International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 4 Issue 08, August-2015

Optimum Architecture Design of PCI bus

Amit S. Mamidwar
Department of E&TC
Sinhgad College of Engineering
Pune, India

Prof. Vrushali G. Raut
Department of E&TC
Sinhgad College of Engineering
Pune, India

***Abstract*— Optimum utilization of power and area along with variable bandwidth are requirements of digital component interconnect in several embedded system. Peripheral component interconnect (PCI) bus is widely used in embedded system for communication. In present applications such as computers, routers, XBOX and so on, many embedded component needs connection with PCI bus. However, these digital components don't have the bus interface of PCI with minimum power and area. In order to solve this problem, this project intends to implement FPGA-based PCI bus of low power and smallest area. By the analysis of PCI bus and corresponding signal sequence for interface, the top-down method is applied to modeling of PCI bus into seven functional modules such as base configuration register module, state machine module, address check module, etc. This helps to design an optimized architecture for each module easily. Besides, the master and target modules of PCI bus are structured in finite state machine and programmed with utilization of VHDL language. It helps to minimize the state transition delay. Through the simulation it has been proved that the interface can meet the requirements of PCI bus for communication between digital components.**

Keywords*— *PCI bus, Power Consumption, Optimum Architecture, State machine design

I. INTRODUCTION

The peripheral component interconnects (PCI) bus is widely used in the embedded system. PCI is a bus for attaching digital component in hardware devices. This works as an intermediate bus for communicating between the system I/O's and other peripheral devices. It helps to communicate with a high data rate. In applications like computers, Routers, ATM's or XBOX and so on, PCI used as a communication bus between the processor and other components like graphic cards, memory; etc. [5].

Energy saving is a key parameter of Embedded equipment. The PCI bus uses parallel communication; therefore it may consume high power, hence there is a need to design low-power PCI. The PCI bus performs read-write operation for single or multiple bytes of data. It takes initiative to read-write operation. Generally PCI takes time to switch from single byte to multiple byte read-write operation. This state transition delay has to be minimized, which improves the speed of the device [4].

PCI bus processes the control signals, system functions, address and data. It has to be accessed throughout the modules (master or slave). The master device directs the bus and it chooses the target. It consists data/address (AD) bus. For transmission of data, it takes one or more clock delay which depends on the transmission mode of operation. Also, the digital components need minimum power interfacing. The PCI bus uses parallel communication which increases power consumption [1].

This paper is designed FPGA based PCI bus for low power and minimum area. The top-down method is applied to the PCI bus. It divides the PCI into seven functional modules which helped to optimize the architecture for each module easily. The state machine is designed for the master and target modules. It minimizes the extra state transition delays which introduces at the time of read or write operation.

II. LITERATURE SURVEY

Literatures have been collected not only from research but also from review articles. The PCI Bus was initially developed in 1992, for graphics oriented operating systems like Windows. It has needed interconnection between the processor and display peripherals of standard PC architectures. The peripheral uses with high bandwidth specifications nearer to the processor bus can improve the speed. Significant performances are seen with graphical user interfaces and other high bandwidth functions like full motion video, SCSI, LANs, etc. when a bus designs is used. PCI meets these specification difficulties of the industry successfully and is now the most widely used and realized expansion standard in the world. [5]

The initial version of conventional PCI established in consumer desktop computers was a 32-bit bus with a 33 MHz bus clock and 5 V signal, although the PCI standard gave a 64-bit variant as well. The PCI standard introduced 3.3 V slots, physically differentiated by a flipped physical connector to avoiding accidental insertion of 5 V cards. Universal cards, which can work on any voltage, have two notches. Version 2.1 of the PCI model introduced optional 66 MHz operation. A server-oriented alternative of conventional PCI, called PCI-X operated at frequencies up to 133 MHz for PCI-X 1.0 and up to 533 MHz for PCI-X 2.0. An internal connector for laptop cards was introduced in version 2.2 of the PCI specification. The PCI was also adopted for an external laptop connector standard. The first PCI specification was introduced by Intel, but following development of the standard became the responsibility of the PCI Special Interest Group (PCI-SIG) [7].

The latest version of the PCI bus is PCIe 3.0x16 works for a point to point connection. It works for two devices only and other devices cannot connect. PCIe works for serial communication where as PCI bus works for parallel communication. The PCI is highly performed I/O bus, which use peripheral devices to interconnect in the application of communication. PCI interconnects the I/O devices for the connection and computing of embedded system. The PCI bus is having the capability to update operation to purpose with power management. But each internal device like state machine, configuration space and address command controller needs to inform the register information, state

information and enhancement in play and plug capability. It consists device controller, message controller, mode controllers and so on. It increases the power consumption of the device up to several mili watts for normal PCI bus [8]. To this active system, this design tries to replace the PCIe with PCI bus with high bandwidth which connects to more than two devices for the required application and adding the devices in the PCI bus, direct master and target state machine which will minimiz the power consumption [9].

The processor connects to the memory controller through front side bus. This controller connects to the PCI. PCI provides the separate memory and I/O port address spaces. Address spaces provides the addresses by the software. It uses the fix addressing scheme. PCI consisting of the 32-bit and 64-bit PCI cards to process [10]. It causes problems in general PCI as:

A. Mixing of 32-bit and 64-bit PCI cards:

A 32-bit PCI will function well appropriate in 64-bit PCI slot, but speed will limits the clock frequency of the slowest card. Many 64-bit PCI cards designed to work with 32-bit mode, if inserted in shorter 32-bit connectors, with some loss of speed. When a 64-bit card installs in 32-bit slot will leave the 64-bit portion of the card edge connector not connected and overhanging [11].

B. Communication system for FPGAs:

A capability requires a considerable infrastructure, which is built into the PCI chip of the device. For data achieving system, it is often interested to have capability on FPGA where other functionality can be used as well. There are two types of FPGA-based PCI solution- embedded system design, where there is an entire CPU infrastructure on FPGA, and standalone design with limited functionality and interface, but with low resource requirement. The main focus is to minimize the resource utilization on FPGA while maintaining reliability and efficiency of the design [12].

III. SYSTEM MODELLING

The PCI bus implementation designed using several modules; it uses master as well as target module. PCI Bus divides the module in master/target control fsm, PCI address decoder controller, parity checker, configuration space, info transfer and device clock manager.

A. Efficient PCI Model Design:

Figure 1 shows the functional block diagram of PCI Bus with several modules. These modules connect with a modular method to design operation. The entire read-write operations must be allocated by the state machine to control the bus. When used as the master device, it can apply to take the bus initiative. Data are then transmitted in bursts or single byte to the destination address. A burst transfer carries segment of address and several segments of data. It requires that the target device and the master device must understand the implicit addressing. The state machine module is separated into a master device and the target device. Each module is planned with VHDL by XILINX.

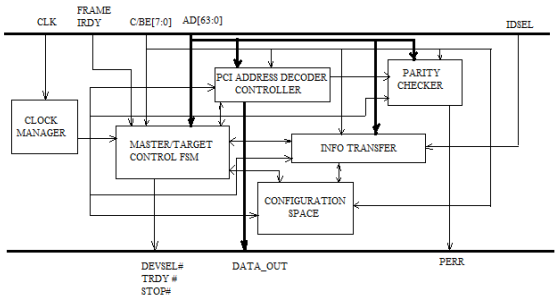

Fig 1: Functional Block Diagram of PCI Bus

a. Configuration Space

This module is used to configure registers with configuration space. Each device space on PCI gets the configuration space for the bus function. The core interface implements a zero configuration space header. The initial 64 bytes are used for standard space header of the bus. The next 64 bytes are used for the future capabilities items for the PCI bus interface. The next 128 bytes are used for the user applications. The user application returns to zero for all accesses of configurations. These user applications have ability to implement additional capability items for this space area, or implement registers for outside scope of specifications.

b. Info Transfer

This module uses different control logic design. These control logic design is complex control logic. The logical sequences are intended on the basis of address and command behaviour. A different stage of data, address and command is used for multiplexed the output. When the state asserts the address signal, the output also addresses the transaction of address.

c. Address Decoder Controller

This module is used to confirm the device's IO/MEM base address. It has been allotted by the system during the base register configured write cycle, and then in the address cycle of write and read operations. This module evaluates the address from PCI bus data sampled with the previous base address, so that it can determine whether the current operation of the device is for itself.

d. Parity Checker

This module is used to compare the master device address and target device address. If the address of target and master matches or data transmission is correct, it asserts the parity. The device verifies the AD and C/BE through latching. The XOR gate is generates or checks the parity. At the next clock edge, the device will generate the XOR between its calculations for even parity. If the two values are same, even parity is generated. Otherwise, it reports thee data error at the next clock pulse.

e. Master/Target Control FSM

The module of the control FSM is the most useful part of the PCI bus interface. The FSM machine selects the current status of operation by determining the signal on the bus. It can deal with the process with other modules and it receives signal from other modules. It changes the control signal by analyzing the PCI and the signal from other modules,. The

state FSM relates the signal in accordance with PCI bus and with input from the PCI bus.

The PCI bus transaction completes the state machine design, which is divided into the master device and the target device.

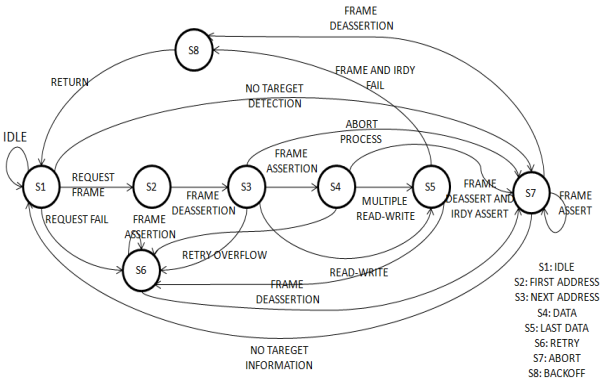

Fig.2 State Machine for Master

Figure 2 shows the FSM state machine for master. It has eight states which take one clock for each transition. The state machine has 5 kinds of situations as unsuccessful occupancy, no answer from target; single data read operation, information retry operation and multi-data read-write operation.

The state transition of PCI master state machine has 5 kinds of situations as follow:

1. Unsuccessful bus occupancy: Here, idle state checks for the operation. If it won't get command or data, it goes to retry state. A retry will try to get command and data for a small number of counts. If the counter overflows, then condition again comes back to idle.
2. No answer from the target: Similar to the first condition, if device is not providing any control signals to master, it goes to the abort state. This state resets all output of the master and move to backoff state. Again, it comes back to idle state.
3. An effective single-cycle operation send: When idle states receive command for single data operation, it will check to address at addr1 state. After receiving address, it checks for data at addr2 and moves to last_data. Finally, it approaches back to idle state via backoff state.
4. An effective burst operation send: When idle states receive command to burst operation, it checks the first address at state addr1. After getting address, it checks for first data at addr2. Later it will struggle for second address and moves to last_data. Here it receives second data. Finally, send all data to target and comes back to idle state via backoff.
5. The operation of retry: When idle states receive command for single or burst operation, it confirms the first address at state addr1. After receiving address, it checks for first data at addr2. When addr2 will not receive any data, it will struggle for small number counts. If the counter overflows, condition comes back to idle.

Figure 3 shows the state machine for target device. It has twelve states. Similar to master FSM state machine, it needs single clock cycle for each transition. FSM state machine works for unsuccessful occupancy, read-write operation for configuration, single data read, multi-data read, single data write, multi-data write, retry operation and backoff operation.

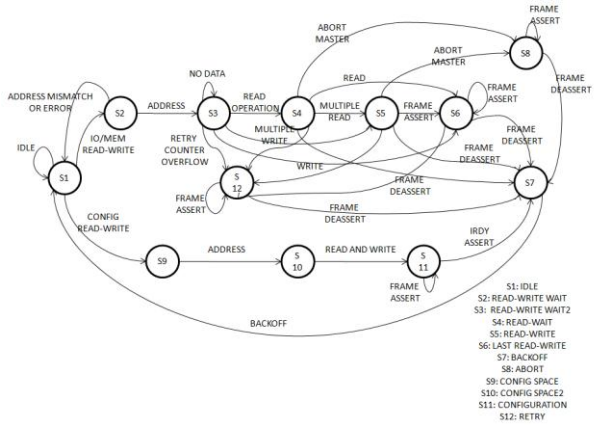

Fig.3 State Machine for Target

The state transition of PCI target state machine has nine kinds of situations as follows:

1. Wrong address: Here, idle state checks control signals for target and moves to Read-write wait state. If device will not be able to get address or mismatch occurs for address, it goes back to idle state.
2. Read and write of configuration: To read and write operation of configuration space, it checks for address at config_wait. It checks data at state config_wait2. Finally, it goes to backoff state by obtaining condition from configuration_ state.
3. An effective single-cycle read: For single cycle read operation, it ensures for address at read_write_wait state. At read_wait_2, it checks for command, whether it is single cycle or burst read operation. For data read, it moves to read_wait state and stops at last_read_write. Finally it goes to idle state through the backoff state for resetting all conditions.
4. An effective burst read: For burst read operation, it checks for address at read_write_wait state. At read_wait_2, it checks for command, whether it is single cycle or burst read operation. For initial data, it moves to read_wait state and checks next address at read_write. It receives next data at last_read_write state. In the end, it goes to idle state through the backoff state for resetting all conditions.
5. An effective single-cycle write: Similar to single cycle read operation, an effective single-cycle write operation works through read_wait, read_wait2 and last_read_write state, finally it comes to idle state.
6. An effective burst write: This condition is very similar to burst read operation. Writing operation applies the read_write_wait state for address, read_write_wait2 for data, read_write for next address and last_read_write for last_data.
7. The operation of retry: If device is not getting any information it will try for little count for retry operation through read_write_wait, read_write_wait2, and if its counter overflows, it goes to backoff state.
8. The operation of Target failure: If target is not able to receive any data or command, then it goes for an abort condition through read_write_wait2, read_write or

read_wait. After reaching state abort, it goes back to backoff state.

9. Disconnected operation: If the device gets disconnected from PCI, it will stop all operations from any state (read_write_wait2, read_write or read_wait) and comes back to idle state.

IV. Results And Discussion

The PCI bus defines three basic read-write operations: configuration, Memory and I/O read-write. Configuration refers that the PCI master does the operations of read-write to configuration registers. Memory or I/O read-write is effectively the same. The only difference is that the corresponding operation addresses are different. The corresponding test bench is written respectively by XILINX ISE 13.2 Design Suite.

A. Simulations of Modules:

The PCI bus implementation designed using several modules. The simulation of read, write, error detection and configuration operation is as follow:

a. Master/Target Control FSM:

PCI takes time to switch from single byte to multiple byte read-write operation. This state transition delay has been minimized by designing the FSM machine by using both master and target. It improved its work by minimizing additional states required for read-write operation. Figure 4 shows the control FSM state machine for the master and target that explains the combined operation for single cycle read-writes, multiple cycle read-writes and retry operation. Status outputs used to wait until address and data load.

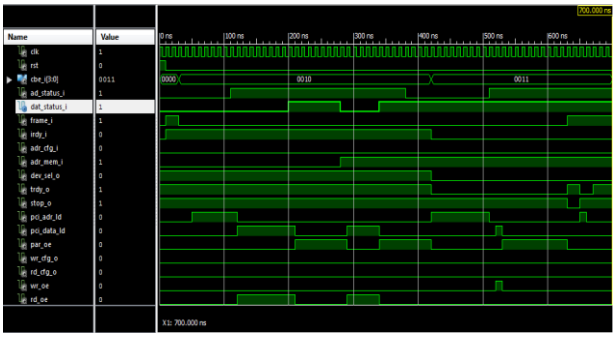

Fig. 4 Simulation for Control FSM

Initially the cbe[3:0] takes “0000”, it gets interrupt command, i.e. no operation command. For “0000”, frame_i is getting assert for less cycle and later it gets deassert. If frame asserted and address status gets low; a dev_sel_o, stop_o and target ready trdy_o goes to ‘0’. Similarly, it verifies parity condition on par_o. Also, it confirms the operation for read-write on rd_oe, wr_oe, rd_cfg_o, wr_cfg_o for I/O devices, memory or configuration space. If the value of cbe_i [3:0] is “0011” means it takes I/O write command, similarly if it is “0010”, so it takes I/O read data.

b. Address Decoder Controller:

Fig. 5 showing simulation for data decoding, which takes command from input and decoded for required form of signal for I/O, memory or configuration space. It gives byte information to design to the configuration space bar_0.

In fig. 5, c/be(7:0) line takes command for PCI and tries to decode information. When id_sel_i is ‘1’ then the device will start work. Bus ad(63:0) is a bidirectional bus which takes data and decodes. A pciadr_ld_i is used for loading the data. A bar0_1 and bar0_2 takes the byte information and moves information to configuration space by using adr0_1 and adr0_2.

Fig. 5 Simulation for Address Decoder Controller

When io_en_i is ‘0’ then adr0_1 and adr0_2 goes to high impedance state ‘Z’. Output cmd_o takes the commands cbe(3:0) for read-write command operation.

c. Info Transfer:

Fig. 6 shows the simulation for Info Transfer. It takes data from data_i and transfer to the I/O or register depends upon command line. It takes enable inputs to read write and also takes information from pci input or register input.

Fig. 6 Simulation for Info Transfer

The cbe_i(7:0) takes command for read-write operation. A data_i(63:0) and rg_data_i(63:0) takes data from I/O or memory and configuration space (register). If command cbe(3:0) is “0010” then it transfers data to data_i and if cbe(3:0) is “0110” then it transfers data to data_o. Similarly if command is “1011” then it transfers data to rg_data_o.

d. Configuration Space:

In the process of configuration write test bench simulates the system to allocate configuration space resources to the target. It represents that the system can receive the command of master aborts. The results of the configuration write operation is successful.

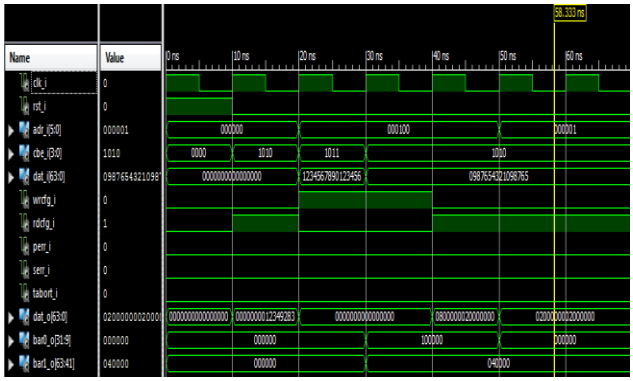

Fig. 7 Simulation for Configuration Space

The values of cbe_i are 1010 and 1011, which means that at this time the operation of a configuration read and write are done, respectively. After the given configuration register address, the data immediately send in the next cycle.

e. *Parity Checker:*

The core interface generates and checks parity and reports error as required by the PCI Local Bus Specification. The function is completely transparent to user. It needs to know if an error occurred. It introduces if parity error occurs during the data phase.

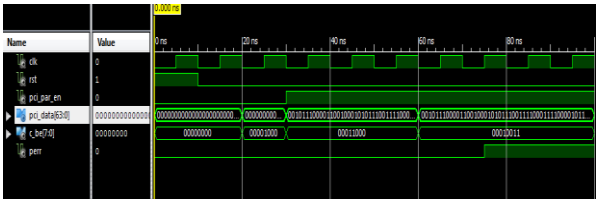

Fig. 8 Simulation of Parity Checking

Figure 8 shows the result for parity detecting for target. According to user necessity it will use either several checkers which will recover its operation faster for the device. Both devices work faster which minimize the power consumption and area. If pci_par_en is '1', it gets enable to ensure the parity condition. Perr gives parity checking error for both pci_data[63: 0] and cbe[7:0] with single bit output. Initially pci_data[63:0] and cbe[7:0] sets to all '0'. If pci_data and cbe gets even, then perr gives output '0' and if data and cbe is odd, then it shows the parity error as '1'. It works as successful even parity checker for getting the parity error information.

f. PCI Simulation:

Fig. 9 shows the simulation of PCI bus top module. PCI bus takes command of read operation from I/O when cbe(7:0) is X"02". It transmits data for I/O write operation when cbe(7:0) is X"03" outside the PCI. Data_in receives data and gives data to data_out on right command. In simulation, Data has received at 40 ns, and data_out receives it at 220ns.

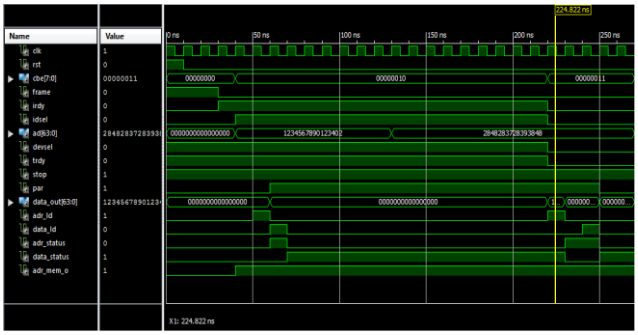

Fig. 9 Simulation for top module PCI Bus

B. Performance Measurement Parameter:

The proposed system in this project is not only used to minimize the power the PCI. The system also minimize to utilization of devices. Table 4.1 shows the power consumption. Table 4.2 shows test results for utilization of slices, slice flip-flop and 4-input LUTs for the PCI device on SPARTAN-3E kit.

Table 1: Text Power Report

	Clocks	Quiescent	Total
Power (mw)	3.80	320.67	324.48

Table 2: Number Of Device Utilizations

Device Utilization		
Slices	Slice Flip-flops	4-Input LUTs
191	172	253

V. CONCLUSION

The work presented in this project gives the fpga-based pci bus of low-power devices. The interface of pci bus is representing program with the top-down approach that modules into top layer, state-machine, configuration register, base address check and even-parity are designed. This minimizes the area of pci bus due to optimize design of modules. The state transition delay has been minimized by designing the fsm by using moore state machines for master and target. It improved work of pci by minimizing additional states required for read-write operation. The utilization of number of slices are 191, number of slice flip-flop are 172 and number of 4-input luts are 253. The simulation of read, write operation will prove that the design conform pci bus specification as compare to xilinx pci 64_ug262. This work gives the low-power embedded device increase to applications of pci bus. It will meet requirements of the design. For manufacturers, pci bus interface would be reserved. It will not affect on resources and its waste.

REFERENCES

[1] Mingji Yang, Lei Wu, Haokun Shi, and Harbin, "Design of PCI-104 Bus Interface of Low-Power Embedded CPU Based on FPGA", IEEE International Conference on Measurement, Information and control, vol. 01, pp. 275-279, August 2013.
[2] Hossein Kavianipour, Christian Bohm, "A High-Reliability PCIe Communication System for Small FPGAs", 2013 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), pp.1-4, Oct.2013-Nov.2013.
[3] Mohammad S. Sharawi, "Signal Integrity Characterization and Modeling of a PCI/PCI-x 66/133 MHz Bus", Circuits and Systems, 2008. MWSCAS 2008. 51st Midwest Symposium on , pp.490-493, August 2008.
[4] Ms. Awani N. Gaidhane, Mrs. Manish Pankaj Khorgade, "FPGA Implementation of Serial peripheral Interface of Flex-Ray Controller", Computer Modelling and Simulation (UKSim), 2011 UkSim 13th International Conference on , pp.128-132, March 2011.
[5] PCIBook, "PCI Local Bus Specification", PCI Group, revision 2.3, Portland; 2002.
[6] XILINX User Guide, "LogiCORE IP Initiator/Target v4.18 for PCI", pci_64_UG262, User Guide XILINX, October 2012.
[7] S. Sumathi; P. Surekha (2007). LabVIEW based Advanced Instrumentation Systems. Springer. p. 305. ISBN 978-3-540-48501-8.
[8] Qiang Wu; Jiamou Xu; Xuwen Li; Kebin Jia, "The research and implementation of interfacing based on PCI express," Electronic Measurement & Instruments, 2009. ICEMI , vol., no., pp.3-116,3-121, 16-19 Aug. 2009
[9] Wang Lihua, "Design and Simulation of PCI Express Transaction Layer," Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on , vol., no., pp.1,4, 11-13 Dec. 2009.
[10] Mu-Shan Lin; Chien-Chun Tsai; Chih-Hsien Chang; "A 5Gb/s low-power PCI express/USB3.0 ready PHY in 40nm CMOS technology with high-jitter immunity," Solid-State Circuits Conference, 2009. A-SSCC 2009. IEEE Asian , vol., no., pp.177,180, 16-18 Nov. 2009
[11] Saleh, H.; Engels, R.; Reinartz, R.; Reinhart, P.; Rongen, F., "A flexible compatible PCI interface for nuclear experiments," Nuclear Science Symposium, 1997. IEEE , vol., no., pp.704,706 vol.1, 9-15 Nov 1997
[12] Chee Wei Liang; Zain Ali, N.B.; Seth Nair, R., "Design of low cost FPGA based PCI Bus Sniffer," FPT, 2003. Proceedings. 2003 IEEE International Conference on , vol., no., pp.420,423, 15-17 Dec. 2003