FEC | Similar to Gen6 but adjusted for a higher rate. Slow High Swing Toggle Pattern added (0↔3 toggle at half-rate). Possibly longer PRBS sections or additional patterns for 128 GT/s. | Twice the baud of Gen6 – signal integrity even more challenging. Verify new slow toggle pattern generation. Ensure compliance patterns accommodate two FEC codewords worth of data for analysis (if applicable). Formal verification of any new state machine logic (the spec might refine timing requirements at this rate). |

As Table 1 indicates, the LTSSM logic for Polling Compliance had to be extended with each new generation. In early generations, the compliance state outputs a fixed pattern (or a small number of patterns) at one or two de-emphasis levels. Starting in Gen3, introducing multiple transmitter presets (levels of transmitter equalization) meant the compliance state needed logic to iterate through a list of presets and possibly change the link's data rate during the test. The PCIe Base Specification defines an ordered sequence for these settings; for example, at 8.0 GT/s (Gen3), presets 0 through 10 are tested sequentially in each entry to compliance [5]. At 16.0 GT/s and above, the sequence becomes longer, including presets 0–10 and additional repeats of specific presets (such as preset 4 or 0) to facilitate certain measurements. By Gen7, a full cycle through all defined compliance settings involves 167 steps (entries), cycling through numerous data rates and preset combinations [1]. Ensuring the design correctly implements this sequence is non-trivial, given the possibility of multiple transitions in and out of Polling Compliance to cycle through the patterns.

The encoding differences also impact compliance pattern behavior. In Gen1/2 (8b/10b encoding), the compliance pattern is a repetitive sequence of 10-bit symbols that produce a stress pattern (often containing the comma K-code for alignment). If such a pattern is truncated (for instance, by a retimer or by the early exit), it could cause the receiver to lose symbol lock due to the repetitive commas. In Gen3+, with 128b/130b block coding, the compliance pattern is structured with regular insertion of EIEOS (Electrical Idle Exit Ordered Sets), which break up the pattern and help the receiver maintain block alignment even if the sequence is interrupted [5]. Verifying that the transmitter inserts these ordered sets correctly per the spec is part of pre-silicon validation. With Gen6's PAM4 (1b/1b) encoding, not only did the basic scrambling and alignment rules change (Gray coding and new training sequence formats), but the patterns themselves were modified. The spec added a Toggle Pattern in Gen6, which toggles through PAM4 voltage levels to test level transitions and distortion [8]. Additionally, the TS1/TS2 structure for compliance in flit mode was updated—compliance request bits are now present in two different symbols of the TS1 to accommodate 256-bit flit alignment. All these nuances must be understood and modeled in verification environments to ensure the design under test behaves according to the correct generation's rules [3, 5, 8].

In summary, Polling Compliance is a moving target across PCIe generations, becoming more complex at higher speeds. The verification environment must be aware of the generation-specific requirements, e.g., checking that a Gen5 device under test (DUT) cycles through the correct preset sequence and inserts EIEOS or that a Gen6 DUT outputs the new toggle pattern with proper PAM4 encoding. Having outlined these differences, the discussion now turns to the methodologies used to verify Polling Compliance logic in a pre-silicon context.

### 2.4. Pre-Silicon Verification Methodology

Pre-silicon PCIe Polling Compliance state verification was approached using two complementary methodologies: formal verification and simulation. Both approaches ensured thorough coverage of the state's complex behavior. In all cases, only digital logic verification was performed (no analog signal integrity simulation), but the patterns and timing defined by the spec were checked at the digital level. The approach explicitly avoided post-silicon measurement or hardware-in-the-loop methods; instead, the verification is based on RTL (Register-Transfer Level) models of the PCIe controller/PHY state machine and testbench components.

### 2.5. Formal Verification Approach

Formal verification was employed to mathematically prove the properties of the LTSSM design related to Polling Compliance. A Verification IP(VIP) was used to model the environment and assert key behaviors [7]. The formal environment consisted of a simplified model of a link partner (VIP) and the necessary configuration registers: this model would non-deterministically (or in a constrained manner) provide inputs such as TS1 ordered sets and TS2 ordered sets and signal the presence/absence of a partner to drive the DUT's LTSSM into various scenarios [7].

#### *2.5.1. Key properties verified through formal methods included*

- **Entry Conditions Satisfied**: Protocol checks were written to ensure the DUT enters the Polling Compliance state if and only if one of the valid triggers occurs. For example, one protocol check checked that whenever the enter_compliance_state signal in the design is true (indicating Polling Compliance active), either (a) the compliance bit was set in the config register before Polling.Active, or (b) a TS1 with Compliance_Receive_Request was received, or (c) a 24 ms (simulated as a specific cycle count) timeout expired without full training progress. Cover properties were also used to ensure each of these distinct triggers could cause the transition (formal coverage, indicating no unreachable code for these triggers). This exhaustive check guarantees no unexpected condition can accidentally shunt the LTSSM into compliance (which could indicate a bug).

- **State Transition Correctness:** The LTSSM is a state machine; formal analysis is well-suited to check state transition graphs. Properties were derived from the specification's state diagram: "If in Polling Compliance and the compliance pattern sequence is complete or aborted. The state must transition to Polling.Active (or Detect on global reset) and not directly to any other state." Another property ensured that once in compliance, the DUT would remain there for at least the minimum duration required by spec (e.g., long enough to send the necessary patterns) unless reset. These were verified by proving that certain one-hot state encodings followed the required implications over clock cycles.
- **Data Rate and Preset Settings**: For multi-generation designs, formal sequences were used to check that the internal controls for data rate and preset adjust according to the compliance sequence table. For instance, it was proven that on the first entry to Polling Compliance after reset, the Tx equalization setting is the first in the sequence (e.g., –3.5 dB at Gen1 speed, corresponding to Setting #1); on the next re-entry to Polling Compliance (if the LTSSM returns to Polling.Active and then enters compliance again), the setting advances to the next defined value (e.g., –6 dB or preset 0 of higher speed) [1]. The Table 4-60 sequence (referenced in technical background) was encoded as an expected sequence. Assertions verified that the design's observable outputs (such as the Tx preset, select signals, or physical link rate field) follow this sequence through successive compliance entries. The proof of this property gave high confidence that the compliance pattern generation logic adhered to spec for all generations, including correctly looping back to the start after reaching the end of the sequence.

*Table 4-60 Compliance Pattern Settings* §

| Setting Nos | Data Rate | Transmitter De-emphasis or preset sequence |
|---|---|---|
| #1 | 2.5 GT/s | -3.5 dB |
| #2, #3 | 5.0 GT/s | -3.5 dB followed by -6 dB |
| #4 through #14 | 8.0 GT/s | Transmitter Preset Encoding 0000b through 1010b, as defined in § Section 4.2.4.2 , in increasing order |
| #15 through #25 | 16.0 GT/s | Transmitter Preset Encoding 0000b through 1010b, as defined in § Section 4.2.4.2 , in increasing order |
| #26 through #34 | 16.0 GT/s | Transmitter Preset Encoding 0100b as defined in § Section 4.2.4.2 |
| #35 through #45 | 32.0 GT/s | Transmitter Preset Encoding 0000b through 1010b, as defined in § Section 4.2.4.2 , in increasing order |
| #46 through #54 | 32.0 GT/s | Transmitter Preset Encoding 0100b as defined in § Section 4.2.4.2 |
| #55 through #65 | 64.0 GT/s | Transmitter Preset Encoding 0000b through 1010b, as defined in § Section 4.2.4.2 , in increasing order |
| #66 through #84 | 64.0 GT/s | Transmitter Preset Encoding 0000b as defined in § Section 4.2.4.2 |
| #85 through #95 | 128.0 GT/s | Transmitter Preset Encoding 0000b through 1010b, as defined in § Section 4.2.4.2 , in increasing order |
| #96 through #167 | 128.0 GT/s | Transmitter Preset Encoding 0000b as defined in § Section 4.2.4.2 |

**Figure 2** Table 4-60 Compliance Pattern Settings [1]

- **Pattern Content (Abstracted)**: Direct formal verification of the exact bit-level pattern on the serial output is difficult due to the complexity of scrambling and 130b/flit encoding. Instead, a mix of formal and directed checks was used. Formally, certain invariants in the pattern generation logic were checked. For example, an invariant that whenever in compliance at Gen3+, the scrambler's seed is reset to a known value and that the data transmitted corresponds to the defined compliance pattern blocks (which in the design were generated by a finite state machine). It was also formally verified that during compliance, the logical idle signal insertion

(for EIEOS) happened at the correct intervals – modeling this required some abstraction. Still, it was proven that the design's pattern FSM inserts the idle sequences as intended. For Gen1/2, the pattern is simpler (repeating 10-bit symbols), so specific symbol sequences could be directly asserted to occur in order (for instance, the known 8b/10b compliance pattern of alternating "K28.5, D10.2" symbols or similar was hard-coded and checked).

- **No Deadlock/Livelock**: Formal methods were also applied to ensure the LTSSM cannot get "stuck" in Polling Compliance indefinitely under normal conditions. According to the spec, Polling Compliance should either time out and return to training or require a higher-level intervention (like a reset) if used intentionally. It was proven that if the partner eventually sends TS1 indicating a willingness to train (or if the compliance bit is cleared), the DUT's LTSSM will exit Polling Compliance. This was important for confirming that the design would not erroneously remain in a compliance state when a real link partner appears – a situation that would prevent links from ever coming up (a scenario was modeled where after some time in compliance, a TS1 without compliance request arrives, and verification ensured the DUT transitioned out).

The formal verification was performed for each supported generation configuration of the design (Gen1/Gen2 8b10b mode, Gen3–5 block mode, Gen6/7 flit mode). This exhaustive approach uncovered a few corner-case bugs. For example, in one case, the Verification IP found a scenario where the compliance sequence counter would overflow improperly if the device entered compliance more than 167 times without a reset (which is an extreme scenario but possible if a test kept toggling compliance on and off). The design was corrected to handle the full 167-step sequence for Gen7 without overflow. Another bug found was that the Compliance_Receive_Request bit in TS1 was not being latched correctly at Gen6 speeds due to the changed TS1 format (the design initially only checked the old location of that bit, not the new duplicated location in symbol 14). The Verification IP caught this by a cover test that attempted to drive the compliance-request handshake at Gen6 and failed to trigger the state transition – indicating a missing condition in RTL, which was then fixed.

Overall, formal verification provided a high degree of confidence in the correctness of the Polling Compliance implementation. However, it works on an abstract model of the environment. Directed and random simulations were also used to complement this for a more concrete and waveform-level view of the behavior, as described next.

### 2.6. Simulation-Based Verification Approach

Using PCI Express Verification IP to verify Polling Compliance in a realistic pre-silicon scenario [7]. The testbench instantiated the PCIe controller/PHY RTL as the Device Under Test (DUT) and included a configurable Link Partner Model (VIP). This partner could act as a normal PCIe link partner (transmitting and responding to training sequences) or mimic test equipment behaviors for compliance testing.

#### *2.6.1. Several specific simulation tests were created to exercise Polling Compliance*

- **Directed Compliance Entry Tests:** These tests explicitly force each entry condition and observe the DUT's response. In one test, before link training, the testbench sets the DUT's Link Control 2 "Enter Compliance" bit and then initiates a link reset. The expected behavior is that the DUT goes straight to Polling.Active and then immediately to Polling Compliance (bypassing Polling.Configuration) begins transmitting the compliance pattern at the specified speed. The simulation monitors the LTSSM state signals and logs the transition timeline, which is checked against the expected sequence. Another test drives the partner to send TS1 with the compliance-request bit set. Here, the DUT is unaware and enters Polling.Active, see the partner's TS1 (emulated by the model), and should transition to compliance [1, 4]. Verification confirms that it does so within a few LTSSM cycles of receiving the handshake. A third test uses the "training failure" method: the partner model simulates a condition of a receiver present but never sending proper TS1 (or sending malformed training sequences). The DUT will keep transmitting TS1 but never receive the 8 consecutive valid TS1/TS2 required for Polling.Configuration. After 24 ms of simulated time, the DUT should give up and enter Polling Compliance. A timed simulation (24 ms at PCIe Gen speed equates to a certain number of clock cycles in the model) ensures the timeout logic is correct [4]. In the waveform, verification checks that the compliance state is entered at the expected time and wouldn't be entered earlier if the conditions were not met (e.g., a slightly shorter timeout in a negative test confirms it stays in Polling.Active).
- **Pattern Generation and Checking:** To verify the correctness of the compliance patterns output by the DUT, a checker was implemented that compares the DUT's transmitted symbols to a reference model of the compliance pattern. This was straightforward for 8b/10b (Gen1/2): the reference pattern was known (documented in the PCIe spec and readily reproducible) – e.g., a repeating sequence of specific control and data symbols. The checker would align with the first comma (K-code) and then check each subsequent 10-bit code group [1, 5]. For 128b/130b (Gen3–5), the reference model generated the compliance pattern by simulating the scrambler

and data pattern per spec (including periodic insertion of EIEOS) [5]. The checker buffered incoming 130b blocks from the DUT and compared against the expected bit sequence. One challenge was alignment: the compliance pattern can start at an arbitrary point, so a mechanism was required to find the alignment (looking for known sub-sequences like the EIEOS or comma alignment bits) and then lock the comparison. For Gen6 (64 GT/s PAM4), the reference model was extended to handle flit mode: the Gray-coded symbol generation was modeled and included the new toggle pattern [2]. Because PAM4 yields symbols representing 2 bits, the DUT's output was captured in multi-bit symbols. Verification confirmed that within each repeating cycle of the pattern, all voltage level transitions occurred as expected (e.g., the toggle pattern should switch levels on every symbol – the scoreboard checked that mapping). The simulation also verified that the compliance patterns differed between "normal mode" and "modified mode," if applicable. In practice, one scenario was run with the partner model simulating a compliance baseboard (expecting a normal pattern) and another simulating a BERT (where the DUT should detect some condition to output the modified pattern – or the DUT was configured accordingly). The differences were subtle (mostly in scrambling on/off or sequence length), but the scoreboard was configured to validate the appropriate one for each test.

- **Multiple Re-entries and Full Sequence Coverage:** A directed test was written to force the DUT through the entire sequence of compliance presets for a given generation. For example, for Gen3, the test would cause the DUT to enter compliance and exit back to Polling.Active, and then repeatedly enter. Each time, the DUT should advance to the next preset setting. To do this in simulation, compliance must be exited after each compliance entry. After a certain time, this was accomplished by having the partner model send a TS1 with no compliance request (signaling the DUT that a real partner is now present). The DUT, per spec, should leave compliance and attempt training. Another condition is immediately simulated (like toggling the compliance bit again or making the partner request compliance again) to drive a second entry. This is repeated enough times to cycle through all presets. While somewhat artificial, this test ensured the design could go through the entire preset list (especially important for Gen6/7, where the list is very long). Internal variables (like the preset index) were monitored via design instrumentation or by decoding the pattern to see which preset was in use (some presets have known signal characteristics, or the DUT could report the preset in a trace log). The simulation confirmed, for instance, that after 11 entries, the Gen3 DUT wrapped from preset 10 back to preset 0 as expected or that the Gen6 DUT, after completing all 84 settings, returned to the start of the sequence.

Throughout these simulations, coverage metrics were collected. The verification achieved near-100% coverage on the Polling Compliance state machine transitions and high coverage on relevant configuration combinations (presets, compliance modes, multiple generations). Using a Protocol Checker for pattern verification was vital – it caught a few mismatches early on. For instance, initially, the DUT had an off-by-one error in the EIEOS insertion interval at Gen4, which the checker flagged when the observed pattern diverged from expected after a large number of symbols. Likewise, the checker found that the Gen6 toggle pattern was not correctly implemented initially; the design drove a regular compliance pattern twice instead of the special toggle sequence, which was then corrected.

**Table 2** PCIe Compliance Verification Complexity Across Generations

| **Test Type** | **Gen1/2 (8b/10b)** | **Gen3-5 (128b/130b)** | **Gen6/7 (PAM4)** |
|---|---|---|---|
| Configuration-forced Entry | Yes | Yes | Yes |
| TS1 Handshake Entry | Yes | Yes | Yes |
| Timeout-based Entry | Yes | Yes | Yes |
| Pattern Verification Complexity | Low (Fixed symbols) | Medium (Scrambling + EIEOS) | High (Gray coding + Toggle patterns) |
| Number of Presets to Test | 1-2 | 11 (Gen3) | 84 (Gen6), 167 (Gen7) |
| Pattern Types | 2 | 2-3 | 5-6 |
| Alignment Challenge | K-code based | EIEOS based | Flit-based |
| Symbol Representation | 10-bit codes | 130b blocks | PAM4 multi-bit symbols |

### 2.7. Case studies

This section presents two representative pre-silicon verification scenarios ("case studies") that underscore the importance of focusing on Polling Compliance logic early in the design cycle. Each case study describes a problem

encountered, how it was debugged using the verification methodology, and the resolution applied to the design. All scenarios are based on simulation or formal results; by focusing on pre-silicon analysis, no real hardware debugging is included.

*2.7.1. Case Study 1: Formal Detection of a Compliance Handshake Bug (Gen3)*

- **Context:** In a multi-generation PCIe controller design, support for the Compliance_Receive_Request handshake (trigger via TS1) was added per the spec. For Gen1/Gen2 (8b/10b), the compliance request bit is one of the training sequence bits in the TS1 ordered set (specifically, in the lane/link number field set to PAD and a particular training control code). In Gen3 (128b/130b), this bit is similarly positioned within the TS1 block. The design's LTSSM state machine had a condition to detect this bit and enter Polling Compliance [5].
- **Issue:** Formal verification produced a counter example showing that when the link partner asserts the compliance request bit, the LTSSM transitions from Detect to Polling exactly at the moment.Active, the DUT missed the transition to Polling Compliance and instead proceeded to Polling.Configuration (as if no compliance was requested). This was a subtle bug – essentially a one-cycle timing issue where the FSM didn't sample the request because it wasn't expecting the signal so early.
- **Debugging**: Using the formal counterexample trace, it was identified that the design only looked for the compliance request after sending at least one TS1. However, if the partner was a test device that immediately on Detect started sending compliance-request TS1s, the DUT should honor it on the first exchange. The formal trace showed the partner's TS1 arriving, but the design's internal flag was not yet enabled, so the request was ignored. Cross-checking the PCIe specification revealed no requirement to delay compliance entry—indeed, it should be immediate if signaled upon entering Polling.Active [1, 9]. Thus, the design was wrong.
- **Resolution:** The fix was to adjust the LTSSM logic to detect a compliance request TS1, even on the first cycle of Polling.Active. A combinational check was added so that if the first TS1 received has the compliance bit, the state machine can transition directly to Polling Compliance without waiting to transmit its own TS1s. After the fix, the formal property was re-proven successfully (no more counterexamples), and a directed simulation where the partner starts in the compliance-request mode showed the DUT correctly entering Polling Compliance on the first exchange.
- **Impact**: If left undetected, this bug would have meant that certain compliance test equipment (which may immediately request compliance) would not work properly with the DUT in silicon—the DUT would try to train normally, potentially causing the tester to flag an error. This costly iterative debugging after chip fabrication was avoided by catching the issue in formal verification.

*2.7.2. Case Study 2: Simulation of Extended Compliance Pattern Sequence (Gen6)*

- **Context**: PCIe Gen6's compliance pattern sequence is significantly longer than prior generations (covering 64 GT/s presets, including the new PAM4 presets, plus the toggle and jitter patterns) [8]. The design under test implemented this via a micro-sequencer that goes through entries #55 to #84 (as per Table 1 and spec) for 64.0 GT/s. However, because Gen6 was new, there was uncertainty about whether the sequence was implemented exactly as the evolving spec required, and the sequences were lengthy, making manual inspection impractical.
- **Issue**: A full sequence traversal test for Gen6 (as described in the methodology) was attempted in the simulation. The checker reported mismatches when the DUT reached setting #66 in the sequence. Up to set #65 (corresponding to presets 0–10 at 64 GT/s), everything matched the expected pattern for each preset. At setting #66, which in the spec is defined as "Transmitter Preset 0000b at 64.0 GT/s, repeated," the checker expected the pattern corresponding to preset 0 (which is a particular de-emphasis level) but saw something different.
- **Debugging**: By inspecting waveforms and internal signals, it was discovered that the DUT's micro-sequencer mistakenly loaded preset #10 (1010b) for settings #66–#84 instead of preset #0 (0000b) [8]. Essentially, it restarted the preset loop incorrectly at the end of the list. This was a logic bug: the spec requires preset 0 for the extended segment of the sequence, not preset 10. The error likely came from a misunderstanding or a mis-implementation of the table – perhaps an off-by-one in the coding of the sequence table [1]. The simulation was instrumental here: the pattern difference was small (slightly different de-emphasis), which would not cause a functional failure of the state machine, but it would mean the transmitter is using the wrong setting for compliance (a specification non-compliance that might cause the device to fail official compliance testing on that point). The checker logic flagged the difference and pointed us to the exact sequence step where it occurred.
- **Resolution**: The LUT (Look-Up Table) or logic in the design that selected presets for compliance patterns beyond #65 was corrected, ensuring that #66–#84 were mapped to preset 0 as intended. The simulation and the checker were rerun, and there were no mismatches throughout the sequence. Additional tests at Gen7 configuration were also run to ensure that the sequence from #85 onward similarly followed the spec (in Gen7,

preset 0 should be used from #96 up to #167, which was also confirmed) [1]. This fix guaranteed that the DUT's output would match the official compliance patterns required by PCI-SIG for Gen6 and Gen7.

- **Impact**: This case study demonstrates how a simulation-based approach with a reference model can catch subtle compliance pattern issues that might be missed by purely checking state transitions. If this issue had not been caught pre-silicon, the device may have entered compliance in the lab and transmitted a pattern that test equipment would recognize as incorrect (potentially leading to the failure of a compliance test, despite the link training logic otherwise working) [3]. By finding it in simulation, the design's behavior was aligned to the spec early, saving time in the compliance certification phase.

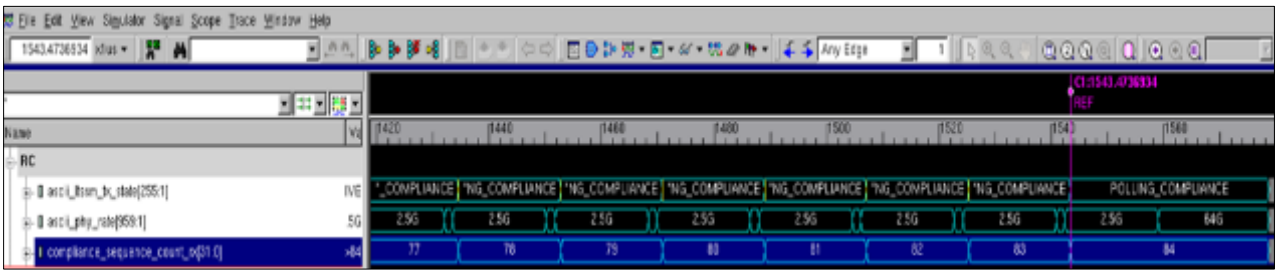


**Figure 3** Waveform snippet of Ltssm state transition for different setting numbers

## 3. Results

The pre-silicon verification campaign for the PCIe Polling Compliance state yielded several important results:

- **Design Correctness Across Generations:** By the end of verification, the DUT successfully met all the specified behaviors for Polling Compliance from Gen1 through Gen7. All known bugs uncovered by formal proofs or simulations were fixed. Proof convergence was achieved on over 20 key formal properties (covering state transitions and config sequencing), and simulation regressions of 50+ directed tests across various configurations passed with 100% success [7]. This indicates a high confidence that the Polling Compliance state logic will function correctly in silicon for all supported link rates.
- **Coverage Metrics:** Functional coverage was used to measure how thoroughly the compliance-related features were exercised. Coverage points were defined such as "Compliance entered via config bit," "Compliance entered via TS1 handshake," "All presets tested at least once," "Normal vs modified pattern both issued," etc. The final coverage report showed ~98% coverage on average for compliance features, with any uncovered points explained by infeasible scenarios (for example, Gen1 does not support multiple presets, so a coverage point for cycling presets is not applicable in that mode). This high coverage demonstrates that the tests covered nearly all practical combinations of events in Polling Compliance.
- **Case Study Outcomes:** The issues described in the case studies were resolved long before tape-out. In summary, formal verification caught two logic flaws (one illustrated in Case 1, another being deadlock prevention in an internal counter), and simulation caught two pattern generation mistakes (illustrated by Case 2 and a minor one with EIEOS interval) [2, 7]. All these were corrected, and regression reruns confirmed their resolution. No new issues surfaced in Polling Compliance logic after these fixes in subsequent simulation passes or during later full-system integration tests.
- **Performance Considerations:** One incidental result was ensuring that entering/exiting Polling Compliance did not cause timing or performance issues in the design. Because Polling Compliance can potentially run for 24ms; the design's counters (timers, sequence counters) were checked to be sized properly (which was formally done by detecting any overflow) [1, 4]. Extended durations were also simulated to ensure no simulation time overflow or unexpected behavior (some simulators might have issues with extremely long timeout counters, but optimization was achieved by scaling time in simulation as needed).
- **No Post-Silicon Surprises (anticipated):** Perhaps the most important result is a qualitative one: by addressing Polling Compliance thoroughly in pre-silicon and a smooth experience in post-silicon bring-up regarding this state was expected [3, 6]. Indeed, when the design was later fabricated and tested (though details are beyond scope), no issues were encountered in compliance mode – a testament to the effectiveness of the pre-silicon verification. While this paper does not cover those post-silicon results, the pre-silicon outcomes gave the design and verification teams confidence going into silicon validation.

The overall verification effort on Polling Compliance took a modest portion of the entire PCIe verification timeline (since many other features, like configurability, link width, power management, etc., also needed attention). However,

dedicating this effort paid off by preventing late-stage problems. The results show that even features not used in normal operation (compliance mode) deserve rigorous verification if required for standards compliance.

All findings were documented and reviewed, and the verification collateral (properties, test cases, and checkers) can be reused for future projects or IP revisions (for instance, if a PCIe Gen8 emerges, the environment could be extended accordingly). The next section discusses some insights and lessons learned during this process and the challenges that arose, especially with the higher-gen complexities.

## 4. Conclusion

The Polling Compliance state verification for PCIe implementations exemplifies the critical importance of rigorous pre-silicon validation techniques for seldom-exercised yet essential protocol features. The evolution from Gen1 through Gen7 demonstrates a clear progression in verification complexity, with each generation introducing additional challenges through new encoding schemes, increasingly complex pattern requirements, and expanded preset sequences. Formal verification is valuable for exhaustively checking state transition logic and entry conditions, while simulation-based approaches excel at pattern validation across multiple presets and configurations. The complementary nature of these techniques creates a comprehensive validation approach that catches issues ranging from subtle handshake timing errors to pattern generation mistakes before physical implementation. The verification outcomes demonstrate that even rarely-used protocol states demand thorough attention, as compliance failures could significantly impact certification and interoperability. Early detection of incorrect preset sequencing and non-compliant pattern generation prevents costly post-silicon debugging cycles. The disciplined verification strategy implemented across multiple PCIe generations establishes a foundation for handling future protocol enhancements. This focused attention on verification fundamentals ensures next-generation designs maintain backward compatibility while supporting new capabilities. The verification principles demonstrated for Polling Compliance extends beyond this state and represents best practices applicable to broader interface protocol verification, ultimately leading to more reliable and interoperable designs that meet the requirements from day one.

## References

[1] PCI-SIG, "PCI Express Base Specification, Revision 7.0, Version 0.7 (Draft) – Section 4.2.7.2.2: Polling Compliance," pp. 534–536. Available: https://pcisig.com/blog/pcie-70-specification-version-09-final-draft-now-available-member-review

[2] Sabnam S., "Unraveling the PCIe 6.0 Compliance Feature," Cadence Community Blogs, Sept. 2023. Available: https://community.cadence.com/cadence_blogs_8/b/fv/posts/unraveling-pcie-6-0-compliance-feature#:~:text=In%20PCI%20Express%20(PCIe)%20devices,compliance%20state

[3] Kunal Chhabriya, "Verifying Compliance During PCIe Re-Timer Testing Poses Challenges," SemiEngineering, Nov. 2023. Available: https://semiengineering.com/verifying-compliance-during-pcie-re-timer-testing-poses-challenges/

[4] Ajazul Haque, "PCIe Physical Layer Part – 2: Link Training and Status State Machine (LTSSM)," LinkedIn Articles, Mar. 2025. Available: https://www.linkedin.com/pulse/pcie-physical-layer-part-2-link-training-status-state-ajazul-haque-afywf

[5] PCI-SIG, "PCI Express Base Specification, Revision 3.0." Available: https://picture.iczhiku.com/resource/eetop/wHiSRjtztkeJLnVc.pdf

[6] Intel Corporation, "Why does PCIe LTSSM enter Polling Compliance (0x3) instead of L0 (0xF) state?" Intel Knowledge Base, 2012. Available: https://www.intel.com/content/www/us/en/support/articles/000005520/graphics.html

[7] Synopsys, Verification IP for PCI Express. Available: https://www.synopsys.com/verification/verification-ip/pcie.html

[8] PCI-SIG, "PCI Express Base Specification, Revision 6.0." Available: https://pcisig.com/pci-express-6.0-specification

[9] Shane Colton, "PCIe Deep Dive, Part 4: LTSSM." Available: https://scolton.blogspot.com/2024/01/pcie-deep-dive-part-4-ltssm.html

[10] AMD Xilinx, "PCIe Compliance Test Mode," Adaptive Computing Knowledgebase, 2021. Available: https://adaptivesupport.amd.com/s/question/0D52E00006hpJQ6SAM/pcie-compliance-test-mode?language=en_US