System-Level Design of IEEE1394 Bus Segment Bridge

Hirofumi Yamamoto, Keishi Chikamura, Atsuhito Shigiya, Kosuke Tsujino,
Tomonori Izumi, Takao Onoye, Yukihiro Nakamura

Dept. Communications and Computer Eng., Kyoto University
Kyoto, 606-8501 Japan
Phone: +81.75.753.4804, Fax: +81.75.753.4804
E-mail : {hiro, keishi, ashigiya, tsujino}@easter.kuee.kyoto-u.ac.jp
{izumi, onoye, nakamura}@kuee.kyoto-u.ac.jp

ABSTRACT

A system simulation environment is constructed dedicatedly for IEEE1394 high-speed digital communication. In this environment, various network transactions inherent in communication systems are taken into account for system simulation, which is indispensable to enable IP (Intellectual Property)-based design of the systems. By using the proposed environment, system-level design of IEEE1394 link layer controller and bus segment bridge is achieved with great ability of network transactions as well as connectivities with physical layer chips. Functionalities of the designed bus segment bridge has been verified according to its FPGA implementation.

Categories and Subject Descriptors

B.6.3 [**LOGIC DESIGN**]: Design Aids; C.2.3 [**COMPUTER-COMMUNICATION NETWORKS**]: Network Operations

General Terms

Design

Keywords

HW/SW Co-simulation, C/C++, Verilog-HDL, PLI, IEEE1394, Bus Bridge

1. INTRODUCTION

With the recent advances in semiconductor technology, the number of logic gates in a single chip is growing rapidly where functionalities implemented in the chip are becoming complicated. In this situation, reuse of IP (Intellectual Property) is necessary to provide such system-on-a-chip LSIs timely to the market.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ISSS'02, October 2–4, 2002, Kyoto, Japan.
Copyright 2002 ACM 1-58113-576-9/02/0010 ...$5.00.

On the other hand, with the advent of information technology, network interface is included in SoC designs as a key IP module. Moreover, there are various kinds of network protocols in terms of transfer speeds and functionalities, and hence IP-based network interface design is indispensable for practical applications.

Network devices are demanded to provide many functions dealing with various protocols, and such functions are implemented as hardware or software. To reduce design complexity, these devices are designed through such a layered model that all functions are partitioned into layers, each of which is implemented independently. For example in IEEE1394[1], five protocol layers (physical layer, link layer, transaction layer, serial bus management layer, and application layer) are defined to simplify the implementation of devices. Among these layers, the physical layer and the link layer are generally realized in hardware, while other layers are realized in software.

Based on the harmonization of these layers, network devices can communicate each other, and thus hardware and software co-simulation environment must be necessarily employed so as to check whether working correctly. A number of approaches has been attempted to facilitate hardware and software co-simulation[2][3][4][5], which mainly aims at verification of interoperability among layers in a single device. However, there are few established ways of hardware and software co-simulation to cope with various kinds of transactions, which are inherent in network interfaces such as dynamic insertion and deletion of devices during they are working.

In this paper, a system simulation environment is constructed dedicatedly for IEEE1394 high-speed digital communication based on discrete event simulator and Verilog-PLI (Programming Language Interface), which is the interface between Verilog simulator and C/C++ function. By using the proposed environment, system-level design of IEEE-1394 link layer controller and bus segment bridge is achieved with great ability of network transactions as well as connectivities with physical layer chips. Functionalities of the designed bus segment bridge has been verified according to its FPGA implementation.

2. IEEE1394 NETWORK

2.1 Overview

IEEE1394 aims at the fields of PC peripherals and electric household appliances, which handles broadband data streams such as moving pictures and a set of audio streams. The primary features of IEEE1394 are summarized as follows:

1. **Scalable performance** Transfer speeds of 100, 200, and 400 Mbps are supported.
2. **Hot insertion and removal** Devices can be attached or removed from the bus dynamically without powering the system down.
3. **Plug and Play** Each time a device is attached or detached the bus is reenumerated. Nodes on the bus are to a large degree self-configuring, and configuration does not require intervention from a host system.
4. **Support for two types of transactions** Isochronous and asynchronous transfers are supported.
5. **Peer-to-Peer transfer support** Serial bus devices have an ability to perform transactions between themselves, without the intervention of a host CPU.
6. **Cable power** Power available from the bus can be either sourced or sinked by a given node.

Layered model of IEEE1394 is illustrated in Fig. 1, where the link layer and the physical layer (hereafter "the *link*" and "the PHY", respectively) are generally implemented in hardware and other layers are done by software. Data communication is executed through the transaction layer with the *link* and the PHY, and bus management, such as bus cycle and bus resource assignment, is done by the serial bus management layer.

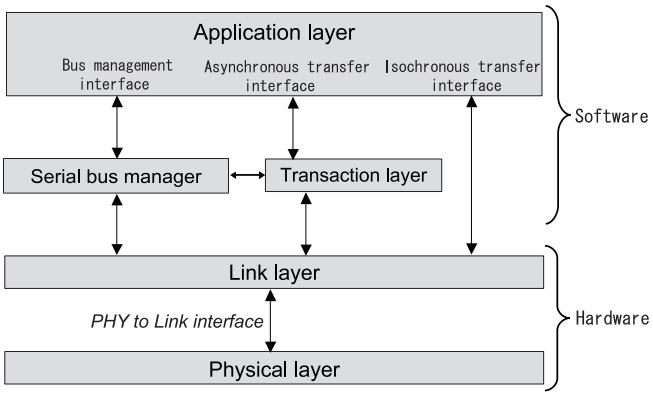

Figure 1: Layered hardware and software model of IEEE1394.

2.2 Functions of link layer

The *link* is placed between the transaction layer and the PHY, which receives data packets and status information from the PHY via the "PHY-to-link" interface. IEEE1394-1995[6] and P1394a[7] specify the PHY-to-link interface in full detail as is shown in Fig. 2.

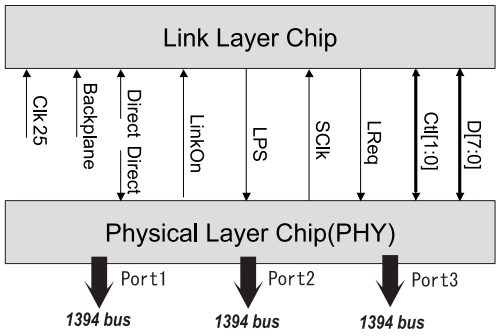

Figure 2: IEEE1394 PHY-to-link interface.

The interface is used bidirectionally by both the *link* and the PHY; i.e. the *link* initiates transactions by sending requests to the PHY, or the PHY forwards packets received from 1394 buses to the *link*.

A packet data is delivered via the data lines(D[7:0]), where the number of data lines used depends on the transfer speed such that D[1:0], D[3:0], and D[7:0], are for 100, 200, and 400Mbps, respectively. Control lines(Ctl[1:0]) notify the state of a layer when being driven by the *link* or the PHY. LReq line is used by the *link* to initiate a request for sending packets and to request to read local PHY registers. SClk line is used by the PHY to supply 49.152MHz clock for the *link*.

On the other hand, the interface between the *link* and the transaction layer is not fully specified, since it depends on the implementation.

2.3 IEEE1394.1 Bus Bridge

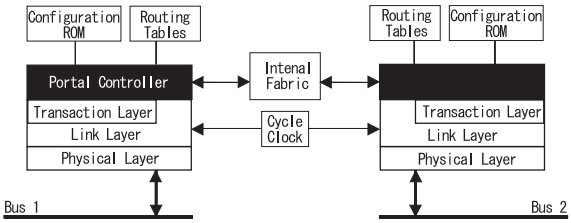

Figure 3: Bridge model of IEEE1394.1

As shown in Fig. 3, a bus bridge is composed of two bridge portals which are connected to two different IEEE-1394 buses. Two bridge portals communicate with each other through the internal fabric, and each with its own address space as well as the node_ID of connected bus. The cycle clock provides both portals with the same clock in order to synchronize both buses.

In this manner, an IEEE1394 network can be constructed by a set of buses. Specifically, there may be several bus portals on a bus, among these portals one portal is selected as the alpha portal, and among alpha portals on the network one portal is selected as the prime portal.

The main facilities of a bus bridge are;

- Assigning virtual_IDs which are used for a remote transaction.
- Routing asynchronous and isochronous packets.

- Executing net update process when a network topology is changed.

- Adjusting their own bus cycle to net cycle master, which is the cycle master belonging to the same bus as prime portal, for routing isochronous packets.

3. HARDWARE AND SOFTWARE CO-SIMULATION ENVIRONMENT

3.1 Overview

In the design of the *link*, the interoperability in the layered network model must be considered and be verified efficiently. For this purpose, hardware and software co-simulation environment is required since the *link* and the PHY are implemented by hardware while other layers are generally implemented as software. Moreover, since dynamic insertion or removal of nodes occurs without powering the system down, the number of conditions on which the verification should be carried out is so large that the simulation environment should also be capable of handling these conditions successfully.

However, conventional hardware and software co-simulation schemes can hardly treat these layered network models with dynamic operations sophisticatedly. Thus we have developed a novel hardware and software co-simulation environment, in which each layer of the network model can be described whether in hardware or in software.

This nature contributes simplification of the functional verification. For example, let us consider the case of designing the *link* LSI by using hardware model of the PHY through the conventional design scheme. The network transaction of the PHY is so complicated that HDL description with limited ability to express transactions is not suitable for. On the contrary, our simulation environment allows C++ description of the PHY, instead of its HDL description, with the use of discrete event simulator, which has great ability to express complicated network transactions. An outline of our system simulation environment is shown in Fig. 3. The discrete event simulator uses so-called Verilog-PLI (Programmable Language Interface)[8] between C and Verilog simulator, which enables verification of all layers based on the IEEE1394 layered model.

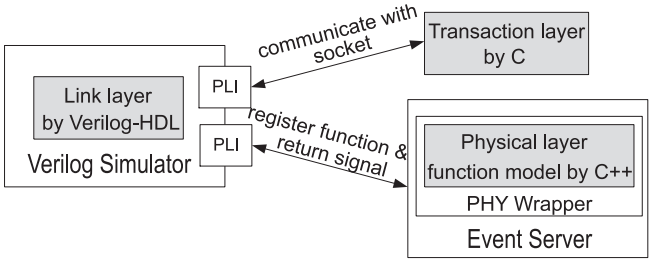

Figure 3: IEEE1394 system simulation environment.

3.2 Discrete event simulation and its interface

The discrete event simulator treats events of the function model of a layer in function unit basis, e.g. bus reset detection, data transmission, data decoding, and state control of the PHY. Each function unit corresponds to an "object" of C++. In other words, the function model consists of a set of function units. Mechanism how our discrete event simulator cooperates with Verilog simulator is depicted in Fig. 4. Each object passes events with time stamps to the event server. The event server sorts these events in order of time stamps, and registers the event to the Verilog simulator via Verilog-PLI.

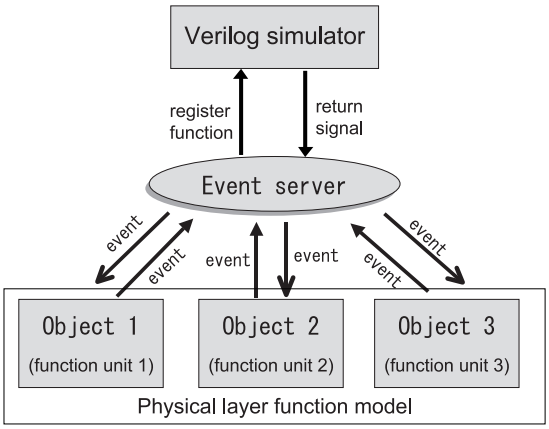

Figure 4: Event server and objects.

If the time to execute the event comes during the Verilog simulation, the Verilog simulator returns the signal to the event server via Verilog-PLI. Then the event server indicates execution of the event to the object.

As a result, very flexible system simulation is enabled. In this system, each component part of a network device, no matter which is written in HDL or C, is exchangeable freely. Hence, we can use the function model by C++ instead of the HDL description if the behavior of a component is too complicated to be expressed by the HDL.

4. SYSTEM LEVEL DESIGN OF IEEE1394 DEVICES

4.1 Link Layer Controller Design

4.1.1 Architecture

For the co-simulation of the *link* described in Verilog, we described the function model of the PHY by C++ and the function of the transaction layer by C.

The function model of the PHY is based on lots of state machines as described in IEEE1394-1995. In C++ description of the function model, each function unit is mapped to an object. It is also required that a mechanism of transforming actions of the PHY function model to the signals of register transfer level (RTL). The "PHY wrapper" in C++ serves this role, which can pass/receive RTL signals to/from Verilog simulator through PLI.

On the other hand, the transaction layer relays any transactions except for the isochronous transactions. This function is implemented in C and communicates to the *link* via a socket, as is often employed in a software simulation of network model. While in the conventional design methodologies, software and hardware parts communicate each other

through device driver which is included in an operating system. By using the socket interface, the co-simulation between the transaction layer and the hardware layers can be successfully executed.

As depicted in Fig. 5, the link layer controller consists of a set of blocks. It employs six data buffers (shadowed in the figure), each of which can be distinguished in its role, receive or send, asynchronous or isochronous, and so on.

In addition, the "State control" block have a state machine which send signals to other modules, and according to these signals, for example, the "Link request" module send a request message to the PHY through the LReq line in proper timing. The "Transmit data control" block decides which packet to give priority, for example, an *acknowledge* packet is the most prior and a *cycle start* packet is second.

When a packet data is received via the data lines, it is forwarded to the "Data decode" block and the "CRC error check" block. The data decode block scans received packet and checks data type. The CRC error check block checks any errors of the packet, and if the CRC error at the header section or the data section of the packet is detected, the link layer controller initiates an acknowledge packet such as "ACK_HEADER_ERROR" or "ACK_DATA_ERROR."

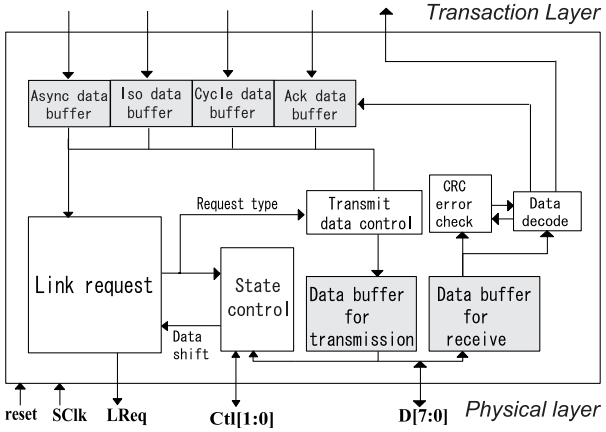

Figure 5: Block diagram of link layer controller chip.

4.1.2 Implementation Results

With the proposed system simulation environment, the link layer controller is designed in Verilog HDL. As summarized in Fig. 6 , a number of dynamic/static network transactions of the PHY are tested to confirm the interoperability of the designed module. For example, simulation of dynamic node insertion is executed in the following way: When a node is inserted to an active IEEE1394 network while other nodes in the network communicate some packets, a bus reset occurs and all nodes send self-ID packets to each other for updating their own configuration. Then, finally, all nodes resume general network transactions. In addition to this, simulations of such a dynamic node deletion or static topologies can be easily done in the same way. Furthermore, the function model of the PHY is effectively described in C++ by 2500 lines, while in the case of HDL description more than 13000 lines are needed.

Fig. 7 demonstrates the execution of the co-simulation, in which the PHY model reports dynamic operation of the

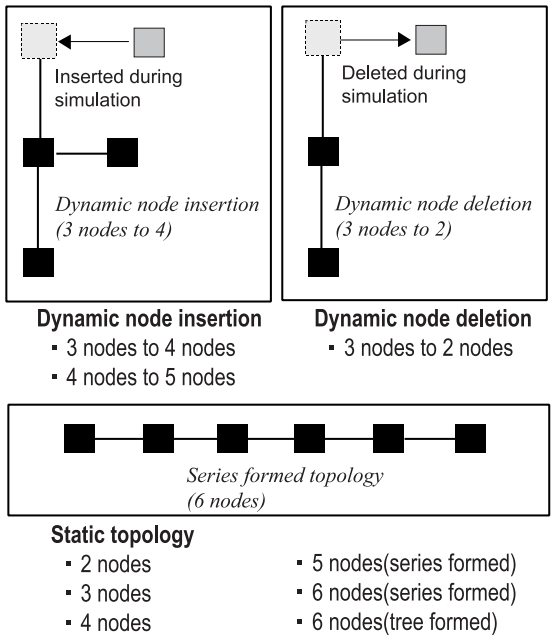

Figure 6: Example of dynamic/static network transactions of PHY.

network and Verilog simulation waveforms of the *link* at the time of the operation are depicted.

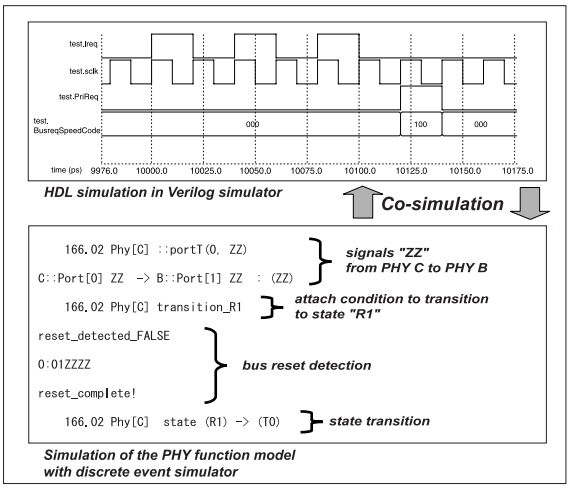

Figure 7: Screenshot of co-simulation.

The link layer controller has been synthesized to XC4000-XLA high performance FPGA by using Verilog-XL and Xilinx Foundation. Table 1 summarizes main features of the link layer controller FPGA.

4.2 Bus Segment Bridge Design

4.2.1 Architecture

Utilizing the proposed environment as a different aspect to construct a new subsystem by IEEE1394, the design of "IEEE1394 bus segment bridge" has been attempted. Apart

Table 1: FPGA implementation results.

Critical path	19.245ns
Clock period	20.345ns(f = 49.152MHz)
Gates	19220
Technology	0.35 μm CMOS
Source Voltage	3.3V
Device	XILINX XC4000XLA

from IEEE1394.1 bus bridges, our newly proposed IEEE1394 bus segment bridge enables legacy devices to use network resources effectively and avoid bus reset issue by separating a bus into several segments. Under this environment, a legacy device can communicate with nodes on remote segments without any updates. In addition, this bus segment bridge can be embedded in and works properly in IEEE1394.1 network.

The model of a bus segment bridge resembles bus bridge model of IEEE1394.1. It is composed of two portals, and each portal has an own routing table and configuration ROM. However, as compared with that of IEEE1394.1, the configuration ROM is much smaller since the number of nodes which a bus segment bridge must manage is up to 63, while up to 65,000 in the case of IEEE1394.1.

The main features of proposed bus segment bridge are;

- *Node_ID assignment.*
  Each bus segment bridge assigns virtual_IDs of remote segment nodes at bus configuration process.

- *Asynchronous and isochronous packet routing.*
  The routing of asynchronous packets is processed according to the algorithm of the IEEE1394.1 specification, except for the remote time-out.

The key issues of this architecture are how to enable above mentioned features in an ordinary IEEE1394 network, which are summarized in the following.

4.2.2 Node_ID assignment

Node_IDs can be assigned to legacy devices only at bus configuration, and the devices can not detect a change of a network topology in other bus states. The process of bus configuration is performed as is illustrated in Fig. 8, where the physical_IDs of nodes on the segment 1 and virtual_IDs of nodes on the segment 2 are shown in the squares. When bus reset occurs at the segment 1, then portal 1 will inform nodes on the segment 1 of existence of segment 2 nodes by using information from portal 2. In a bus configuration process of segment 1, p_1 generates self_ID packets to pretend that nodes on the segment 2 is connected to p_1 directly. Therefore, the nodes on the segment 1 recognize the nodes on the segment 2 as connected to the same bus, while they are separated into different segments by the bus segment bridge.

When a node is removed from or is connected to a segment, bus reset occurs and the nodes on the segment can detect a change of the bus topology. However, nodes on other segments can not detect it until bus reset occurs at their own segment. In the case of a node removal, a portal which receives packets to the removed node returns an error packet. Then, in the case of a node addition, portals

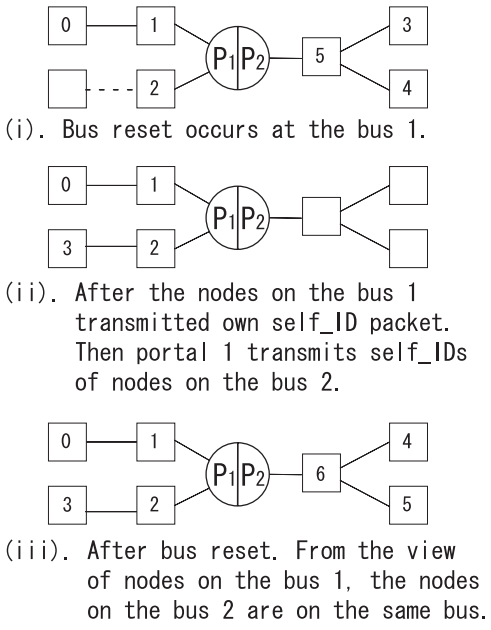

Figure 8: Bus reset process in IEEE1394 bus segment bridge environment

on other segments transmit bus reset signals to their own segments at the appropriate time to inform the change of the network topology. The time when a portal transmits bus reset signal is determined by referring to the influence on application over IEEE1394 bus. For example, a time when there are no isochronous transitions can be considered. According to this, the time until a change of topology is detected becomes longer, however, practically it is not a serious since the nodes that communicate infrequently with each other are tend to be connected to different segments.

4.2.3 Response time-out issue

As a legacy device can not set remote time-out for remote transaction, remote and local transaction must be finished in the same time-out period. The default time-out value of a legacy device is 100ms, however its value can be set up to 8s. The time required for our implementation to route asynchronous packets is verified in the next section.

4.2.4 Isochronous packet routing

A proposed bus segment bridge can not route isochronous packets by its destination_ID since they do not include destination_IDs but channel numbers. In an IEEE1394.1 environment, a node which transmits isochronous packets to remote bus builds path by a command packet defined in IEEE1394.1. Since the bus segment bridge intends that a legacy device can communicate without any changes, the same method can not be used in this case.

There are two methods for routing isochronous packets in proposed environment. One method is that, a bus segment bridge routes all isochronous packets that is not registered beforehand, and a node that transmits isochronous packets can register the path not to route own packets by the command packet. By this rule, a legacy device can transmit isochronous packets to remote segments, and also it is possible to use the network resource effectively. In the

other method, a bus segment bridge decides which channels are to be routed automatically. This method can be achieved by applying a special asynchronous packet which is utilized by applications for isochronous transactions, like AV/C command packets[9]. By this method, applications which can utilize isochronous transactions are restricted, however, none of updates is required for legacy devices.

4.2.5 Implementation Results

In order to verify functions of bus segment bridge, an evaluation board is implemented, which is composed of commercial physical layer LSI[10], a CPLD[11] for link layer controller, and a PCI interface. Overall organization of the board is illustrated in Fig. 9. In order to constitute an IEEE1394 bus segment bridge, two evaluation boards are equipped with PC. The transaction layer, serial bus manager, and portal controller are implemented as a device driver based on an open source 1394-OHCI for Linux operating system. DMA controller of the board reduces the host CPU load, and enables the host CPU to process portal's operations in real-time. The logic synthesis results of the link layer controller, DMA controller, and PCI interface are shown in Table 2, and Fig. 10 depicts the designed evaluation board.

Processing overhead from a physical layer to the other one through the implemented bus segment bridge can be estimated less than 100μs, and thus the bus segment bridge conforms to the default time-out value of a legacy device, i.e. 100ms.

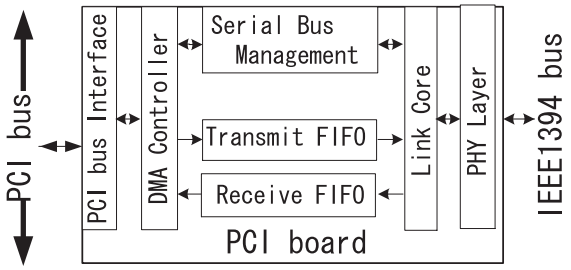

Figure 9: Organization of evaluation board

Table 2: Synthesis results

CPLD	APEX20K
Cells	13239(79%)
Memory	13KB
Clock	49.94MHz

Figure 10: Evaluation board

5. CONCOLUSION

This paper has described a novel concept of IEEE1394 hardware and software co-simulation, which is compliant to the layered network model, and the implementation of the link layer controller chip.

The proposed co-simulation scheme can contribute reduction of functional verification period, especially in the case of various static/dynamic operation are incurred in the verification condition such as networking devices.

To demonstrate the practicability of the proposed design system, system level design of link layer controller and bus segmenet bridge for IEEE1394 is attempted and experimentally implemented by using FPGA.

Development is continuing on much more sophisticated system simulation environment for networking devices such as precise simulation of transmission lines, etc.

6. REFERENCES

[1] Don Anderson, *Fire Wire System Architecture, 2nd ed.*, Addison Wesley Publishing Company, 1998.

[2] Gajski, D. D., Zhu, J. and Domer, R. *Hardware/Software Co-Design*, Kluwer Academic Publishers, 1997.

[3] Kurt Keutzer, *Hardware-Software Co-Design and ESDA*, In Proc. of the 31st DAC, 1994.

[4] J. Van Praet, D. Lanneer, W. Geurts, G. Goossens and H. De Man, *Modeling Hardware-specific Data-types for Simulation and Compilation in HW/SW Co-design*, In Proc. of SASIMI'96, 1996.

[5] M. Yasuda, K. Seo, H. Koizumi, B.Shackleford, F. Suzuki, *A top down Hardware/Software Co-Simulation Method for Embedded Systems Based Upon a Component Logical Bus Architecture*, In Proc. of ASP-DAC'98, 1998.

[6] IEEE *"1394-1995 IEEE standard for a high performance serial bus,"* IEEE, 1995.

[7] IEEE *"P1394a draft standard for a high performance serial bus,"* IEEE, 2000.

[8] Stuart Sutherland *The Verilog PLI Handbook*, Kluwer Academic Publishers, 1999.

[9] IEEE *"AV/C digital interface command set general speicification version 4.0,"* IEEE, 2000.

[10] Texas Instruments *"IEEE1394a-2000 three-port cable transceiver/arbiter Data sheet"* Texas Instruments, 2000.

[11] Altera *"APEX20K programmable logic device family Data sheet"* Altera, 2001.