Implementation and verification of PCI express interface in a SoC

Vinay Kumar Pamula\textsuperscript{a)} and Sai Raghavendra Mantripragada\textsuperscript{b)}

Department of ECE, University College of Engineering Kakinada, JNTUK, Kakinada 533001, India
\textsuperscript{a)} pamulavk@ieee.org
\textsuperscript{b)} raghavendra2059@gmail.com

Abstract: This paper describes the implementation and design of interface for peripheral component interconnect express (PCIe) interconnect and memory in a complex system on chips (SoC). PCIe bus traffic is made of a series of PCIe bus transactions. The direction of the data will be from initiator to completer (for write transaction) or vice-versa (for read transaction). The interface will read the command of the master and send corresponding response to the master. The major objective of the project is performance verification of SoC on a dedicated channel between PCIe end point and memory using performance models. We are using direct memory access (DMA) type of requests and bandwidth is measured at bottleneck for different PCIe generations, lane configurations and payloads. Bandwidth obtained is being compared with theoretical peak bandwidth calculated.

Keywords: PCIe, memory, SoC, DMA

Classification: Network

References

[1] N. Kamat, “IP testing for heterogeneous SOCs,” 2013 14th Int. Workshop on Microprocessor Test and Verification, Austin, TX, pp. 58–61, 2013. DOI:10.1109/MTV.2013.19

[2] S. Badhe, K. Kulkarni, and G. Gadre, “Accelerating functional verification of PCIe express endpoint by emulating host system using PCIe express core,” Computational Systems and Communications (ICCSC), 2014 First Int. Conf., Trivandrum, pp. 333–338, 2014. DOI:10.1109/COMPSC.2014.7032673

[3] Pci sig, PCI Express Base Specification Revision (3.0) [Online]. Available: https://pcisig.com (2010 Nov. 10).

[4] I. Sazzad and S. Kaiser, “Hardware implementation of PCI interface using Verilog & FPGA,” Int. Conf. Electrical, Electronics and Civil Engineering (ICEECE 2011).

[5] L. Rota, M. Caselle, S. Chilingaryan, A. Kopmann, and M. Weber, “A PCIe DMA architecture for multi-Gigabyte per second data transmission,” IEEE Trans. Nucl. Sci., vol. 62, no. 3, pp. 972–976, June 2015. DOI:10.1109/TNS.2015.2426877

[6] J. Lawley, Understanding Performance of PCI Express Systems [Online] (2014 Oct. 14).
1 Introduction

Over the past years, processor performance is going on improving, but the technological progress of system interconnects are not matching with them. Comparing with existing system interconnects such as Ethernet and InfiniBand, the PCI Express (PCIe) device has advantages such as higher protocol efficiency, low-power consumption per port, and low unit price [1]. To help the processor and interconnect have balanced performance, we propose a system interconnection device using PCIe. PCIe is high-speed point-to-point interconnection technology that comes from the typical PCI bus technology. It is used as the primary interface between processor and IO devices. It is offering high-speed data transfers over a physical link composed of multiple lanes that are scalable (from $\times 1$ to $\times 32$). Starting from the first generation (GEN 1) with 2.5 Giga Transfers per second (GT/s) speed per lane, the PCIe has been continually enhancing its performance and reached 16 GT/s per lane speed as its fourth generation [2]. Second generation (GEN 2) and third generation (GEN 3) of PCIe have speeds as 5 GT/s and 8 GT/s respectively.

2 Review on PCIe bus

PCIe employs point-to-point interconnects for communication between two devices as against its predecessor buses that used multi-drop parallel interconnect. A point-to-point interconnect implies limited electrical load on link allowing higher frequencies to be used for communication. Because of the serial interconnection, the board design cost and complexity has reduced considerably.

PCIe can be demonstrated as layered architecture. It has three different logical layers: Transaction layer (TL), Data link layer (DL), and Physical layer (PHY). PCIe uses packet to transmit information from a transmitter to receiver. Hence it is known as a packet based protocol. The packet is transmitted through different layers. Each layer adds its additional information as described below [3].

Transaction layer adds the Transaction Layer Packet (TLP) header to the data payload. Data link layer adds the sequence id and link level cyclic redundancy check bits to the TLP. At physical layer, the framing bits are added to the packet coming from data link layer. The conversion of logical signals to electrical signals happens in physical layer.

3 Interface design

PCIe interface is a device that acts as a transitional medium between initiator and completer devices. In our design we have PCIe as the initiator or master and memory as completer or slave. Here, we can read data from the memory and can also write data from PCIe to memory [4]. The read request gets a response data back from the memory to PCIe. While the write request gets an acknowledgement signal.

There are different signals that are modelled in interface block. They are named below along with brief functionality.

- **Valid** – It is the activation signal. When it is low the device is valid to perform.
  - Data transmission starts only after high signal value of valid.
• **Command** – It is a 6 bit bus. If this is 000100 then it indicates that the request command is the read. If it is equal to 101100 then it indicated that the request command is write.

• **Address** – It is a 36 bit bus. The request address is transmitted over this signal.

• **Length and Virtual Channels** – Both are 4 bit fields. Length is amount of data per channel. Virtual Channels (VC) is the number of channels incorporated in this. Product of both the values gives the actual amount of data.

• **Ready** – This is a single bit bus. There are 2 types of ready signals. One is initiator ready and other one is completer ready.
  
  o **Initiator ready (Irrdy)** – If it is high then the initiator is ready to perform an action.
  
  o **Completer ready (Crrdy)** – If it is high then the completer is ready to perform an action.

• **Tag** – Count of number of requests issued.

### 4 FSM implementation

The Finite State Machine (FSM) implementation is designed as shown in Fig. 1.

![FSM state diagram for PCIe interface.](image)

**Fig. 1.** FSM state diagram for PCIe interface.

**A. State_idle (S0):** In this state all the signals are in idle mode. Or if they are not in idle state then we are resetting all the signals to idle state. That means all the signals are inactive to perform operation and wait for the master device activation signal. When the valid signal goes high, it activates the master device that means the PCIe is now active to read or write data to slave devices [4, 5].

**B. State_initiator_ready (S1):** If Irrdy changes its state from idle state that is 1, it also changes the state and goes to the next state which is state_completer ready. But
if the Irrdy signal is not high then it waits for this signal to be high. And if the valid again changes its state from high to low, it goes back to the idle state [4].

**C. State completer_ready (S2):** In this state, the Crrdy signal is checked. If it becomes high, then the state change is happened to the state_send_address. If the initiator ready is low again, then is initiator is not ready for the transmission. Hence the state changes back to state_initiator_ready.

**D. State_send_address (S3):** The fourth state is the state send address. In this state at first it checks that weather it is a write transaction or a read transaction. If it is a write transaction and the slave device is memory, then it makes the other internal signals as its need and makes the memory activation signal chip select (CS) as low and write enable high, thus the transaction occurs. If it is a read operation from memory, then again it makes the other internal signals as its need and makes the memory activation signal cs low and read enable high.

**E. State_final (S4):** The tag value is compared with the total number of transactions specified (N). If the tag equals N, then state goes back to idle state. This process will continue until or unless the valid signal goes high. If it goes high, then it stops the transaction and goes back to the idle state.

### 5 Simulation results

Compilation and Simulation of the design is done on Synopsys Verilog Compiler Simulator (VCS) tool. Simulation waveforms for Direct Memory Access (DMA) write and read are as shown in Fig. 2. First set of waves are for DMA write transaction and second set represents the DMA read transaction.

![DMA transactions simulation waveforms](image)

**Fig. 2.** DMA transactions simulation waveforms

Considering the parameters described in previous sections, different test cases are formulated and made to be run on SOC. It is possible to calculate or at least
obtain reasonable estimates of the performance values. In general, the bandwidth is total amount of data transferred in particular amount of time. Overall bandwidth includes the bandwidth values generated by different types of transactions [6].

The theoretical performance values of different DMA cases obtained using formula. For GEN 3 128b/130b encoding is used, while 8b/10b encoding scheme is used for GEN 1 and GEN 2. Encoding scheme cannot be altered unlike some performance factors [6].

\[
\text{Bandwidth} = \left( \frac{\text{Total data transferred}}{\text{Total transfer time}} \right) \times \frac{\text{GB}}{s}. \tag{1}
\]

Hence the final bandwidth values depend on the number of outstanding transactions sent from the PCIe endpoint and the initial and final time instances of the transactions. The total number of outstanding transactions gives a measure of the data transferred. Obtained values are being compared with theoretical maximum values to verify the performance as shown in Fig. 3.

![Comparison of measured bandwidth values](image)

**Fig. 3.** Comparison of measured bandwidth values

### 6 Conclusion

In this paper, we have presented a PCIe interface design to efficiently transmit the initial upstream request information to the memory and downstream response information to the device. We have implemented the interface design using a FSM for DMA transactions initiated from PCIe endpoint to the memory. Bandwidth measured itself is concluding that the PCIe interface is a promising system interconnect for complex SoC’s. Test case scenarios are developed with different configurations to perform the memory access routine. The current performance testing results demonstrate that PCIe link with DMA feature enabled outperforms the remaining interconnects.