



Merging to unified Pre-sil Post-sil validation
environment

Assaf Eldan

Cadence Club Verification
July 11th 2006

Content

- **Motivation**
- **Moving from pre to post silicon validation environment**
- **Results**
- **Risks, Limitations and Potential Future Improvements**
- **Summary**

Motivation

- Schedule
 - 12 MM were required to support new project modifications.
 - Needed immediate support for FPGA emulation model of the design.
- Unified environment for Pre-Post Silicon
 - Reuse.
 - Flexible human resource control.
 - Easier failure reproduction in simulation.
- Previous Post-Sil tool limitations:
 - Limited random capabilities.
 - No coverage collection and tracking capabilities.
 - Limited interactive debug capabilities.
 - Performance problems.

Moving from pre to post silicon validation environment (High Level)

- Code Replacement

- Add #ifdef around white box code (e.g. Internal signals).
- Remove/Replace BFM's

- Performance

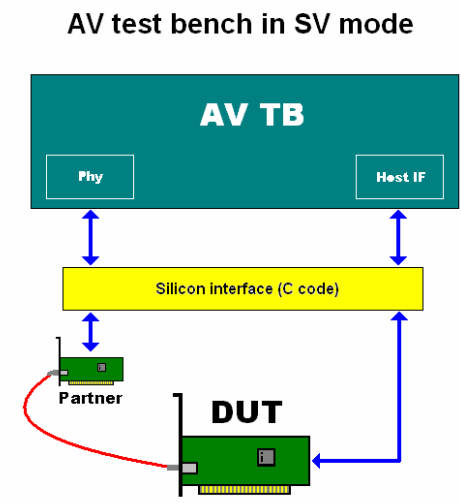
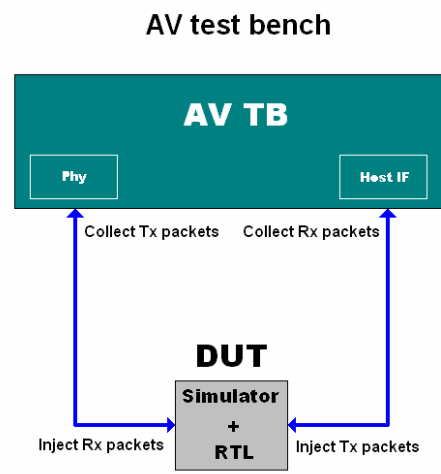
- No generation OTF
- Avoid clock cycles

- Coverage

- Remove irrelevant files.
- Make sure that relevant events are emitted.
- Add post-sil related coverage points.

- Checkers

- Remove/Replace checkers which are white box dependant.



Moving from pre to post silicon validation environment (Details)

- Phase A – Getting an environment which works on silicon.
 - Remove/Disable EVCs which are provided by the HW of the post-sil platform (e.g. PCI-E BFM).
 - Implement memory access methods using Specman C I/F (WR/RD register, WR/RD mem).
 - Override white box code with the above C functions.
 - Replace PHY EVC low level functionality with a “Link Partner”.
- The result of this stage is a working environment which works in lab but is still heavy, due to inefficient (for silicon validation) code which is aimed for simulation and not for real time testing.

Moving from pre to post silicon validation environment (Details)

- Phase B – Achieving real time performance.
 - Test time of our system is divided to two:
 - Non real time code – generation, configuration, checkers and coverage
 - Real time code - transmission and reception of packets.
 - Main goal is to minimize the real time run overhead. This can be done by:
 - Removing generation from run time areas.
 - Minimizing GC by Avoiding memory allocation during run time.
 - Minimizing TCM usage.
 - In our environment this was achieved by:
 - Moving sequenced to pre-run generation.
 - Building TX/RX memory structures (Descriptor rings and data buffers) in advance and retransmitting the same data many times.
 - Writing a “mirror” scoreboard, which uses same data but in a more efficient way.
 - Comparing data (List of bytes) in C rather than in E.
 - Implementing our own non TCM scheduler which calls the main TX/RX drivers (Which are also post-sil specific).

Moving from pre to post silicon validation environment (Details)

- Phase C – Tying the edges
 - Check that coverage and checker events are emitted by the modified environment.
 - Clean white box coverage and checkers.
 - Add post-sil specific coverage points and checkers.
 - Verify that post-sil tests can run in pre-sil and vice versa.

Results

- General:

- Schedule
 - Effort for first integration with FGPA code – 3 MM.
 - Most of it is one time task. For next product expected duration is 2-3 weeks.
- Smooth integration
 - 4 days vs 3 weeks in previous projects for initial integration.
 - Advance feature integration took few hours per feature (vs several days in previous projects).
- New code written
 - C code - 2-3% of the environment.
 - E code – 10-15% of the environment.

- Performance:

- With partial data match wire speed is met.
- Same with full matching of every second packet. Working in order to improve full match performance.
- Performance is better than in previous tool.

Results (Cont)

- Stability:

- Environment and regression are stable. Overnight tests pass with no issues (Billions of transactions).
- Achieved stable state relatively fast (few weeks).

- Coverage:

- Most coverage points are reused from pre-sil environment.
- First time in our post-sil environment that coverage analysis exists.

- Simpler Debug:

- Design engineers are familiar with the environment and can easily navigate in the environment data structure tree.
- Specman supports breakpoints and interactive mode (Can be done in other tools as well but we didn't have this option in the past).
- Reproducing failures in pre-sil is simpler and the debug can be done in simulation.

Risks and limitations

- One environment for pre-sil and post-sil
 - No double check of the silicon.
 - Impact: pre-sil bugs and holes are copied to the post-sil environment.
 - Mitigation:
 - post-sil team defined its own test plan separately.
 - In most cases the tests were written from scratch.
 - On silicon both relevant pre-sil tests and the new post-sil tests ran.
 - The environment areas which were reused were reviewed based on risk assessment.

Risks and limitations (Cont)

- Multi Specman machines
 - Specman doesn't support multi CPU and also can't easily work on several machines (Controller – Agent mode)
 - Impact: Partners must be on the same machine, all drivers runs on the same CPU. Both issues loads the CPU and limits the performance.
 - Mitigation:
 - In current project can stay with current status.
 - For next project checking option for running the same test on several machines in order to get the same generation. Disable the non relevant parts of the test in each machine (i.e. In the DUT machine, the partner part will be removed), and use external trigger to sync between the test phases.

Summary

- Project started as a proof of concept over a FPGA board.
- Resource limitation played major part in this decision.
- After several weeks it was decided to use it for silicon as well.
- Generally, all goals were met, current environment is stronger and less costly than previous one:
 - Stronger generation.
 - Coverage tracking (controlled by VManager).
 - Stability (Which is achieved fast).
 - Relatively low effort.

Summary

- Before deciding if this approach can be used for a specific project the following things should be considered:
 - What the required performance is and if this solution can reach it (Main limitation is single threaded system).
 - Is there a way to control the interfaces, directly or indirectly with an application which runs on the DUT machine?
 - Is there a strong pre-sil environment? What is the risk of duplicating critical bugs and is there a reasonable mitigation.
 - What is the status of the current post-sil tools and what is the required modification of these tools when moving between projects.
- Bottom line is that it is feasible to Specman simulation environment for post-sil. Based on the points above one should decide whether to use this option or not, but the important thing is that it can be done and in our case we consider it as a great success.



Q & A