International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 09 | Sep 2019 www.irjet.net p-ISSN: 2395-0072

Development and Integration of GUI Application with the PCI Express IP-Core VHDL Application

Kavya S[1], J.C Narayana Swamy[2]

[1]M.Tech Student, Dept of ECE, Bangalore Institute of Technology
[2]Assistant Professor, Dept of ECE, Bangalore Institute of Technology

---***---

Abstract - *High bandwidth video bus is a video interface protocol identified in avionics devices for large bandwidth requirements, low latency and uncompressed digital video transmission. This protocol is used to interface system(s) of aircraft to the display in the cockpit. Current day technology uses the Line Replaceable Units (LRU) based on the Field Programmable Gate Array (FPGA) Internet Protocol (IP) cores for communication between two systems. These communication protocols are implemented using FPGA and FPGA integrated modules based Peripheral Component Interconnect (PCI) are used as part of the implementation. These integrated modules are interfaced with the host computers in order to communicate the real LRU as analogous to the aircraft communication. Host machines therefore need to be integrated with the necessary GUI implementation to interact with the PCI Express bus. This paper discusses the development and integration of GUI implementation for PCI Express IP-core with VHDL implementation using JUNGO (WinDriver) device drivers for memory write/read transactions and data transfer for discrete and continuous data.*

***Key Words*: PCI Express; Graphical User Interface; Data Simulator, JUNGO, Chipscope**

1. INTRODUCTION

PCI Express is a standard type of connection for internal devices in a computer. It is a serial expansion bus standard operating at multi-gigabit data rates. It uses a serial interface and allows point-to-point interconnection between the devices. PCI Express provides lower latency and better rate of data transfer compared to the PCI and PCI-X technologies. To enable communication between these devices, device drivers are required, which can be generated using tools like JUNGO (WinDriver) [4].

The JUNGO (WinDriver) tool generates an example code to communicate with devices from the user end operating in byte level write or read transaction (s) and for viewing the configuration registers. The example application generated by the tool was studied, installed and executed. With this example application as the basis, an application adapted to the requirements of data communication was developed.

2. RELATED WORK

The paper [1], examines the improvement of PCI-e interface utilized for fast data transfer to PC. This serial bus architecture has many advantages over parallel PCI bus. The application that require low speed can utilize just a single path of PCI-e wherein applications that requires fast transfer of data can use 2,4,8,16 or 32 lanes of PCI-e depending on speed requirement. Data transfer speed of PCI Express is 2.5 Giga bits per second per lane. The PCI Express interface on virtex-6 series of FPGA is detailed.

The paper [2], introduces the interface between the host computer and the evaluation board of Xilinx ZC706 containing Zynq-7000 XC7Z045, All Programmable System-On-Chip (APSoc) device using PCI Express. It also details the Linux-based write/read operation implementation, allowing information transfers in both the directions via PCI express bus. The performance comparison of the data transfer time at different rates have been detailed.

3. PROPOSED WORK

To establish a communication between the IP-core under test and the host computer, a Xilinx PCI Express IP-core needs to be developed with an appropriate application interfaces. The host computer, writes/reads from/to the specified offset of the memory (in the FPGA) as required by the application. The FPGA blocks include the PIO example design and Xilinx's PCI Express IP-Core. The example PIO design includes a single memory buffer to demonstrate write/read operations of the PCI-e IP-Core. The write port of it is interfaced for writing the data received from the host computer. The read port of the memory is interfaced for reading and looping the data back to host computer. Thus the PIO example application provides a means of testing the PCI Express IP-core write and read operations and is observed on Chipscope.

To interface the PCI Express IP-core with the IP-core under test, the application will provide two parts, a write port to take in the data from the IP-core under test and to send the same to the host computer and a read port to read the data sent from the host computer.

Under this effort, it is proposed to disconnect read port of the memory in PIO and to provide the same interface to the IP-core under test for reading the data sent from the host computer. To provide the IP-core under test with a write port to enable it to send the data to the host computer, it is suggested that a second memory be added with its write port as an interface to the IP-core under test. The read port of this memory is suggested as an interface to the PCI express IP-core for the transmission to the host computer of the information obtained from the IP-core under test.

To enhance the user experience, the sample GUI Application generated by the JUNGO tool for the inserted Xilinx SP-605 Evaluation Board should be modified to display the definitions of the configuration registers.

4. METHODOLOGY

For the SP-605 evaluation board, Xilinx's PCI Express IP-core and the Xilinx example Programmed Input Output (PIO) design are produced. This PIO design is altered according to the requirement. This altered IP is called PCI Express based Data Simulator. On the SP-605 evaluation board, this data simulator IP is flashed to the FPGA. A tool called JUNGO is used to generate the required device drivers. The JUNGO tool's Driver Wizard implementation will produce a generic GUI Application together with the Device drivers. The resulting sample application is tailored by modifying the generated forms and created additional forms. This customized GUI is called Data Simulator GUI.

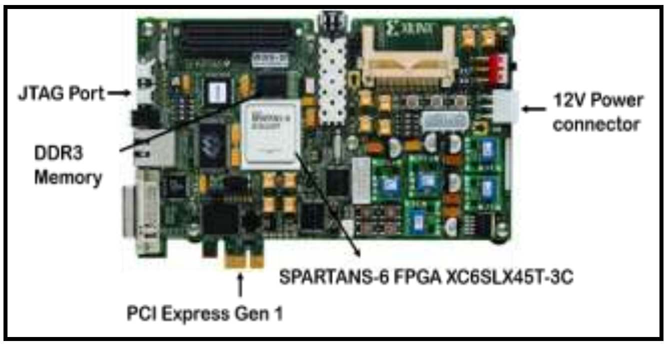


Fig -1: SP-605 Evaluation Board [7]

Device drivers are installed on the host computer and the Data simulator GUI Application is launched to initiate PCI Express transactions. One of the key features of the customized configuration space form is that it provides the user with bit-wise detailed information and thus relieves the user of the need to refer to the PCI Express texts in order to understand the values and their indications.

5. TESTING

5.1 Development of VHDL based Data Simulator

Xilinx's Programmed Input Output (PIO) design is modified to include two buffers, the write and the read buffer which is to be provided as an interface to the communication IP-core under test. This modified PIO is integrated with the Xilinx's PCI Express IP-core to form the data simulator.

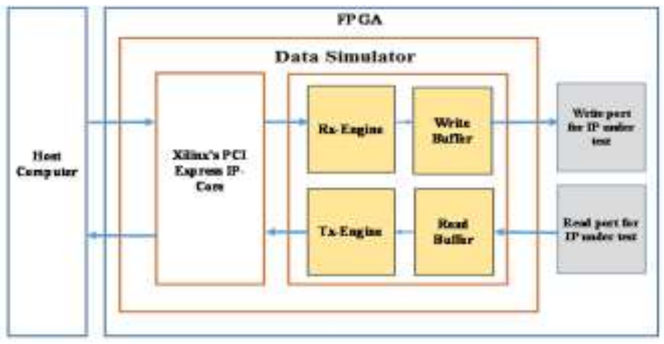

FPGA
Data Simulator
Host Computer
Xilinx's PCI Express IP-Core
Rx-Engine
Write Buffer
Tx-Engine
Read Buffer
Write port for IP under test
Read port for IP under test

Fig -2: Data Simulator

During write operation from the host computer, the data sent from the GUI application by the user received by receive engine of the Data Simulator IP-core on FPGA and the data is written into write buffer and then writes the flag. While performing a read operation from the host computer, it waits for the flag to receive and then reads it in the TX Engine on FPGA. The TX Engine reads the data from that particular address of the read buffer and transmits data on host computer which is read to the user application.

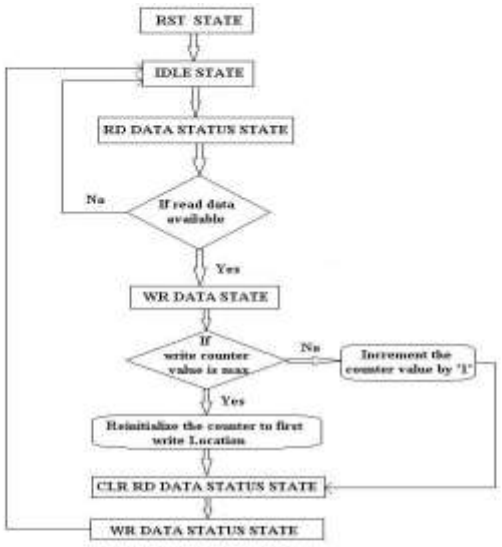


Fig -3: Flowchart for write and read operations in VHDL application

Figure 3, shows the flowchart for the read and write operations which we have followed in VHDL Application. The waveforms of both the data write and data read operation to its write and read buffer are observed on the Chipscope using the Xilinx's Chipscope ILA design unit.

First, initialize the count value to the starting address of write memory. After initializing the count value, check if the read data is available. If the read data is available, then perform write data state as shown in the flowchart above. If not, then go back to idle state. In write data state, check if the counter value has reached the maximum address. If it has reached the maximum address, then reinitialize counter value to first write location. If not, then increment the counter value by "1". After completion of read, clear the read data status of that particular address.

5.2 Device driver Generation

The PCI Express Device driver for the built-in PCI Express interface module is produced using the JUNGO tool. The instrument produces an Information File (INF) with all the libraries needed for the chosen built-in module. Figure 5 demonstrates the flow diagram of the device generation driver. When the built-in FPGA module is inserted into the PCI Express slot, the operating system of the host computer detects device details such as Device Identification (ID) and Vendor Identification (ID). The information will be displayed in the Driver Wizard. Select the device, generate and install the INF. This will install the card's base drivers. The details of the device driver generation and installation process are outlined in Figure 4.

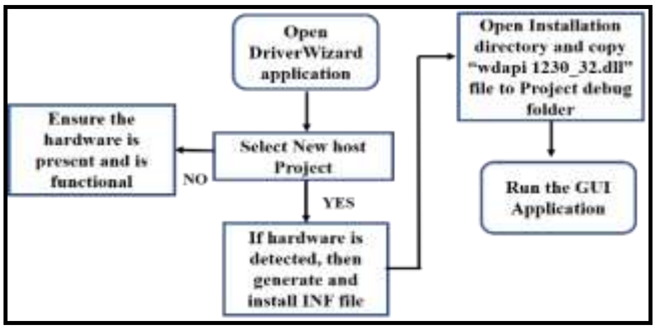

Open DriverWizard application
Select New host Project
Ensure the hardware is present and is functional
NO
YES
If hardware is detected, then generate and install INF file
Open Installation directory and copy "wdapi 1230_32.dll" file to Project debug folder
Run the GUI Application

Fig -4: Device Driver Generation Flow Chart

5.3 Graphical User Interface Development for Data Simulator

In the Laboratory desktop setup, the simulation data/IP-core configuration data/debug data is typically transmitted via the PCI Express bus to the IP-core being tested. One of the critical requirements in this context is the communication with FPGA of data from the host computer application to an external device such as an embedded module (SP-605).

In this case, the host's data will be sent to FPGA and vice versa. The PCI Express data simulator GUI application integrates the JUNGO drivers and communicates the data on the host computer via the PCI Express bus, the IP-core being tested on the FPGA and the data simulator IP-core on the same FPGA as well. The Main Windows form Application (sample form) to access the device is created by JUNGO Tool as shown in the Figure 5.

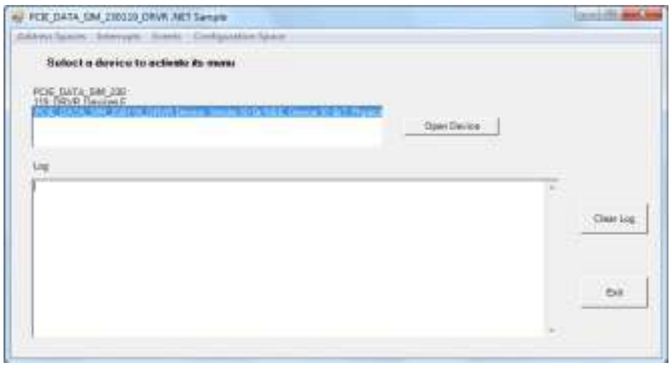

PCIE_DATA_SIM_230120_DRVR .NET Sample
Select a device to activate its menu
Open Device
Log
Clear Log
Exit

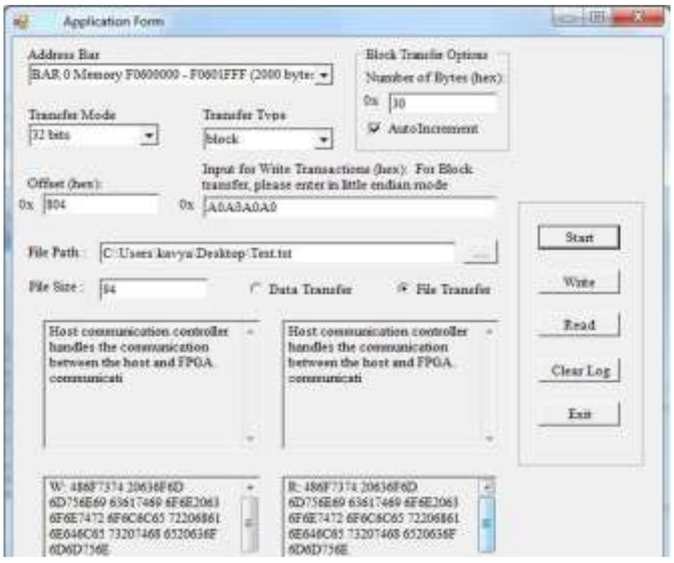
Fig -5: Sample form Generated by JUNGO Tool

5.3 Modified GUI for performing Memory write/read Transactions

The Figure 6 is a modified GUI Application for write/read transactions. The Drivers that are created by JUNGO tool is capable of transmitting single "Double Word" (DWORD), whereas the real prerequisite is to transmit the data continuously. The single DWORD transmitting driver will be used by GUI Application and also designed it to transmit write counter value to write buffer.

The user can browse and select any text file through file transfer button and transfer the data continuously. The data will be converted to hexadecimal and perform write and read operations. This is the prerequisite which has been realized. For converting the data which is in text format into hexadecimal string in C#, "StringBuilder" function can be used.

In the modified Application Form, there are two radio buttons. One will allow the user to select "Data transfer" to perform write/read operations by manually giving the data and the other radio button will allow the user to select "File transfer "where the user can browse and select any text file and size of the file will be automatically displayed to perform write/read operations. The first radio button (Data transfer) allows the user to provide manual hexadecimal data in "Input for write transactions" for write/read operations. Selection of "File transfer" allows the user to browse the path of the file where the input of the data is provided and that will be displayed in "File path". Once the path is chosen, size of the file will be reflected automatically in "File size". The "data transfer" is chosen by default, if the "file transfer" is selected, then the "data transfer" button will be disabled.



Application Form

Address Bar: BAR 0 Memory F0600000 - F0601FFF (2000 byte:

Block Transfer Options
Number of Bytes (hex): 0x 30
☑ AutoIncrement

Transfer Mode: 32 bits

Transfer Type: block

Offset (hex): 0x 804

Input for Write Transactions (hex): For Block transfer, please enter in little endian mode
0x A0A0A0A0

File Path: C:\Users\kavya\Desktop\Test.txt

File Size: 64

○ Data Transfer [● File Transfer]

Host communication controller handles the communication between the host and FPGA communicati

Host communication controller handles the communication between the host and FPGA communicati

W: 486F7374 20636F6D 6D756E69 63617469 6F6E2063 6F6E7472 6F6C6C65 72206861 6E646C65 73207468 6520636F 6D6D756E

R: 486F7374 20636F6D 6D756E69 63617469 6F6E2063 6F6E7472 6F6C6C65 72206861 6E646C65 73207468 6520636F 6D6D756E

Fig -6: Modified Application form for write/read transactions

There are four textboxes as shown in the Figure 8. The first textbox (Top left) writes the data which is in ASCII format as chosen by the file. Once it is written into the memory, it converts the ASCII format of the provided data into hexadecimal format and writes it in the second textbox (bottom left). Then the received hexadecimal data will be read in the third textbox (bottom right) and the hexadecimal data which is read from the memory will be converted back to ASCII format and displays it in fourth textbox (Top right) in order to verify the input data which was provided earlier.

The write/read transaction(s) will be performed and a log will be displayed simultaneously in order to verify the operations which are being performed. These all conversion of ASCII to Hexadecimal and vice versa will be written and read to/from the memory at once by clicking on the "Start" button. To perform write/read operation separately, the user can make use of the buttons "write" and "read" which is shown in the Figure 9. To perform write operation, the data will be written by clicking on the "write" button. Similarly, for the read operation, if we want to read the data from the memory then by clicking on the "read" button, the data will be read.

The write and read operations were performed and the waveforms were observed on the Chipscope.

6. RESULTS

6.1 PCI Express Data Simulator Application screenshots

While performing a write operation from the host computer, the data sent by the user from the GUI Application is received by Receive Engine of the Data Simulator core and written into the write buffer. The write transaction waveforms observed on the Chipscope are as shown in Figure 7.

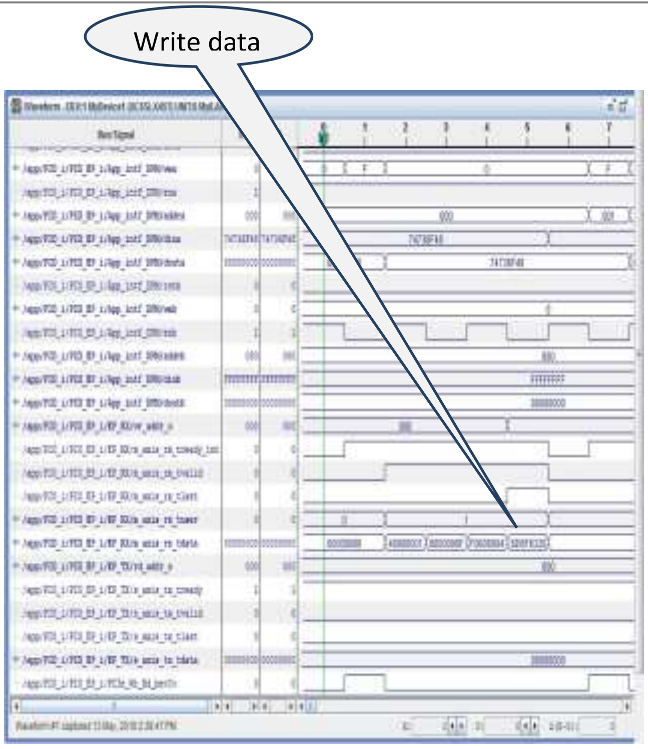


Fig -7: Write operation with Address and Data

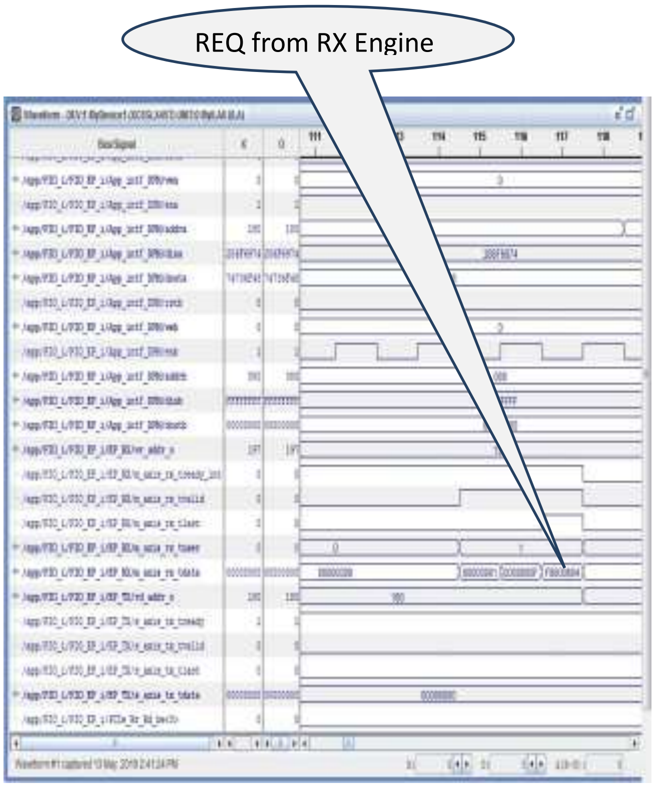


Fig -8: Read operation with Request from Rx Engine

For read operation, the host computer sends a read request to the RX-Engine. The RX-Engine sends the first 3 Double Words (DW's) of the request, which contain the FMT type, requester ID and the address to the TX-Engine. The TX-Engine transmits the header DW's (3 DW's provided by the RX-Engine which contain FMT type, requester ID, address along with the data read from the read buffer to the host computer. The read request from host computer to the data simulator is shown in Figure 8 and the data transmission from data simulator to the host computer is as shown in Figure 9.

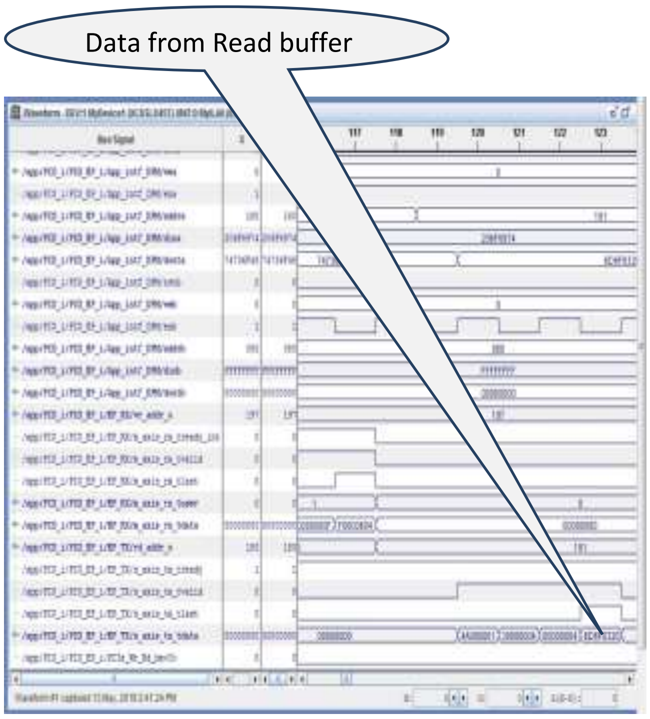


Fig -9: Read operation with Data

7. CONCLUSION AND FUTURE WORK

In this Project, we have presented a case for establishing communication link between the IP-core under test on the FPGA (SP-605 evaluation) board and host computer using PCI Express IP. The waveforms are observed on the Chipscope for write and read operations.

Additionally, elaborated and generated device drivers from the JUNGO (WinDriver) tool which help to have a communication from the GUI Application with the PCI Express card. By selecting the default BAR address, transfer modes, transfer type and offset value, memory write/read transaction(s) were performed and later verified the transmission of data on Chipscope Pro Analyzer. The PCI Express based data simulator GUI is tested successfully for discrete as well as continuous data.

The data was communicated with the Xilinx SP-605 evaluation hardware board and checked for accuracy. The sample GUI provided by the JUNGO tool is modified to transmit random hexadecimal data along with text and word file. This is converted into hexadecimal format so as to perform the write and read operation.

In future the bus master DMA implementation may be considered, for the purpose of faster transfer of data between host and the Data Simulator IP, which is required to cater to the IP-cores under test that demand higher data rates and also to enhance the GUI for improved user experience.

ACKNOWLEDGEMENT

We express our thanks to the Department of Electronics and Communication, Bangalore Institute of Technology (BIT), Bengaluru for supporting the work.

REFERENCES

[1] Vijitha C.V, Najla A.P, Jayaraj.U, Kidav M.E "PCI Express Interface Development and Simulation for High Speed Data Transmission" Applied Electronics, VCEW, Tamil Nadu, India Mtech Electronics Design Technology, NIELIT, Calicut, India Scientist, NIELIT, Calicut, India.

[2] Rjabov A, Sudnitson A, Skliarova I (2016) Interaction of Zynq-7000 devices with general purpose computers through PCI Express: a case study. In: Proceedings of MELECON 2016.

[3] PCI Express system Architecture by Ravi Budrak, Don Anderson, Tom Shanley

[4] WinDriver PCI/ISA User's Manual by JUNGO Connectivity Ltd.

[5] Spartan-6 FPGA Integrated Endpoint Block for PCI Express User Guide.

[6] PCI-e Technology by Mike Jackson and ravi Budrak

[7] http://www.xilinx.com/SP-605

[8] JUNGO Connectivity Ltd Version 12.3.0