A Functional Verification Environment for Advanced Switching Architecture

Min-An Song, Ting-Chun Huang, and Sy-Yen Kuo

Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan
E-mail: dennis@lion.ee.ntu.edu.tw

Abstract

In this paper, Intra-chip design has long been the main research area for the past years, resulting in several folds of computing power increase per year. In contrast, the speed of cross-chip connection, or even cross-system connection, has not been enhanced at the same rate, and gradually became the bottleneck of current computing systems. PCI, PCI-X, and PCI-Express epitomize the evolvement of our peripheral connecting strategy. By radical change in architecture, we successfully increase throughput of connection facilities in one autonomous system. We propose a architecture for verifying Advanced Switch hardware. Also successfully verify the compliance of the design under test's features to the given protocol.

1. Introduction

With the evolvement of CAD (Computer Aided Design) tools, hardware designers are getting more and more facilitated to handle complex designs. Though most of the processes have been automated, functional descriptions, however, are not possible and would never be possible to be automated since we always need a way to describe how our hardware works at the beginning of designing process. Today in circuit design, whether our design functions as we desire is much more concerned about than is whether we correctly map the described functions into layout. In other words, functional verification [1-5] has become the challenge of recent IC design.

From PCI, PCI-X, to PCI Express, the increasing on-board communication speed can never meet the exploding need for fast transferring of data. Especially in the next generation of computing that combines the power of distributed autonomous system, fast and reliable data communication protocol is indispensable. To meet the trend of fast communication, Advanced Switching is proposed as a solution. As well as high performance, Advanced Switching standouts out in its ability to tunneling packets of the protocol without modifying its content. This ability of tunneling packets eliminated the need of innumerable inter-protocol conversion through a transmitting path. Advanced Switching also supports Traffic class and Virtual Channel. Virtual Channels are logically independent data flows over a common physical channel. Advanced Switching also provides many other mechanisms such as power management, and these enhancements in AS protocol greatly raise the complexity of the verification.

The motivation for our implementing the verification environment at behavior level is to accelerate simulation. Here, we introduce Avery Design System's TestWizard, provided as Verilog system tasks that can manipulate these data structures during simulation. With the help of TestWizard, the verification environment is able to provide both speed and easiness for simulation. The proposed verification environment PCI-Xactor is a complete verification environment for PCI, PCI-X, PCI Express, and Advanced Switching, the PCI-Xactor provides a flexible verification environment.

This paper starts with a brief review of Advanced Switching. Then, the verification models and implementation details are put in Section 3. Section 4 Both valid and invalid behavior, convenient way to connect, and several mechanisms for testing flexibility are summarized here. Besides Advanced Switching verification environment, these elements are indispensable in any convenient and effective verification IP and Bus Functional Models. Finally, I conclude with the core elements necessary for a practical verification environment.

2. Advanced Switching

Advanced Switching provides an ecosystem that facilitates the communication of different autonomous systems. In Advanced Switching networks, packets of any protocol can be tunneled to its destination by AS packets. It delivers far superior speed and more capabilities, including hot add/remove, multicast, redundant pathway, and congestion management. Fig. 1 illustrates an example of AS fabrics. Practically, every protocol available today can plug on AS fabric through an as fabric, comprising an ecosystem of a variety of autonomous systems of different protocols. AS fabric tunneling begins at the source bridge and terminates at the destination bridge. AS not only provides an ecosystem that holds different fauna but enable them to interact with each other, it successes in converging communication and computing.

This research was supported by the Ministry of Economics, Taiwan under contract number 94-EC-17-A-08-S1-0006

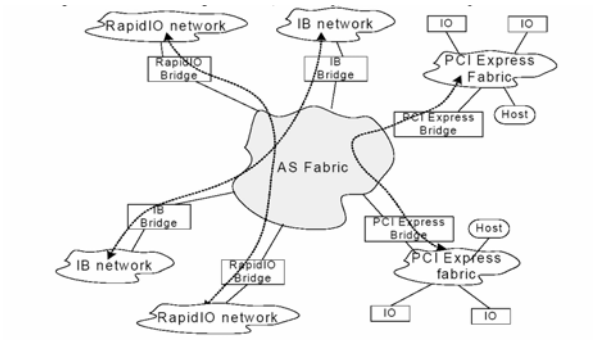

Fig. 1 Advanced Switching Topology.

Advanced Switching inherits Physical Layer and Data Link Layer from PCI Express. Fig. 2 illustrates the relationship of each layer between PCI Express and Advanced Switching.

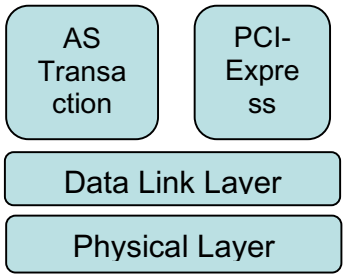

Fig. 2 Relationship in Layers between AS and PCI Express.

Advanced Switching is a source routing protocol: all of the routing information is specified when the packet is generated. The routing information is kept in the aperture of each endpoint as Spanning Tree Table Capability Structure through a process called enumeration at the beginning of the system up. Only after spanning tree generation finishes can the AS fabric transmits packets normally.

3. Verification Architecture

In this section, The PCI-Xactor system provides a set of Bus Functional Models (BFMs) consisting of an Endpoint, a Switch, a single link Monitor, and a system level monitor called Super Monitor. Typically a link monitor listens to a link in the system. The Super Monitor is used to perform end to end verification in systems with multiple links. The AS BFMs are modeled entirely in Verilog and utilize TestWizard for Verilog high level testbench features. The Core BFM provides the core interface to a PCI Express link, including the DLL and PHY layer interfaces, scrambler and SERDES block. The device model is a complete Endpoint BFM, used or modified by the user for their purposes. The link monitor is used to passively monitor a PCI Express link for protocol violations and to store information in the transaction database if desired. The packet record provides the user with a ready made packet structure to be used for PCI Express IO. The transaction database provides us with further extension on functional coverage analysis.

The Core BFM provides an interface to the user for each of the layers in the protocol stack, Transaction, Data Link, and Physical. To ease the job that the user must perform, most of the lower layer tasks are automated, such as link negotiation and ACK/NAK behavior. While the user will primarily interact with the BFM at the transaction layer, interfaces are provided at all layers.

3.1 PCI Express Model

PCI Express Root Complex is composed of parallel PCI Express devices, namely Root Ports. Besides these devices, Root Complex has its internal machine to manipulate cross port communication and to additional logic for Root Complex functioning. The structure of a 6-port Root Complex is depicted as Fig. 3.

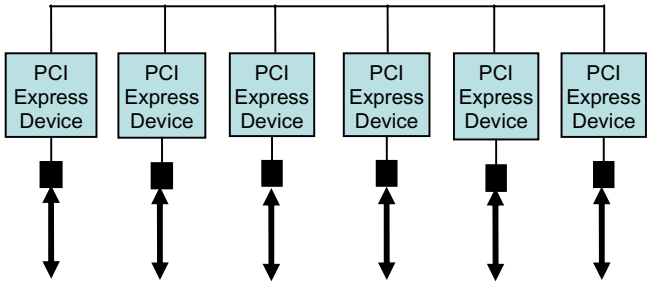

Fig. 3. PCI Express Root Complex Model.

PCI Express Switch is constructed with upstream port and parallel downstream ports. The upstream port is a PCI Express device, and the downstream ports function as a Root Complex. Please refer to Fig. 4 for a 1-to-6 PCI Express Switch.

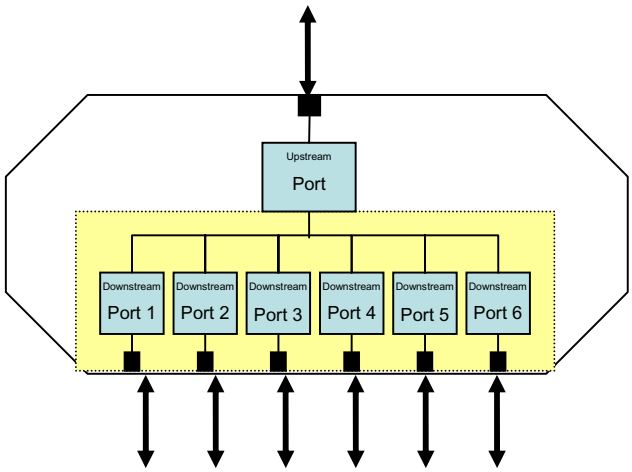

Fig. 4. PCI Express Switch Model.

3.2 Advanced Switching Model

PCI Express Root Complex has no counterpart in Advanced Switching protocol since all of the components linked by AS are autonomous. Interestingly, AS Switches

resemble PCI Express Root Complex more than PCI Express Switches in its architecture. Without upstream and downstream distinguishing among ports, the switch of AS packets is decided by the incoming port and shifts, specified in AS packet.

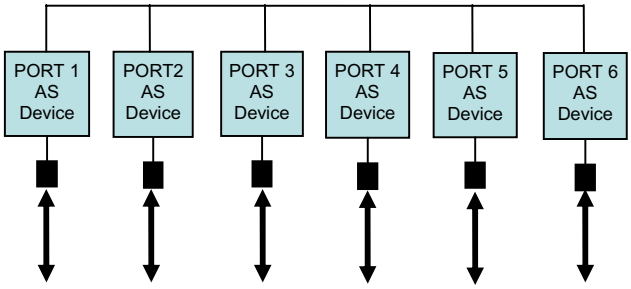

Fig. 5. AS Switch Model.

The architecture of the switch model is showed in Fig. 5. The 6-port switch is composed of dev0 through dev5..

Host Switch and I/O Switch provides transparency between Advanced Switching and PCI Express. With collaboration of Host Switch and I/O Switch, an PCI Express tree can span over different part of the AS connection.

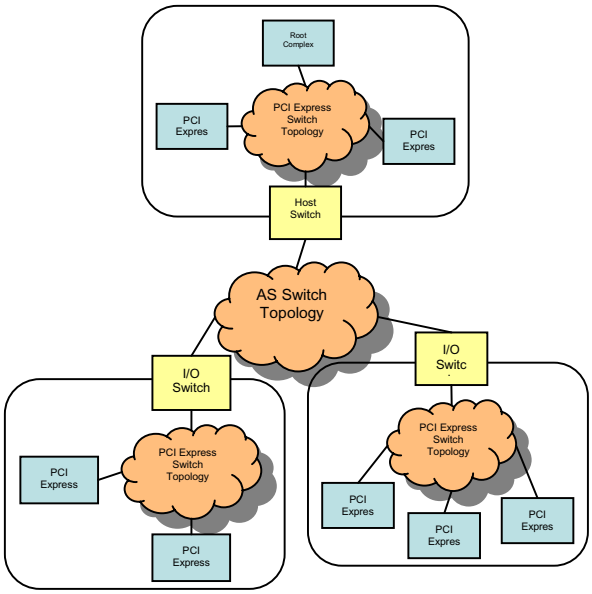

Fig. 6 Mixed AS and PCI Express Topology.

The upstream port of a Host Switch is a PCI Express device and the downstream ports of a Host Switch are AS devices, as in Fig. 7; in the contrary, the upstream port of an I/O Switch is an AS device and the downstream ports of an I/O Switch are AS devices, as in Fig. 8.

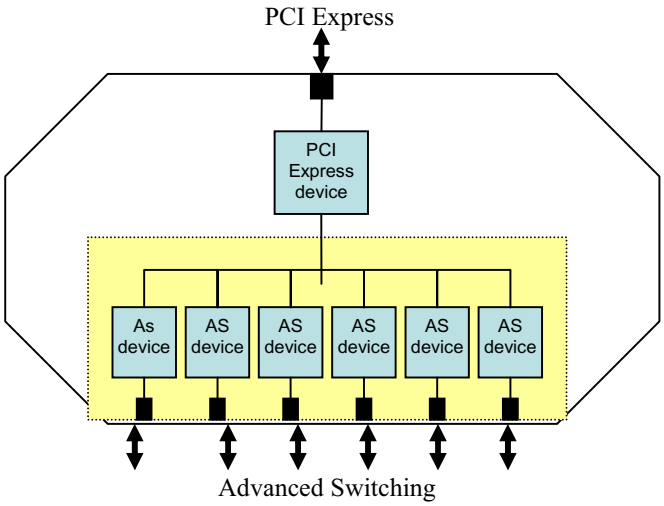

Fig. 7. AS Host Switch Model.

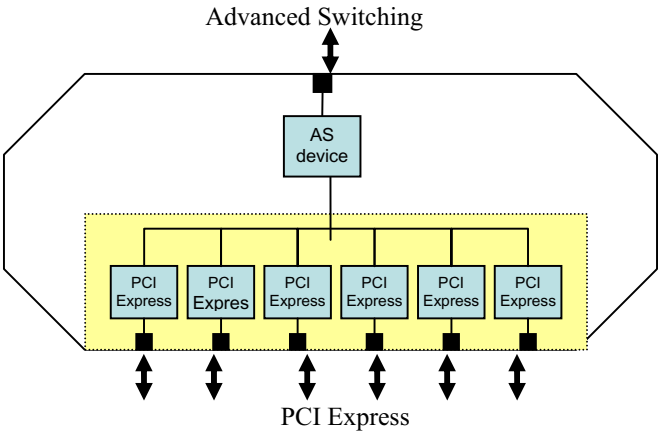

Fig. 8. AS I/O Switch Model.

The link monitor attaches to both directions of the dataflow in one link. It monitors the flow of packets in both directions and provides protocol checking for that link. For single links the link monitor can be used to perform protocol checks, gather coverage information about the traffic on a link, and to save the tracker file that contains textual information about the traffic on the link.

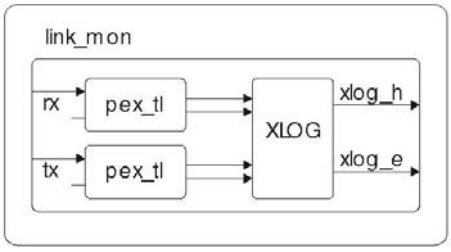

Fig. 9. AS Link Monitor Model.

The Link monitor xlog outputs are provided to allow the user access to the individual transaction database for this link. As each transaction layer decodes all the

packets from links, protocol checks are applied spontaneously by embedding all of the checks in the BFM. Link monitor has the same checks as does a device's transaction layer, and thus link monitor is used as a passive protocol checker through the link on which no other checks are applied.

4. Bus Functional Model

From the perspective of function, while real design only works correctly, verification environment must support both correct and wrong behavior. The correct behavior tests DUT's active behavior, whether the packets originating from DUT violates the transmission rules; the wrong behavior test DUT's passive handling problematic packets and it resistance to error. TestWizard is a test development system designed by Avery Design Systems to simplify transaction-based verification. All of the functions provided in TestWizard are implemented, and can be coded with the designs in Verilog directly. TestWizard Record extends Verilog to support C structures, providing a convenient way to handle complex transaction-based verification.

Here, I introduce the record tasks that would be applied to implement the packets and capabilities of AS protocols extensively. Record enables us to define complex data structures in Verilog HDL. TestWizard supports defining and creating of records, and accessing of their fields directly in Verilog machines. Just as we define a structure, allocate memory space, access its fields, and finally free the memory space in C language, in Verilog we handle data structure as records in the same way. A record type and its fields are first declared with tasks $tb_vrecord (for Verilog-type variables) and $tb_crecord(for C-type variables). Then, record variable are instanced with $tb_var_record. $tb_record_new allocates necessary memory space for the record.

To access a field of the record, we use $tb_record_field for reading and $tb_save_record_field for saving. VCK supports two types of records: Verilog-type records which use the 4-value set, and C-type records which use the binary value set. Finally, when a record is of no need in the future, we free the memory space with $tb_record_free. Table 1 lists TestWizard Record tasks.

Table 1. Test Wizard Records.

$tb_vrecord	$tb_save_record_field
$tb_crecord	$tb_record_compare
$tb_var_record	$tb_record_free
$tb_record_new	$tb_record_copy
$tb_record_field	

Error messages are currently coded in the design. When receiving an invalid packet, errors would be notified once the packet is found to be error. We also check the validity of outgoing packets and these checks are for developing and debugging of our verification environment. Different test segments are represented with different integers. Thus, tests can be applied with most flexibility; we can easily apply tests to each component simply by assigning the test number into the desired bucket at some simulation time.

The design of a verification environment must support DUT integration to provide three layers separately. By providing our verification components as lego, design without Transaction Layer and Data Link layer can easily integrates with our components, providing verification in early stage of designing.

5. Conclusion

For these testing, we make use of TestWizard to provide a direct and flexible mechanism. After all, Verilog is a HDL designed for hardware simulation. It lacks the flexibility and convenience of C. TestWizard is of the same functions as is these augmentative languages. Rather than a totally different language, TestWizard provides these functions in the form of Verilog system tasks. TestWizard provides a easier way to this enhancement since the engineers need not to learn a new language. In convenience, our way to connect DUT and verification components is wire link in Verilog. At the connection file, users can easily set up parameters for verification components (flexibility). Besides, we can change our topology as we desire easily since all of the components are available.

References

[1] F. Sforza, L. Battu, M. Brunelli, A. Castelnuovo, M. Magnaghi, *"A 'design for verification' methodology"*, Quality Electronic Design, International Symposium on, pp. 50-55, 2001.

[2] Bernd Stöhr, Michael Simmons, Joachim Geishauser, *"FlexBench: Reuse of Verification IP to increase Productivity"*, IEEE Proceedings of Design, Automation and Test in Europe Conference and Exhibition, 2002.

[3] T. Kuhn, T. Oppold, M. Edwards, Y. Kashai, M. Winterholer, W. Rosenstiel, *"A framework for object oriented hardware specification, verification, and synthesis"*, Design Automation Conference, 2001. Proceedings, pp. 413-418, 2001

[4] Albin, K, *"Nuts and bolts of core and SoC verification"*, Design Automation Conference, 2001. Proceedings, pp. 249-252, 2001

[5] F. Pogodalla et al., *"Fast Prototyping: a System Design Flow Applied to a Compex System-on-Chip Multiprocessor Design"*, DAC 1999 proceedings, pp.420-424.