International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 06 Issue: 11 | November - 2022 Impact Factor: 7.185 ISSN: 2582-3930

A DRIVER DEVELOPMENT FOR PCI EXPRESS 3.0 IN OPEN POWER ARCHITECTURE

K.Pooja [1],
M.Tech[1], CSE Dept[1], JNTUACEA[1], ANANTHAPURAMU[1] A.P[1], India[1]

Abstract— Peripheral Component Interconnect express (PCIe) is a high speed serial data bus standard. Every computer motherboard has a number of PCIe slots used to add RAM, GPUs, Wi-Fi cards or SSD (solid-state drive) add-on cards. PCI Express is a serial connection that works more as a network than as a bus. Instead of one bus handling data from various sources, it includes a switch controlling various point-to-point serial connections. Such connections will spread outward from the switch leading to the devices where the data is required to go. Devices no longer share bandwidth as they do on a conventional bus. Many different versions of PCIe are developed in order to increase speed, bandwidth and data width. Device drivers are software applications that connect software programming to hardware devices. As part of a huge operating system, device drivers are thought to be particularly challenging to achieve. C as well as other low-level programming languages are regularly used to generate them. Device driver developers must have a detailed understanding of the hardware and software system they are working with. C code will be compiled to binary which will be transmitted into the processor to conduct operations with the specified module. This project goal is to design and develop a driver for PCI Express which is connected to A20 core through AMBA AX14 bus interface. The GCC compiler will be used for design and development of drivers. Drivers will be written in C programming language.

Keywords- Advanced microcontroller bus architecture (AMBA) System-on-chip(SoC), Intellactual Property (IP), AMBA, AXI, VCS, Verilog

I. INTRODUCTION

In the past, there were two general ways to implement transfer system based on PCIe. One is using bridge chip, such as PEX 8311 [3], which not only increases the cost of board making but also brings problems of excessive large board size and high power consumption [4]. The other is making use of Xilinx PCIe Core, but the cores are too rudimentary in nature to be of immediate use as discussed in [5]. Therefore, DSP based on KeyStone architecture, newly produced by TI, embeds ×2 PCIe lanes, and its peak bandwidth can reach a raw speed of 20Gbps [6], which perfectly meets the requirement of high-speed communication between DSP and other PCIe compatible equipment. Regrettably, TI does not provide standard driver template to user just like interface chip manufacturers do, which results in a more difficulty driver development as well as being hard to guarantee its correctness. Therefore, considering the characteristics of PCIe module in Keystone architecture, The AMBA data bus width can be 32, 64, 128 or 256 byte, address bus width will be 32bits wide. The AMBA AXI4 [3] specification to interconnect different modules in a SoC was released in March 2010. A. AMBA AXI4 architecture AMBA AXI4 [3] supports data transfers up to 256 beats and unaligned data transfers using byte strobes. In AMBA AXI4 system 16 masters and 16 slaves are interfaced. Each master and slave has their own 4 bit ID tags. AMBA AXI4 system consists of master, slave and bus (baiters and decoders). The system consists of five channels namely write address channel, write

data channel, read data channel, read address channel, and write response channel. Device drivers are software interfaces between software applications and hardware devices. .As part of complex operating system, device drivers are considered extremely difficult to develop. They are usually developed in low-level programming languages, such as C. The device driver developers must have an in-depth understanding of given hardware and software platforms. Code written in C language will be converted into binary which is fed into processor to perform operations with desired module.

II. AMBA AXI4 PROTOCOL

AMBA AXI3 protocol has separate address/control and data phases, but AXI4 has updated write response requirements and updated AWCACHE and ARCACHE signaling details. AMBA AXI4 protocol supports for burst lengths up to 256 beats and Quality of Service (QoS) signaling. AXI4 has additional information on Ordering requirements and details of optional user signaling. AXI3 has the ability to issue multiple outstanding addresses and out-oforder transaction completion, but AXI4 has the ability of removal of locked transactions and write interleaving. One major up-dation seen in AXI4 is that, it includes information on the use of default signaling and discusses the interoperability of components which can't be seen in AXI3

A. AMBA AXI4 master

To perform write address and data operation the transaction is initiated with concatenated input of [awaddr, awid, awcache, awlock, awprot, awburst]. On the same lines for read address and data operations the concatenated input is [araddr, arid, arcache, arlock, arprot, arburst]. The addresses of read and write operations are validated by VALID signals and sent to interface unit.

B. AMBA AXI4 Interconnect

The interconnect block consists of arbiter and decoder. When two masters initiate a transaction simultaneously, the arbiter gives priority to access the bus. The decoder decodes the address sent by master and the control goes to one slave out of 16. The AMBA AXI interface decoder is centralized digital block. The decoder decodes the address sent by master and goes to one slave out of 16. 0-150 locations are meant for slave-1, next 151-300 addressable locations are meant for slave-and so on till slave-16.

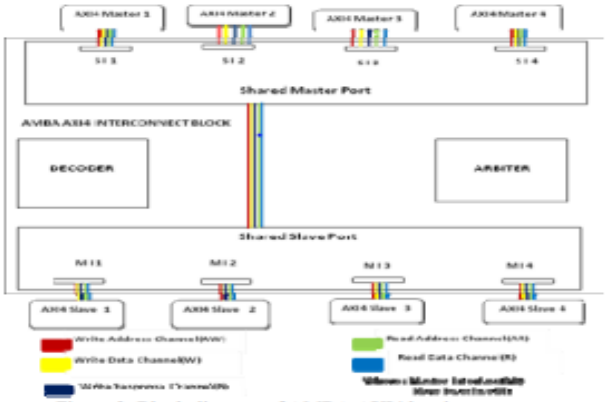


Figure 3: Block diagram of AMBA AXI4 bus interconnect.

C. AMBA AXI4 slave read/write block diagram

The AXI4 slave consists of common read/ write buffer which stores the read/ write address and data as shown in fig. 4. Pending read address register stores the remaining read addresses to be sent; pending write address register which stores the remaining write addresses to be sent and pending write data register which stores the remaining write data to be sent. The read/write state machines receive internal inputs from the read/ write buffer. The AXI4 slave test bench initiates the read or write transaction and the output from the AXI4 slave are standard read/write channel signals. The AXI4 slave receives the write data in the same order as address. Signals used to design slave module is shown in fig. 5. The test layer shown in the fig. 5 has 2 test cases.

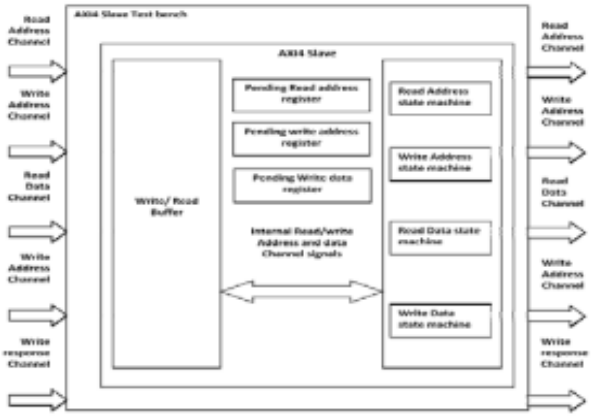


Figure 4: AMBA AXI4 slave Read/Write block Diagram.

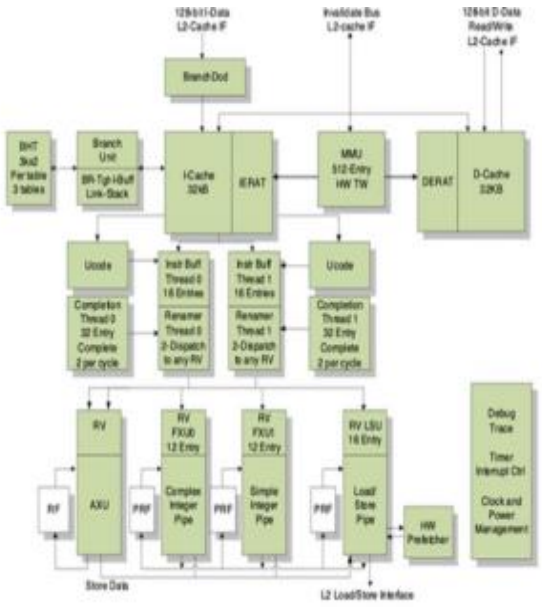
III. EXISTING WORK

With improvement of semiconductor technology and large scale integrated circuit, digital signal processor (DSP) is developing rapidly. Take DSP-TMS320C6678 produced by TI (Texas Instruments) for example, the operation speed has reached 160GB FLOP [1]. Such massive data brings serious problems of data exchange between DSP and host PC. Thus, the PCI and PCI-X bus has been unable to meet the demand of transfer because of its low theoretical bandwidth and high latency time. However, the emergence of PCIe (PCI Express) technology makes people see a new dawn. Since it has numerous improvements over the aforementioned bus standards, including higher maximum system bus throughput, smaller physical footprint and lower I/O pin count, a more detailed error detection and reporting mechanism [2], PCIe gets outstanding effect in high-speed communication. In the past, there were two general ways to implement transfer system based on PCIe. One is using bridge chip, such as PEX 8311 [3], which not only increases the cost of board making but also brings problems of excessive large board size and high power consumption [4]. The other is making use of Xilinx PCIe Core, but the cores are too rudimentary in nature to be of immediate use as discussed in [5]. Therefore, DSP based on KeyStone architecture, newly produced by TI, embeds ×2 PCIe lanes, and its peak bandwidth can reach a raw speed of 20Gbps [6], which perfectly meets the requirement of high-speed communication between DSP and other PCIe compatible equipment. Regrettably, TI does not provide standard driver template to user just like interface chip manufacturers do, which results in a more difficulty driver development as well as being hard to guarantee its correctness. Therefore, considering the characteristics of PCIe module in KeyStone architecture, this paper proposes a general method that can significantly increase transfer rate and decrease the difficulty when designing driver

IV PROPOSED WORK

In driver implementation, data exchange program includes the following three parts: user application, driver in kernelmode and embedded program on DSP. We will introduce specific design details from two aspects: one is communication between the driver and host, and the other is communication between the driver Peripheral Component Interconnect express (PCIe) is a high speed serial data bus standard. Every computer motherboard has a number of PCIe slots used to add RAM, GPUs, Wi-Fi cards or SSD (solid-state drive) add-on cards. PCI Express is a serial connection that works more as a network than as a bus. Instead of one bus handling data from various sources, it includes a switch controlling various point-to-point serial connections. Such connections will spread outward from the switch leading to the devices where the data is required to go. Devices no longer share bandwidth as they do on a conventional bus. Many different versions of PCIe are developed in order to increase speed, bandwidth and data width. Device drivers are software applications that connect software programming to hardware devices. As part of a huge operating system, device drivers are thought to be particularly challenging to achieve. C as well as other low-level programming languages are regularly used to generate them. Device driver developers must have a detailed understanding of the hardware and software

system they are working with. C code will be compiled to binary which will be transmitted into the processor to conduct operations with the specified module.



V RESULTS

Simulation is carried out in VCS tool and Verilog is used as programming language.

A. Simulation result for write operation

The Resent signal is active low. Master drives the address, and the slave accepts it one cycle later. The write address values passed to module are 40, 12, 35, 42 and 102 as shown in fig. 8 and the simulated result for single write data operation is shown in fig. 9. Input AWID[3:0] value is 11 for 40 address location, which is same as the BID[3:0] signal for 40 address location which is identification tag of the write response. The BID[3:0] value is matching with the AWID[3:0] value of the write transaction which indicates the slave is responding correctly. BRESP[1:0] signal that is write response signal from slave is 0 which indicates OKAY

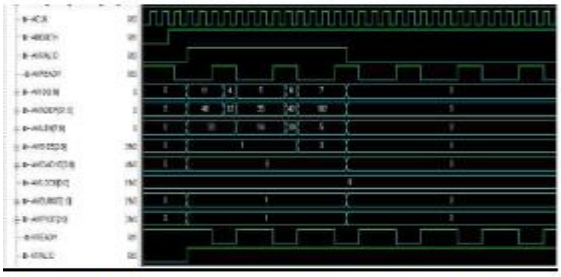


Figure 8: Simulation result of slave for write address operation.

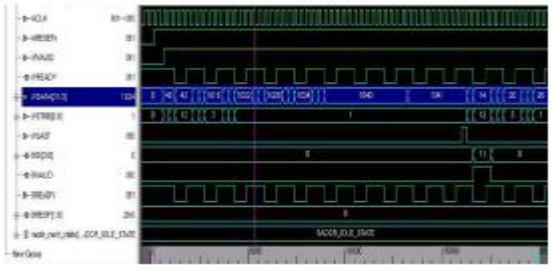


Figure 9: Simulation result of slave for single write data operation.

B. Simulation result for read operation

The read address values passed to module are 45, 12, 67, 98 as shown in fig. 11 and the simulated result for single read data operation is shown in fig. 12. Design of AMBA AXI4 protocol for System-on-Chip communication 42 Simulation result of slave for read address operation. Input ARID[3:0] value is 3 for 12 address location, which is same as the RID[3:0] signal for 12 address location which is identification tag of the write response. The RID[3:0] and ARID[3:0] values are matching, which indicates slave has responded properly. RLAST signal from slave indicates the last transfer in a read burst.

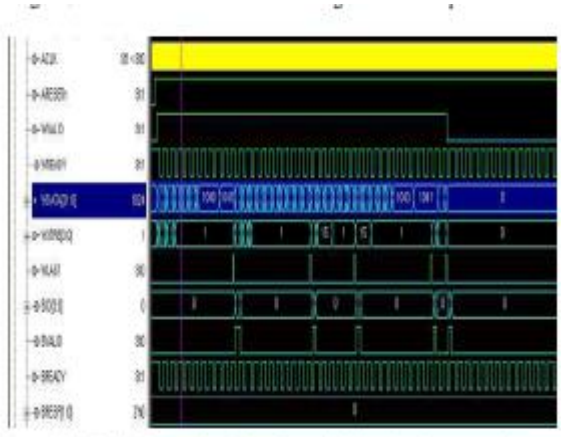


Figure 10: Simulation result of slave for multiple write data operation.

VI. CONCLUSION AND FUTURE SCOPE

PCI Express is a serial connection that works more as a network than as a bus. Instead of one bus handling data from various sources, it includes a switch controlling various point-to-point serial connections. Such connections will spread outward from the switch leading to the devices where the data is required to go. Devices no longer share bandwidth as they do on a conventional bus.

Many different versions of PCIe are developed in order to increase speed, bandwidth and data width. Device drivers are software applications that connect software programming to hardware devices. As part of a huge operating system, device drivers are thought to be particularly challenging to achieve. C as well as other low-level programming languages are regularly used to generate them.

The plan for the future is to provide large amount of memory and efficient data transfer using wi-fi.

REFERENCES

[1] TMS320C6678 Multicore Fixed and Floating-Point Digital Signal Processor, Texas Instruments Inc, http://www.ti.com.cn/cn/lit/ds/ symlink / tms320c6678.pdf.

[2] Ravi Budruk, Don Anderson, and Tom Shanley, “PCI Express System Architecture,” Addison Wesley, 2004.

[3] Zhou Liguo, Liang Huaining, Xie Dongdong, and Wang Zhaokai, “Design and implementation of data transfer card based on PCI Express bus,” Electronic Measurement Technology Journal, vol.30, pp. 28-32, Nov 2007.

[4] Li jing, and Zhao Baojun, “Device Driver Exploitation Based on TMS320C6416 Inline PCI Interface,” Microcomputer Development. vol 15, pp. 135-137, Oct 2005.

[5] Bittner, Ray, “Speedy bus mastering PCI express”, 22nd International Conference on Field Programmable Logic and Applications (FPL), pp.523-526, Aug. 2012.