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

Zahid Fazal, Design Engineer-II

Moola Jeevan Chaitanya Goud, Design Engineer-I

Hamish Hendry, Design Engineering Group Director

Sakthivel Ramaiah, Design Engineering Architect

cādence®

accellera
SYSTEMS INITIATIVE

Agenda

Introduction

Background

- **PCI Express (PCIe)** is a high-speed serial interface standard used in modern computing systems.
- Each generation of PCIe doubles the data rate: **Gen6:** 64 GT/s & **Gen7:** 128 GT/s

Verification Strategy

- Formal Verification applied to numerous blocks within previous Gen5 and Gen6 controllers
- Formal proven to be highly effective at both bug detection and coverage closure
- However, formal environments are costly to build and tune for each new generation

Opportunity

- Efficient reuse of PCIe Gen6 formal environments for Gen7 critical to:
  - Reduce environment development time
  - Lower verification cost
  - Accelerate bug detection

Problem Statement

Core Problems

- **Gen6 formal environments** are not directly compatible with **Gen7 designs** due to data rate mismatch (64 GT/s vs. 128 GT/s).
  - Datapath widths are different between generations
  - Complex assertions, constraints and auxiliary code structured to assume a specific number of bytes per clock cycle. Not a simple change in parameter value to migrate most environments.

Consequences of Direct Reuse

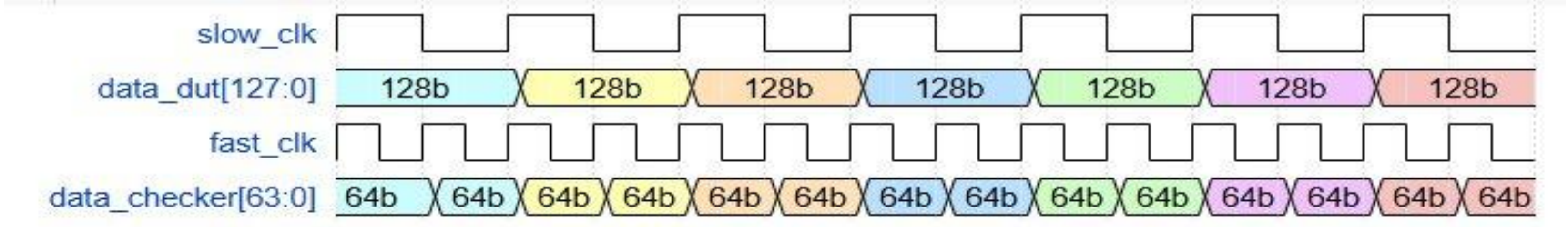
- False failures in formal checks and invalid property triggering

Verification Challenges

- Rebuilding Gen7 formal environments from scratch is:
  - Time-consuming , Resource-intensive, Risk-prone
- Aggressive test chip milestone date
- Changes in verification team who are not familiar with existing environments

Solution - A gearbox method

- Use the **gearbox** technique to reuse the Gen6 checker, which operates on a fast clock. The DUT is a Gen7 design that runs on a slow clock. Note that the data width has doubled in Gen7.



Why Gearbox?

Parameterization = High Risk

- 1000's of lines of tightly coupled Gen6 code
- Complex FSM aux code, assertions
- Refactoring → requires heavy revalidation

Time-critical: Gen7 test chip near code freeze

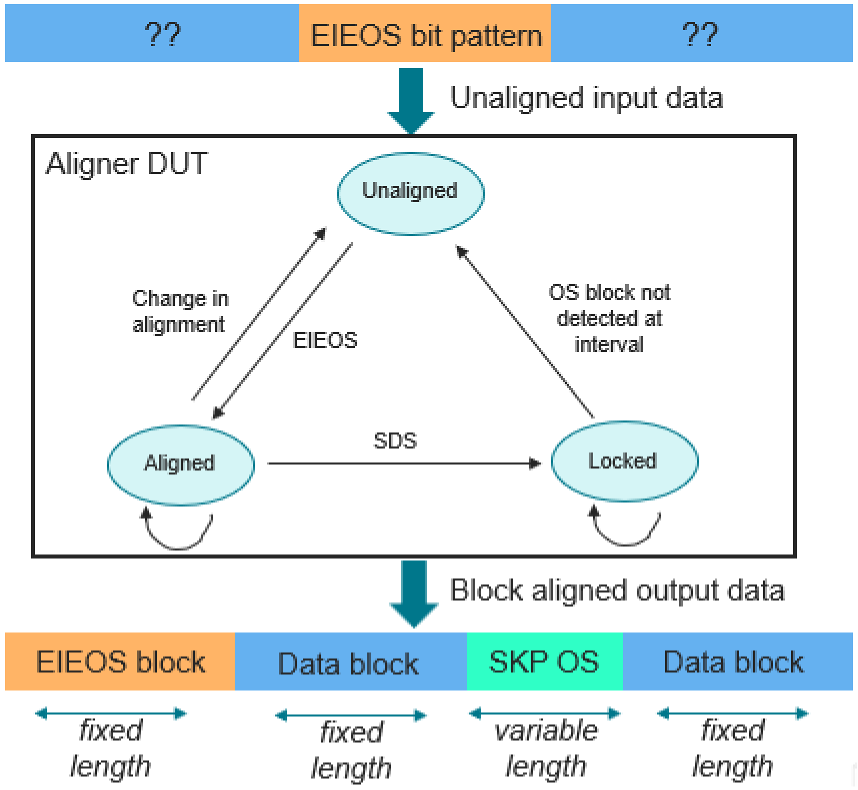
Gearbox Advantages

- Acts as **translation layer** between Gen7 DUT & Gen6 formal environment.
- Preserves existing verification logic
- Minimal code changes, faster onboarding
- Maintains **assertion quality** & coverage

Outcome - **Faster turnaround** under stringent timelines & **No compromise** on verification quality

Implementation: Case study 1 - Aligner

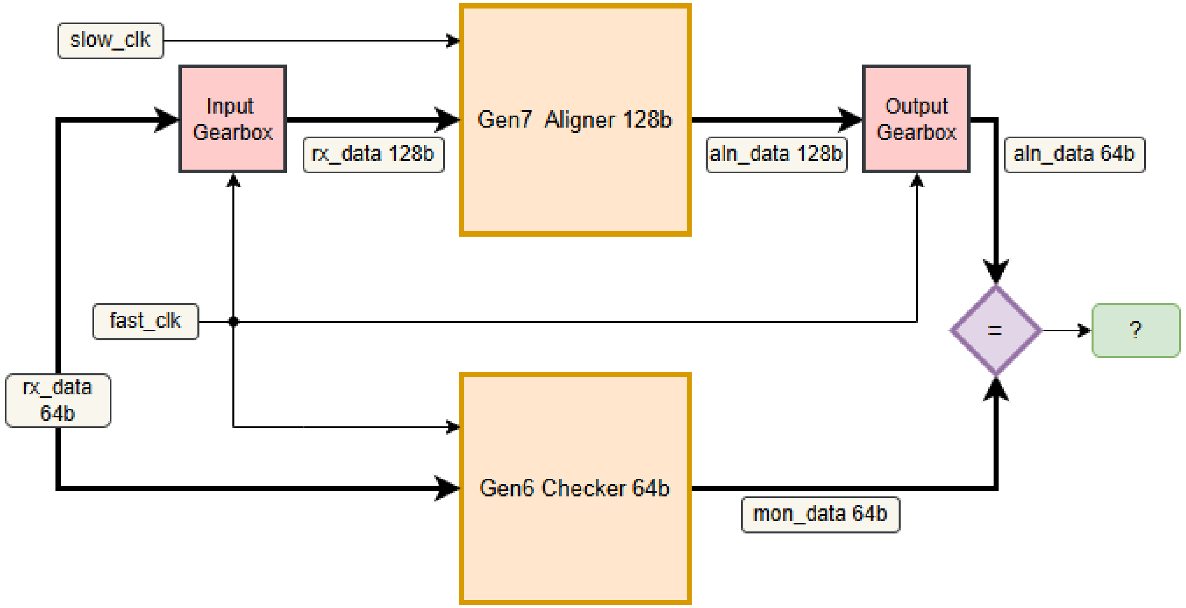
- **Role in PCIe PCS**
  - Positioned at the **frontline of the Physical Layer**
  - Receives **unaligned 128-bit data** from PMA
  - Ensures **block boundary recovery** and **protocol compliance** before passing data upstream
- **Key Functionalities**
  - **Data Alignment & Boundary Recovery**
    - Detect EIEOS markers as per PCIe Gen7 spec
    - Realign 128-bit data to byte and block boundaries
    - Maintain dynamic alignment FSM for lock recovery
  - **Skip Detection**
    - Identify SKP ordered sets for clock compensation
  - **Error Correction**
    - Detect anomalies and apply spec-defined correction schemes



- **Design Challenges**
  - Operates in high-speed environment (128 GT/s) and must handle channel-induced disturbances and signal integrity issues

Aligner verification strategy

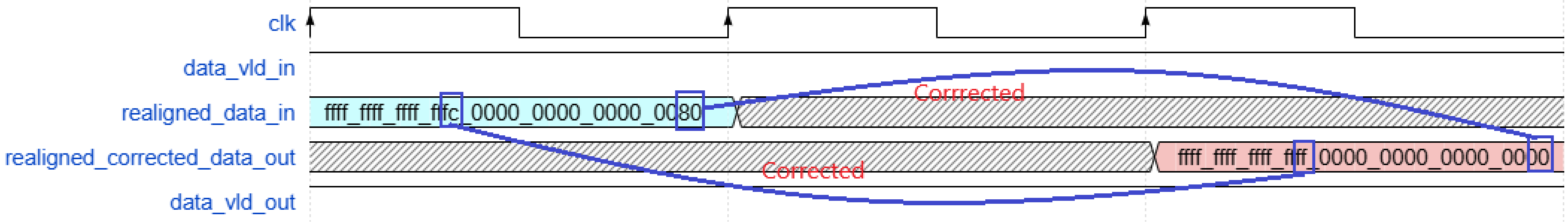
- **Objective**
  - **Reuse Gen6 formal environment** for Gen7 verification
  - Reduce **setup effort** and **time-to-verify**
- **Challenges & Solutions**
  - **Input Data Synchronization**
    - Gen7: **128-bit width**, Gen6 checker: **64-bit**
    - Introduced **gearbox** for data width conversion + assumptions (data & control inputs)
  - **Checker Updates**
    - Tuned **auxiliary logic** in checker
    - Added **new assertions** at DUT interface for gearbox integrity check and to find basic issues at early stages
- **Implementation Details**
  - **Gearbox-based Architecture:** Input: **64 → 128 bits** (formal env → DUT), Output: **128 → 64 bits** (DUT → checker)
  - **Cover Properties:** Ensure **valid input scenarios** are not omitted
  - **Relaxed Constraints:** Allow **bit-error scenarios** (inside & outside correctable limits)



Bug examples

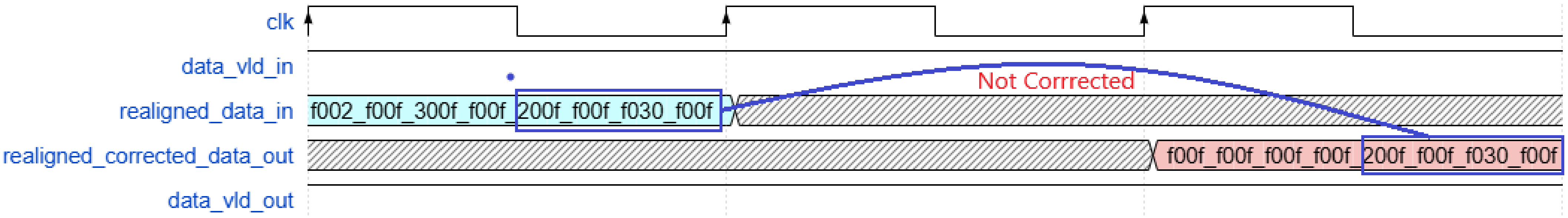
Bug Example 1: Issue: DUT performs **unexpected EIEOS correction**

- **Spec Rule:** EIEOS valid if: 1) ≥ 5 consecutive symbols match EIEOS pattern and 2) symbol0 or symbol8 must be a valid EIEOS symbol
- **Observed:** DUT corrected even though **symbol0 & symbol1 were not valid EIEOS symbols**



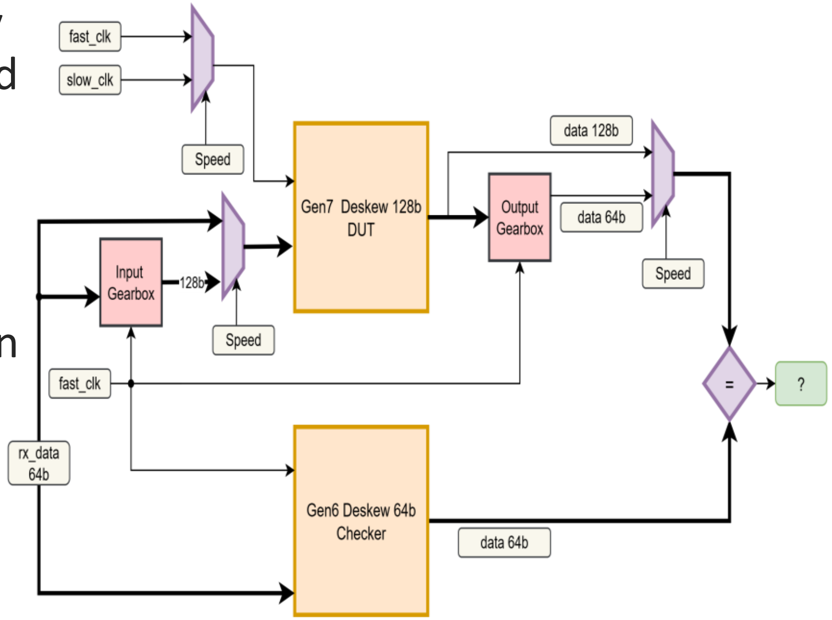
Bug Example 2: Issue: DUT **fails to correct first half of an EIOS block**

- **Spec Rule:** EIOS valid if: 1) ≥ 5 consecutive symbols match EIOS pattern and 2) symbol0 or symbol8 must be a valid EIOS symbol
- **Observed:** Received stream met correctable limits, but DUT didn't correct



Case study 2 - Deskew

- **Role in PCIe**
  - The deskew block compensates for timing differences between lanes by realigning the data streams, ensuring all lanes are correctly synchronized before further processing
- **Verification strategy**
  - The Gen6 formal environment is complex with 9K lines of code
  - Attempting to refactor the entire environment through parameterization would introduce significant risk and extensive engineering effort
  - Inserting a gearbox wrapper between the Gen7 DUT and the existing Gen6 formal environment, preserved the core structure of the verification environment and avoided invasive rewrites



- **Verification challenges**
  - Differences between intervals of Ordered Sets between Gen6 and Gen7
  - Only Gen7 speed uses all 128bits. The gearbox layer had to be bypassed for Gen1 to Gen6 speeds.
  - DUT contains alignment FIFOs that doubled in size between generations. Large FIFOs are not formal friendly structures and can impact proof times. Some tuning/reworking of impacted assertions.

Deskew- Example Assertion

- Assertion to check all lanes are correctly aligned by using special constraint setup
  - All lanes receive the same data pattern and just have varying degrees of skew inserted by the formal environment. This setup is unique to this 1 assertion and controlled by a task.
  - Exhaustive nature of formal verification ensures all permutations of skew between the lanes are evaluated
  - 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

```
generate for (ln_idx=0; ln_idx< LINK_WIDTH; ln_idx=ln_idx+1)
  begin : deskew_lane_gen
    assert_data_aligned_after_deskew_done: assert property (
 data_valid_out|->
 data_out[LANE_WIDTH*ln_idx+63:LANE_WIDTH*ln_idx] == data_out[63:0]);
  end // deskew_lane_gen
 endgenerate
```

Example Input data stream per lane

Start Of Data Stream (SDS) used for aligning

Lane 0 (zero skew): SDS 1 2 3 4 5 6 7 8 9 10

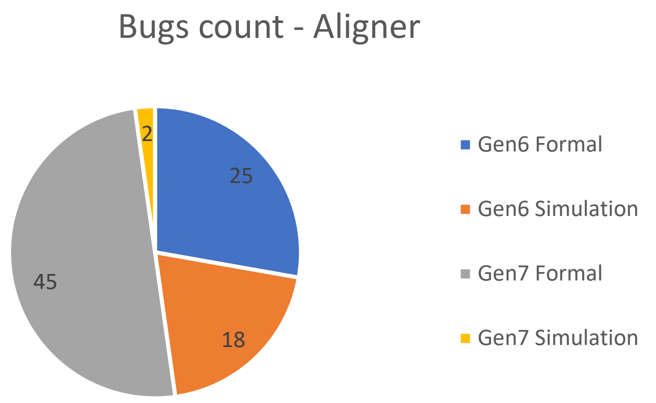
Lane 1 (variable skew):   SDS 1 2 3 4 5 6 7 8 9 10

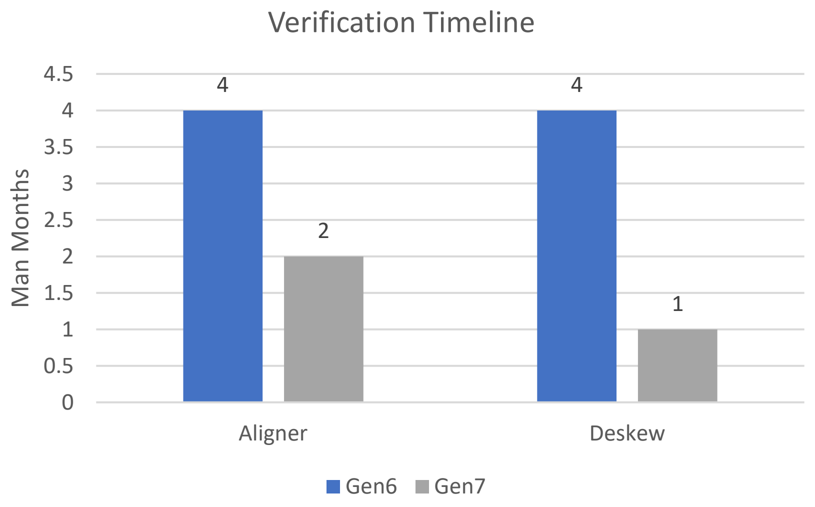
Expected output after deskew process complete

Lane 0 : SDS 1 2 3 4 5 6 7 8 9 10

Lane 1 : SDS 1 2 3 4 5 6 7 8 9 10

Results





Aligner results:

- *Gen7 Aligner*
  - Formal verification used as primary strategy
  - 45 RTL bugs found via formal (minimal simulation)
  - Legacy Gen6 assertions caught most bugs
  - New assertions at Gen7 output interface found initial basic issues
- *Gen6 Aligner*
  - Started with simulation: 18 bugs found
  - Formal added later: 25 more bugs, many noise-related
  - No new bugs post formal sign-off

Deskew results:

- No bugs found at Gen7 using formal verification. 19 bugs from Gen6.
- Reused Gen6 assertions passed with 100% functional & code coverage
- No issues found with block in later top-level simulation
- Only minor environment changes needed going from Gen6 to Gen7

General Observations:

- Formal methods excelled at catching corner case and complex bugs
- Sequential depth of proofs same between Gen6 and Gen7
- Similar proof times between Gen6 and Gen7 variants

Summary

- Successfully bridged PCIe Gen6 and Gen7 verification using reused formal checkers
- Formal achieved high bug detection which reduces debug time during later simulation
- Maintained full coverage levels from formal work, reducing simulation workload
- Strategy proved scalable, efficient, and sustainable for future designs
- Strategy can be applied by new verification team members without lengthy ramp up
- Promoted a consistent and maintainable verification environment

Thank You

Any Questions?