

Transaction-Based Acceleration—Strong Ammunition In Any Verification Arsenal

Chandrasekhar Poorna
Principal Engineer
Broadcom Corp
San Jose, CA USA

Varun Gupta
Sr. Field Applications Engineer
Cadence Design Systems
San Jose, CA USA

Raj Mathur
Sr. Product Marketing Manager
Cadence Design Systems
San Jose, CA USA

ABSTRACT

Register transfer level (RTL) simulation run times are severely impacted by the verification requirements of today’s complex IC designs. Due to fierce market demands of increased functionality, the need to serve multiple applications with the same core design, and shrinking time-to-market windows, it has become increasingly challenging to complete the verification plan on time. Time-critical tests, with requisite scoreboard monitoring, take days of simulation run time, which can extend the time required to meet high quality assurance milestones.

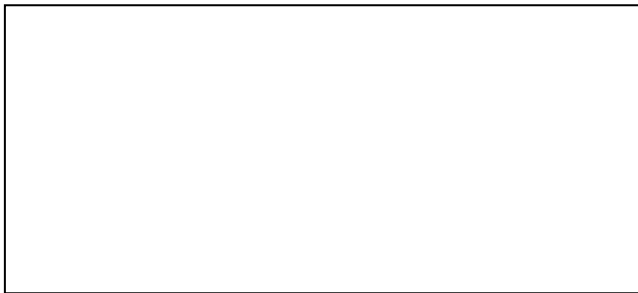
Complementing simulation-based verification with in-circuit emulation (ICE) and/or FPGA-based prototyping can provide much-needed performance relief for software validation and enhance the ability to verify with real-world directed stimulus and response. Both of these verification modes predominantly require full design synthesizability and the availability of software drivers to achieve the higher quality modeling. Hardware acceleration of the existing simulation test bench not only allows the discovery of deep corner case bugs while maintaining metric-driven verification methods, but also serves as a unique bridge between simulation and in-circuit emulation. This early-phase acceleration, which enables gradual movement of behavioral design blocks into the synthesizable domain, allows smoother transition into and faster bring-up of in-circuit emulation, thereby improving overall productivity and reducing time-to-market.

General Terms

Management, Performance, Design, Economics, Standardization, Languages, Verification

Keywords

verification, acceleration, emulation, FPGA-based prototyping, transaction, system-level verification, hardware/software verification, network switch, Ethernet



1. INTRODUCTION

This paper describes the verification environment of an Ethernet switch application that was devised for both simulation and hardware acceleration. Both the challenges and the solutions to verifying millions of Ethernet packets with long verification runs embedded with monitors and checkers in the test bench are described. An example of the use of transaction-based acceleration (TBA) based on a Standard Co-Emulation Modeling Interface (SCE-MI) [1] [2] for accelerated communication between a workstation and the hardware accelerator is provided.

This paper shows how designers can accelerate their SystemVerilog-based test bench run and achieve results congruent with register transfer level (RTL) simulation while preserving select checkers and assertions in the accelerated run. Performance of hundreds of times better than simulation provides the opportunity to not only run long sequences, but also perform diagnostic testing and begin firmware development early. To achieve this performance within a practical schedule requires the up-front investment in a test bench architecture/infrastructure that allows easy transition to acceleration mode, which is a recommended milestone before performing in-circuit emulation and/or FPGA-based prototyping.

2. VERIFICATION CHALLENGES

To provide practical insights into the verification challenges that design and verification engineers face, an Ethernet network switch application (see Figure 1) is used as a reference.

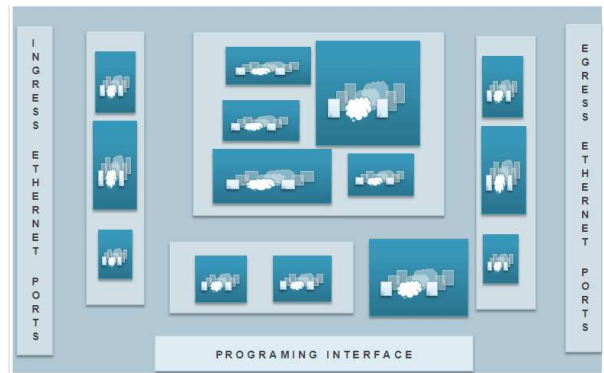


Figure 1: Ethernet switch chip-level diagram

The application consists of multiple programmable Ethernet ports, as represented by the INGRESS and EGRESS blocks, and a custom programming interface that programs the switch before the packets are sent through.

In general, the challenges of system-level verification of hardware and software are magnified by the sheer volume of data that must be processed to achieve a coverage level acceptable for RTL sign-off. In addition, the hardware verification of the design has different tradeoffs at different hierarchical levels.

2.1 Hierarchical Partitioning and Tradeoffs

The hardware verification process is further divided into sub-block, block, chip-level, and system-level categories. For instance, at the sub-block level for the SystemVerilog-based test bench (see Figure 2), applying RTL simulation is typically a veritable solution, providing the necessary visibility into the design for debugging and observing embedded assertions that fire when necessary. This is a plausible solution for sub-block sizes that are smaller than 1 million gates, and if the simulation tests can be run in minutes to hours. When simulation run times start to get longer than few hours, however, productivity is adversely impacted, and simulation alone becomes impractical. Such is the case typically observed at the block level.

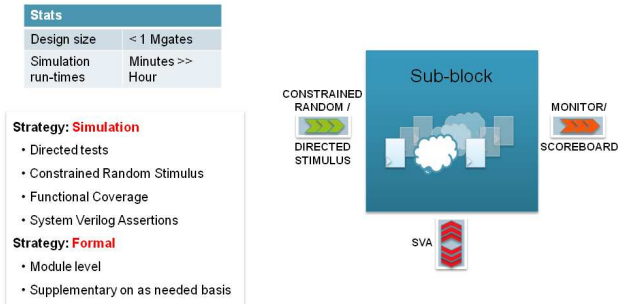


Figure 2: Module/Sub-block-level verification

In addition, at the block level, in which gate counts vary widely within the range of 5–20 M gates, the longer debug cycle begins to hamper the designers’ productivity in running just RTL simulation. Formal techniques have been applied, but the state-space explosion also begins to render formal techniques of limited value. For some, in-circuit emulation(ICE) is an alternative verification vehicle at the block level, but due to proprietary interfaces at the internal blocks, in-circuit emulation is impractical for our block-level verification. ICE, for switching chips in the group is performed at the chip and system level using off-the-shelf Speed-Bridges for PCIe and Ethernet.

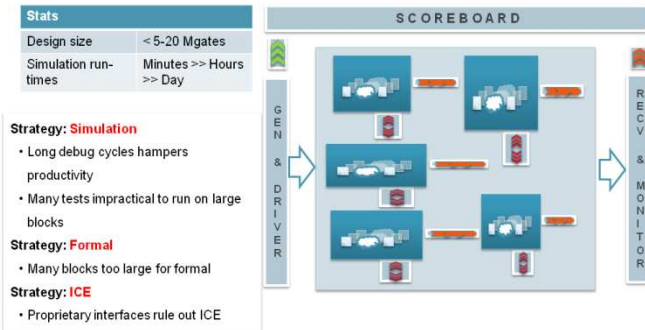


Figure 3: Block-level verification

To run in-circuit emulation at the chip/system level, almost all blocks are required to be synthesis-ready, and real software drivers and firmware are required to interact with the hardware

design. ASIC prototyping leveraging off-the-shelf FPGAs is a complementary addition to in-circuit emulation, but usually additional resources with long bring-up time and limited debug visibility are needed if the design in the FPGA-based prototype is not preverified in an in-circuit emulator.

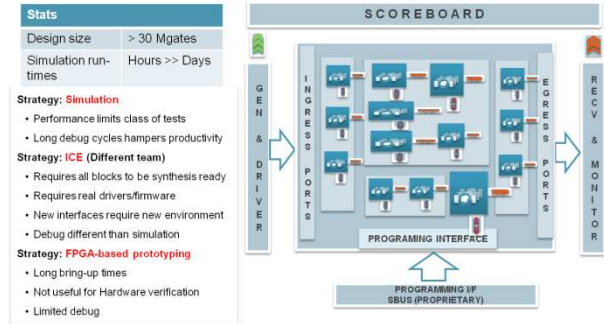


Figure 4: Chip/system-level verification

In summary, different verification techniques provide different values at different levels of design hierarchy. Figure 5 summarizes the experiences observed during the system-level verification of the Ethernet network switch (your experience will vary with your unique design, test, and verification environment).

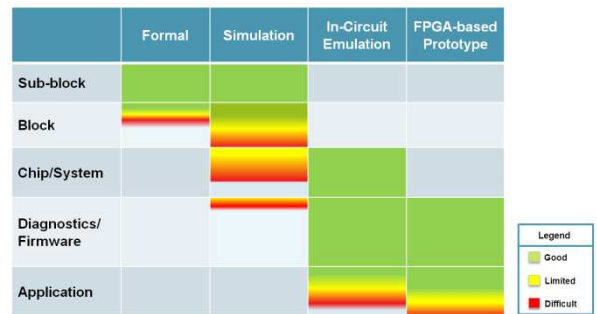


Figure 5: Summary of verification solutions

At the block level, the need to simulate longer tests requires a faster performance platform than RTL simulation, one that can handle proprietary interfaces and leverage the existing test bench environment defined for simulation while reducing the debug cycle.

At the chip level, there is a need to run deeper tests requiring a fast performance engine running earlier in the project cycle.

2.2 Simulate and Debug Phase

Finally, at the project level, the completion of all test plans (sub-block, block, chip, and system) is dependent on the life cycle that a test goes through. At a generic level, most tests for the Ethernet switch application have the following phases in their life cycle:

1. Development. The intent of the test-plan is translated to executable code.
 - The executable code is run on a simulator.
2. Simulate and Debug
 - Iterate between simulation and debug to ensure that the stimulus matches the intent.

- Iterate between the simulation run and debug to ensure that assertions, checkers, scoreboards, and functional and code coverage metrics are able to track and capture the intended behavior correctly and report errors as necessary.
 - Iterate between simulation and debug to cover variations of a set of parameters defined for the test.
3. Passed and Completed. All variations of the test as defined by the test plan have been covered and captured as a quantifiable metric.

The above life cycle of a test is dominated by the iterative “**Simulate and debug**” phase and is widely recognized as a critical path in any verification process. Looking at this phase further, it is easy to see that “simulation time” is machine/simulator-dependent, and hence can easily be quantified; that is, test N takes an average of T seconds to complete on an XYZ simulator on a workstation running at F GHz frequency with M GB of swap memory and C number of cores. This phase usually scales directly as the size of the entity being verified increases. Therefore, its impact on the block-level and chip-level test benches mentioned earlier is much higher than on a sub-block test bench. On the other hand, the “debug time” is highly qualitative because, although it is directly gated by the ability to run simulation on the test and detect erroneous behavior (simulation time), it is also indirectly dependent on a number of other factors such as the nature of error, design visibility level needed, availability of the RTL designer, experience of the verification engineer, and more. When the simulation time increases, as it does at the block and chip level, it exacerbates the qualitative factors that affect the debug time. This is so because when a test takes a long time to run, verification engineers typically shift their attention to other tasks. Later, when the test is completed, the verification engineer may not be able to immediately start debugging failed tests because of the demands of the other tasks.

2.3 Hardware Acceleration Platform

Given the stated challenges and the fact that the “**simulate and debug**” iterative cycle for block-level and chip-level test benches is the major time constraint that determines the time from test development to completion, the need to lower the simulation time is acutely felt. Hardware acceleration platforms are capable of meeting the need to lower simulation time. But to be truly effective to a verification team, the hardware acceleration platform must address the following additional requirements.

- Reuse/leverage the existing test bench environment.
- Preserve the functionality, randomization, controllability, and testability of the existing environment to the maximum extent possible.
- Preserve most of the existing metrics for measurement of test plan completion.
- Mirror the design visibility and debug-ability features of RTL simulation.
- Support commonly used test bench paradigms and structures (behavioral coding, error messaging, and mailbox-like mechanisms to pass transactions).
- Avoid the need for vendor-specific and application-specific APIs to interact with the hardware.

- Be portable across simulators and hardware platforms with minimal change.
- Provide the flexibility in making changes to the verification environment to trade off capacity versus performance.
- Provide the ability to be able to scale from the block level to the chip level seamlessly, yet have the granularity for multiple test benches to use the system simultaneously (for example, both the block-level and chip-level test benches could be running simultaneously as long as they use different physical partitions).

To maintain vendor independence and ensure portability of the methodology, an API defined by the SCE-MI standard from Accellera is requisite to interact with the hardware platform. Also, to achieve maximum performance, an acceleration methodology referred to as **Transaction-Based Acceleration** was chosen. This method uses SCE-MI 2.0 transaction pipes extensively.

3. TRANSACTION-BASED ACCELERATION

The most basic form of simulation acceleration is commonly referred to as signal-based acceleration (SBA). In this acceleration mode (see Figure 6), the design under test (DUT) is compiled into a specialized hardware while the test bench executes in a workstation driving stimulus into the DUT over a communication channel. The overall simulation performance is determined by the formula:

$$T_{\text{simulation}} = T_{\text{Test-bench}} + T_{\text{DUT}} + T_{\text{communication}}$$

As the DUT executes in the specialized hardware, T_{DUT} practically reduces to an insignificant factor, leaving the overall simulation time to be determined by the time spent in the test bench and the time to exchange bit-by-bit signals between the test bench and the DUT.

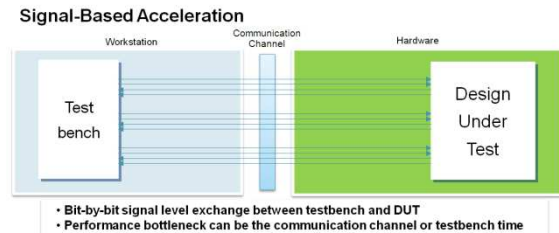


Figure 6. Signal-based acceleration (SBA)

Transaction-based acceleration (see Figure 7) is a simulation acceleration method that improves the performance by further reducing the two remaining components: $T_{\text{communication}}$ and $T_{\text{Test-bench}}$.

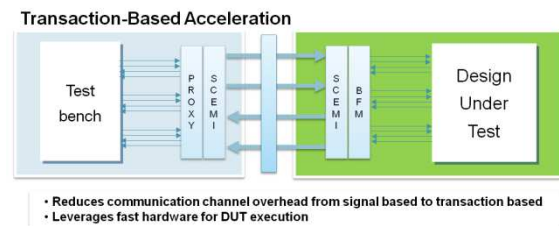


Figure 7. Transaction-based acceleration (TBA)

The first step is to make an informed decision to partition the test-bench functionality as a generator and driver layer and move as many of the simple signal-level interactions as possible to the driver layer. Thus, in this scheme, the generator is responsible for high-level functions such as randomizing and generating transactions, whereas the driver is responsible for receiving the transactions from the generator and performing bus-level functions. In addition, simple threads such as random data/CRC generation/checking, watchdog timers, and low-level arbitration can also be moved to the driver layer. To gain insight into how this is achieved with the Ethernet switch design, a profile of the RTL simulation is needed (see Figure 8) to identify how much acceleration can actually be achieved and to identify the pieces that are compiled into the hardware.

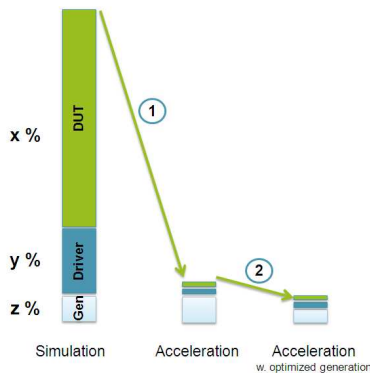


Figure 8. Profiling for RTL vs. accelerated simulation

Figure 8 shows that the RTL simulation time profile consists of three main portions: a stimulus generator, a driver, and the DUT. The test bench itself consists of the two components: Generator and Driver. With some support for behavioral coding constructs, the driver layer along with the DUT can be ported to the hardware accelerator leveraging the multi-parallel processing grid engine in the accelerator for faster execution. Now, because the information that is transferred across the communication pipe from the generator to the driver is minimal (control fields that need randomization only), the $T_{\text{communication}}$ factor is further reduced. Similarly, as the number of threads that run in the Generator are far less and infrequent compared to when only the DUT was accelerated, the $T_{\text{Test-bench}}$ time component is also reduced. The performance can further be improved by optimizing the generation portion by streamlining the randomization process and minimizing interrupts from the DUT side, allowing the generator to stream transactions independent of the DUT side.

3.1 Changes to Test Bench for Hardware Acceleration

A common set of changes are needed in an existing test bench to enable it to undergo acceleration. Most of these changes are enabled via the support for SystemVerilog DPI functions and SCE-MI 2.0-based transaction pipes. The choice between using a DPI function or an SCE-MI transaction pipe to interact between the test bench running on a workstation and the driver layer+DUT running in the hardware platform, primarily depends on the nature of the specific interaction. DPI functions are best suited for use in cases in which the interaction is handshake-based, needing increased communication between the test bench and the acceleratable portion. Transaction pipes are best suited for

streaming transactions that need little or no interaction between the test bench and hardware side.

To achieve maximum performance, it is recommended to move all logic dependent on timing delays into the driver portion (which resides on the acceleration platform). Because the #delays are not synthesizable, all delays in the driver should be specified in terms of clocks. This rule guides the best partitioning methodology and is the key to the success of the acceleration effort.

With the time progression happening only on the hardware platform, the test-bench side uses events (rather than time) to communicate and synchronize with the hardware side. For the SCEMI pipes-based communication channel, such events are SCEMI pipe-full and pipe-empty conditions, which are serviced by the user-defined callback functions. For the DPI mode of communication, a DPI synchronization function can be implemented to generate such events.

Infrequent and asynchronous events such as resets and interrupts, which require immediate servicing, are best handled using DPI export functions that are called in the test-bench side, but implemented in the DUT side.

Clocking specifications are closely tied to the hardware platform, and, hence need to be specified in the required format. Additionally, tuning the period and phase relationship between different clocks in a design can provide large performance benefits. As a result, it is beneficial to isolate all the clock-generation logic to a single module or file.

For the Ethernet switch application, the programming interface programs the switch before the packets are sent through. This programming interface uses a handshake-based (reg-wr/rd-ack) protocol. Therefore, DPI functions were found to be suitable and were implemented for use on the hardware platform.

For regular traffic transactions, which form the bulk of the interaction between the hardware and software, the test-bench side needed minimal interaction, and hence was implemented as SCEMI transaction pipes for maximum throughput. This implementation allows transactions to be streamed from the test bench to the DUT.

A statistics-gathering and reporting block was added, which periodically (programmable) updates bandwidth and other statistics to the test-bench side for further processing.

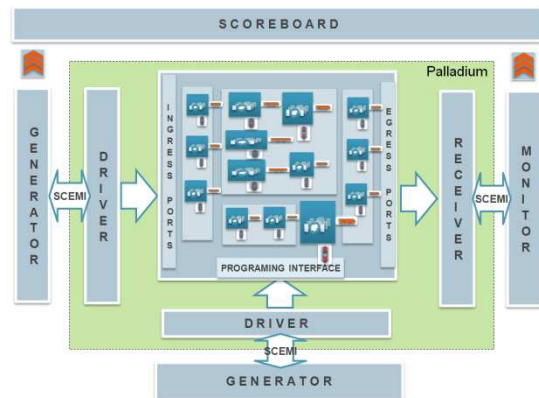


Figure 9. Test-bench TBA hardware acceleration

3.2 Monitors and Checkers

A major consideration while enabling acceleration on the existing test bench was to preserve the monitoring and checking functionality of the legacy end-to-end checkers in the environment. This functionality was achieved by following the same two-layer approach of moving the bus-level functionality and data checking to a monitor, and the higher-level functionality such as scoreboarding, test end checking, and so on to the checker. Monitors, which are hardware acceleratable and connect to the RTL interface to be monitored, are responsible for implementing a simple state machine that can build up a message with the relevant fields based on monitoring a complete transaction over multiple cycles. Once a message is complete, it is passed to the checker level via an SCE-MI transaction pipe for further processing. The checker receives the message and converts the message fields to a format that is used internally and continues with its normal checks agnostic to whether it is accelerated or is pure simulation.

3.3 Use of SystemVerilog Assertions

Hardware acceleration platforms also support SystemVerilog concurrent-assertion statements. This allows offloading the temporal checking in the monitors to assertion modules. Because assertions are compiled into the hardware domain, there is no appreciable degradation in performance. SystemVerilog Assertion modules must be compiled as separate entities during the synthesis phase and then bound into the design at the hardware compile stage. Compiled assertions can be turned on or off at run time to keep track of the assertion count.

4. RESULTS

Hardware acceleration using TBA was applied to two test benches; one at the block level of an Ethernet switching chip (Figure 10), and the other at the full chip level for a different chip (Figure 11). With changes to support acceleration and performance optimizations in place, the following benchmarks were recorded.

Block-level Results (different chip)	
Design Size	~20 MGates
Test	1.2 Million packets
VCS simulation	1.04 packets per second
VCS acceleration with Palladium	304 packets per second
Simulation Speedup	292x
Compile time (RTL-to-debug)	1.5 hr with 1 workstation

In the case of the block-level test bench, a performance speedup of 292x was achieved, good visibility of the design under test and the test bench to quickly find design issues was maintained, plus a fast compile time to iterate through the flow when RTL changes were needed was maintained.

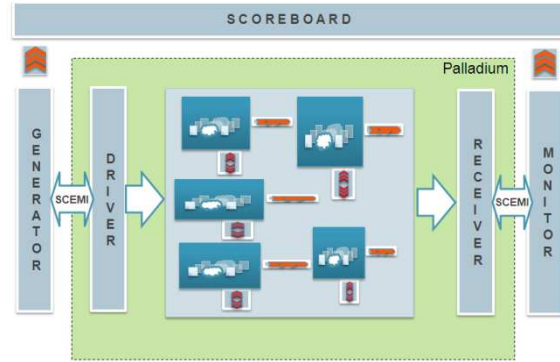


Figure 10. Block-level TBA hardware acceleration

At the chip level (for a different chip), changes to interfaces (to be acceleration compliant) were made at the full-chip digital level, which produced the following results:

Chip-level Results	
Design Size	~32 MGates
Test	1600 packets
VCS simulation	16 hrs
VCS acceleration with Palladium	250 seconds
Simulation Speedup	230x
Compile time (RTL-to-debug)	1.5 hr with 1 workstation

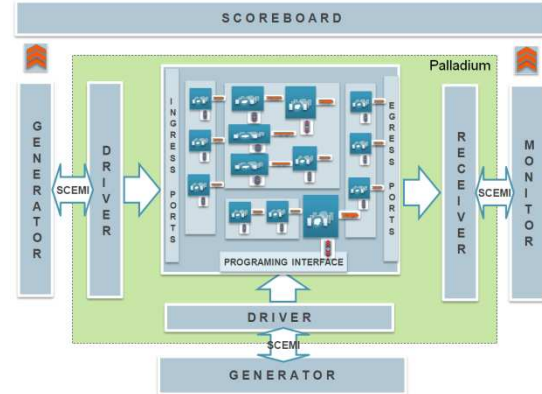


Figure 11. Full-chip digital-level TBA acceleration

5. INVESTMENTS AND ROI

The following table summarizes the current verification solutions that are applied at Broadcom.

	Single platform				
	Formal	Simulation	Transaction-Based Acceleration	In-Circuit Emulation	FPGA-based Prototype
Sub-block	Green	Green	Green	Green	Green
Block	Green	Yellow	Green	Green	Green
Chip/System	Green	Yellow	Green	Green	Green
Diagnostics/Firmware	Green	Yellow	Green	Green	Green
Application	Green	Yellow	Green	Yellow	Green

Figure 12. Summary of Broadcom verification solutions

Transaction-based acceleration has provided much-needed relief at the block-level and chip-level verification. Benchmark tests that

formerly required about a day or more can now be run within minutes. And more importantly, it is now possible to run tests that previously would not have been considered. This capability helps achieve much more debugging earlier in the project cycle and at an accelerated pace, which allows a cleaner handoff to the emulation team who can now focus on running longer tests and not have to worry about catching simple bugs. Overall, and most importantly, transaction-based acceleration helps achieve our time-to-market objectives.

Different hardware-assisted verification products in the marketplace support transaction-based acceleration with varying degrees of functionality, usability, reliability, performance, and support. To maximize the ROI, it is recommended to identify a hardware acceleration platform that can support both transaction-based acceleration and in-circuit emulation.

At the outset, newcomers to hardware-assisted verification should invest sufficient time to learn the methodology and how to use the tools. Also, they should work with EDA vendors to reduce the amount of time it takes to acquire such knowledge. Because the verification environment is reusable, once sound procedures are established, subsequent designs can be verified with much less effort than was initially required.

6. CONCLUSION

Based on our experience with transaction-based acceleration, we find it an essential tool that allows us to compress our verification process. It also helps verify system-level scenarios that were initially out of the reach of simulation due to protracted simulation run time. Using the accelerated simulation platform as a step before starting emulation helps in the delivery of a cleaner database to emulation, which enables the emulation team to focus on chip and inter-chip system-level testing. Of particular value was the ability to run the same tests in a simulation environment for debugging, and then to accelerate by running on the hardware platform. We have extended the test bench to newer chips in production. The platform is extensible and can scale with the increased complexity and functionality of our future chips. Later versions of the hardware platform enable further improvements in performance with increased visibility and design.

7. REFERENCES

- [1] Standard Co-Emulation Modeling Interface (SCE-MI) Reference Manual 2.1.
- [2] Standard Co-Emulation Modeling Interface (SCE-MI) Reference Manual 2.0.

TBA-ETP100-D4 January 13, 2011