# SYSTEM LEVEL ASSERTION-BASED VERIFICATION ENVIRONMENT FOR PCI/PCI-X AND PCI-EXPRESS

*Chien-Chih Yu, Kai-Hui Chang, Yi-Jong Yeh, and Sy-Yen Kuo*

Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan

**ABSTRACT**

Circuit design becomes more and more complicated in system on chip (SoC) due to the increasing capacity of integrating gates into one chip. Verification has taken a major portion of non-recurrent engineering (NRE) cost. Traditional register transfer level (RTL) verification method could take about 60% work of the entire design cycle. System level verification methodology can reduce the time to write a testbench and increase the functional coverage. In this paper, we developed a verification system for PCI/PCI-X and PCI-Express protocols. This system verifies the functionality of a protocol and manages all system resources. Besides, it provides cross-protocol checking mechanism to analyze the behaviors between different protocols and estimates the performance of the design under test.

## 1. INTRODUCTION

Modern system-on-chip (SoC) designs are increasing the complexity, and cost spent on verification skyrockets on each process. Most of current verification strategies use the simulation-based method and circuits for verification focus on register transfer level (RTL). While many theories and methods are developed to reduce the barrier of SoC design, verification can not keep track with the modern design flow especially in hardware-software co-verification applications. The challenge of verification is to speed up the verification process when circuit is in the prototype period without sacrificing functional coverage. Integration is another important issue. A system level function will be partitioned into several parts, and be implemented at the same time [1]. A good verification tool should provide an environment that can contain all components in one platform and verify all miscellaneous protocols, instructions and interfaces at the same time. In a system level environment, design under test will face more complicated interactions with other devices and the situation is closer to the real world.

Today, a single chip probably could be composed with several different IPs, and data transfer from one block to the other block of circuit by handshaking or some specific bus protocols. It implies that most chips use several different bus protocols to control each block of the circuit. Now, it is very difficult to verify the result by traditional RTL level verification [2].

There are various verification methodologies invented or refined form traditional methods including simulation-based verification, functional verification [3], formal verification [4], assertion-based verification and symbolic-based verification [5]. These popular methodologies can be categorized into two major groups: the first group can verify any design without simulation; formal verification belongs to this one, the other group still relies on simulation. Formal verification can guarantee the correctness of any design without test pattern, but this method is limited by the number of transition states of the design [6].

Assertion-based verification is one way to refine RTL level verification, and can reduce the effort to design testbenches. We developed an assertion-based system level verification tool to verify all functionalities of PCI, PCI-X, and PCI-Express protocols. This verification tool is an assertion-based functional verification tool, and it allows mixed protocol devices (PCI/PCI-X, and PCI-Express) to be verified in the same platform. The verification tool consists of bus functional model (BFM) for PCI, PCI-X, and PCI-Express, bus protocol monitor, device behavior checker, and Super monitor. This verification system has pseudo firmware level to support BIOS-like program. By applying firmware software, the verification system has the ability to dynamically control and manage all system resources. Super monitor provides a mechanism to record, trace, and analyze the transactions or bus behaviors across different protocol. A special database is built in the verification system, and this database collects important values from all BFMs, bus protocol monitors and End-to-End checker automatically.

## 2. VERILOG EXTENDED LIBRARY-TESTWIZARD

Current HDL languages such as Verilog and VHDL are lack of the ability to handle high level data structures; it's difficult to develop a complicated test testbench. We created a set of library called TestWizard, to enhance VerilogHDL for testbench development.

### 2.1 Overview of TestWizard

TestWizard is implemented as parts of Verilog HDL via PLI (Programming Language Interface). TestWizard can group variables to a structure, capture a series of variables and event changes, create time domain database which can be accessed by other high level languages, and profile specific performance defined by user. TestWizard greatly simplifies the modeling requirements necessary for functional verification such as coding tests generators, assertion checkers, monitors, timing verifiers, and architectural reference models.

### 2.2 Record Class

Record class is the core class to process the abstract data structure as what high level languages do. Record class allows users to define any record type they need. Designers can encapsulate various kinds of native data types of Verilog HDL to a new record type, and then allocate memory space for each record, and access these records by a uniform interface.

### 2.3 List Class

List is a new data type used to hold a collection of strings, numbers, and number range constants. Lists are immutable, unsorted and can contain constants or other lists. List can help engineers manage and control complex data and information more conveniently than a traditional way.

### 2.4 Temporary Assertion Class

Temporal assertion related functions verify that the system properties and assumptions have not been violated, including protocols, architecture, and algorithms. Specifically, in complex concurrent systems, interaction between protocol rules can be subtle and problems can be easily missed. Assertion checkers are comprised of 2 parts: 1) a specification describing the architecture's rules and properties – invariant and temporal conditional sequences; and 2) checks which compare these specifications to the simulation state of the hardware model. In the event of a miss-comparison, the check can make assertions – a conditional error notification generated to alert the verification engineer.

### 2.5 Transaction Logging Class

Transaction logging class provides the capability to store transaction-level data in a history which can be later accessed using index, record type, record value, and time-based searches. Automatic garbage collection is performed on transaction buffers using length and age criteria. Transaction logging is designed especially for control, management, analysis and effective search. This class is a time-domain with database-oriented management pool.

## 3. PCI/PCI-X AND PCI-EXPRESS BUS FUNCTIONAL MODEL

Bus functional model (BFM) is a virtual circuit with the same functional behaviors which complies with the protocol. Bus functional model focuses on the behaviors and the algorithms of internal functions. This virtual model can become the prototype of a design during the early design stage. Bus functional model can be constructed with a single language or multiple languages depending on flexibility or efficiency.

### 3.1 Layered BFM

All PCI/PCI-X and PCI-Express BFMs are formed by several layers, and each layer has its own input and output interfaces for connecting to other layers. The lowest layer is developed to collect bit level data, convert the data to instruction level format, and pass it to a higher layer. The highest layer is responsible of creating transaction and making appropriate responses for requests from other bus functional models.

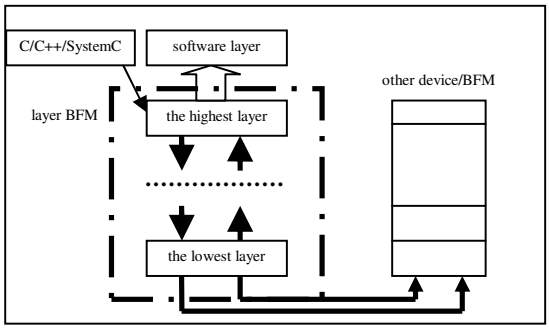

Fig 1 layered BFM

### 3.2 API for Layered BFM

Each layer is independent from others. User can control or send commands to each layer directly via APIs. All layers provide APIs from basic operation for individual layer to complex functions for multiple layers. Error injection is an important issue in a bus functional model because it can

help designers test the robustness of their design. PCI-Express BFM also has various error injection related functions. The advantage of separated layer architecture is to help engineers test their design layer by layer rather than verify the whole circuit at the same time. By applying these APIs, designers can design and verify the circuit layer by layer.

## 4. VERIFICATION FRAMEWORK

This verification system has several parts including:
1. PCI/PCI-X and PCI-Express Bus Functional Model
2. Assertion and bus protocol checker
3. Firmware level component
4. Super monitor
5. Compliance and functional test set

The bus functional model has been introduced in the previous section.

### 4.1 Assertion and bus protocol checker

All bus functional models contain various embedded assertion checkers. These assertion checkers are built by TestWizard and triggered by specific events. Each assertion plays a role as an invisible monitor that will check the system or a block of circuit automatically when one or some events trigger this assertion. The purpose of each event is defined by the designer. For example, an event may represent a signal change on a one-bit line, a system command or a set of commands that interact with other components in the system for more complex procedure. If the design under test is a PCI-Express add-in card, the functional monitor will be triggered at the beginning when electrical signals are received. Then functional monitor judges that current state belongs to symbol level and wakes up corresponding assertion checker. With more and more electrical signals collected, functional monitor will identify whether the captured bits can form a data link layer packet (DLLP) header and then wake up another set of assertion checker to verify all timing constraints and data integrity. A packet will be tested form different angles automatically.

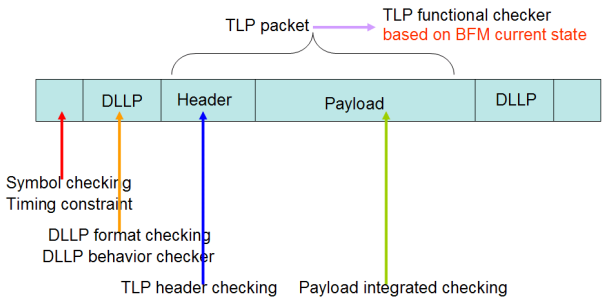

Fig 2 Assertion check procedure

### 4.2 Firmware Level Component

In the real world, at power up time, the system does not know the number of devices in it, and all devices that hook on PCI, PCI-X and PCI-Express bus are needed to set bus, function, and device numbers as their IDs in this system in order to work correctly. System also needs to know the memory requirement of each device, and then system will assign a start address to a device and announce all memory mapping results to a public block. This process is called enumeration. After enumeration, each device can read the information of all mapping memory addresses, understand a whole system configuration and knows how much resource can be used. Enumeration procedure is important for system level verification, especially in mixed protocol environment. Verification system allows static configuration for each scenario or dynamic configuration change during runtime by applying enumeration program to announce new ID and resource allocation. Super monitor must cooperate with firmware to get system topology.

### 4.3 Super Monitor

Super monitor can trace a complete transaction procedure and record latency of every link. This verification system is a mixed protocol environment, and a transaction may be split, merged, or partly split and merged. The super monitor contains two main blocks. The first block is a set of link monitors for each bus, which are used to capture bus signals. The second block is the transaction log database for saving all truncation information. Super monitor will start to extract all necessary parameters when it detects a transaction initiated by any device form link monitor, and classifies these data into specific database for user to access.

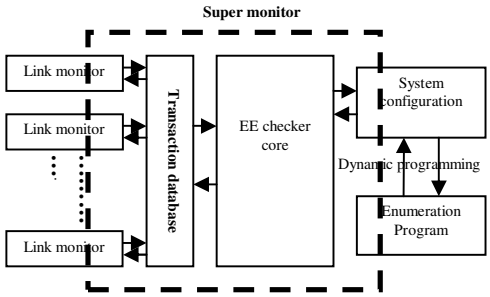

Fig 3 Super monitor

### 4.4 Compliance Test

The functional coverage of PCI/PCI-X and PCI-Express is based on PCI SIG's checklists. PCI SIG defines very detailed assertion list for every component, and test procedure for five parts including BIOS level, transaction

level, link state level, configuration space level, and physical level. The detailed information is listed in reference [7] [8] [9] [10] [11] [12]. We have developed more than 200 test scenarios in the verification system and almost all of these can test any kind of topology. All of these scenarios are capable of self-checking. Self-checking is an important characteristic of this system. In order to make all test scenarios to achieve this goal we classify all test cases to two different parts: one is called DUT-0 and the other is DUT-1. DUT-0 scenarios are scenarios that require no extra integration between verification system and DUT itself, and no information should be gotten beyond the normal protocol behavior. Any directly read or write command belongs to this class. DUT-1 scenarios are the scenarios that need to be integrated more closely to the verification system.

## 5. EXPERIMENTAL RESULT

We provide two template topologies in the verification system. One contains a Root Complex and an Endpoint. The other contains a Root Complex, an endpoint and a switch. We also inject some errors into each BFM in order to trigger some assertions. There are more than 200 scenarios created to test each topology. Scenarios have two kinds of types. One kind of scenarios are applied when device negotiates to other devices, the others are applied after a device has finished the negotiation process.

| | scenario | Pass | Fail | Checked | Asserted |
|---|---|---|---|---|---|
| EP-negotiation | 115 | 71 | 8 | 228 (53.1%) | 37 |
| EP | 49 | 18 | 8 | 126 (29.9%) | 49 |
| EP Total | 164 | 89 | 16 | 354 (83%) | 86 |
| SW-negotiation | 149 | 77 | 45 | 229 (53.1%) | 27 |
| SW | 52 | 21 | 8 | 137 (31.9%) | 49 |
| SW total | 201 | 98 | 53 | 366 (84%) | 76 |

Table 1 Experimental result

The assertions checked by our test case covered 84% items defined in [7] [8] [9] [11]. Some of assertion rules are non-testable. Some are design-dependent and are difficult to verify by observing bus activities. Some of test cases are hard to check automatically, and manual inspection is required.

## 6. CONCLUSION AND FUTURE WORK

An assertion-based verification environment for PCI/PCI-X and PCI-Express is built. This verification system provides all bus functional models and tools including enumeration software, bus protocol monitor, assertion-rule checker, and super monitor. All of these components cooperate with each other to form a whole compliance test suit. Every model or tool based is on single language to reduce the learning time and to keep the compatible of existing RTL design. The assertion-based checker can help designer to find their bugs and narrow down the scope very efficiently. By developing the enumeration program, the verification environment can support system level operation. By super monitor, high level cross protocol verification can be achieved. With this environment, DUT can be fully tested by interacting with other devices in more complex situations. This is useful to verify the robustness and potential drawbacks of the protocol.

In order to handle more and more complicated hardware and software design, current verification trend is to integrate all components in a virtual platform. A generic virtual system that supports multi-protocol and is capable of communicating with device's driver software layer will undoubtedly become prevalent in the future.

## 7. REFERENCES


[1] Ali Sayinta, Gorkem Canverdi, Marc Pauwels, Amer Alshawa, Wim Dehaene "A Mixed Abstraction Level Co-Simulation Case Study Using SystemC for System on Chip Verification", Design, Automation and Test in Europe Conference and Exhibition 2003 IEEE

[2] Monte Becker, "Faster Verilog Simulations Using A Cycle Based Programming Methodology", Verilog Consulting Services IEEE 1996.

[3] Rohit Jindal, Kshitiz Jain, "Verification of Transaction-Level SystemC models using RTL Testbenches", First ACM and IEEE Internal Conference on Formal Method and Models for Co-Design (MEMOCODE'03) IEEE 2003

[4] Abhik Roychoudhuny, Tulika Mitra, S.R. Karri, "Using formal techniques to Debug the AMBA System-on-Chip Bus Protocol", Design, Automation and Test in Europe Conference and Exhibition (DATE'03) IEEE 2003

[5] Emil Dumitrescu, Dominique Borrione, "Symbolic Simulatio as a Simplifying Strategy for SoC Verification with symbolic Model Checking", IEEE Internal Workshop on System-on-Chip for Real-Time Application IEEE 2003

[6] Silvia Brini, Doha Benjelloun, Fabien Castanier, "A Flexible Virtual Platform for Computational and Communication Architecture Exploration of DMT VDSL Modems", Design, Automation and Test in Europe Conference and Exhibition (DATE'03) IEEE 2003

[7] PCI-SIG Base_1.0a_PCI_Express_Endpoint_Checklist_1.0

[8]PCISIG Base_1.0a_PCI_Express_Root_Complex_Checklist_1.0

[9] PCI-SIG Base_1.0a_PCI_Express_Switch_Checklist_1.0

[10] PCI_Express_Test_Spec_Config_Space_0.9_26Aug03

[11] PCI-SIG PCI_Express_Test_Spec_PHY_0.9_26Aug03

[12] PCI-SIG PHY Interface for the PCI Express Architecture