FPGA based PCI Accelerator Card

Group 1
Pooja Nagle (07d04003), Nikhil Kumar (07007022)
Supervisor : Prof. Jayanta Mukherjee

Abstract

The goal is to make a PCI plug-in card with in-system programmable FPGA along with supporting drivers and sample application software. This device can be used to accelerate scientific computing problems by implementing a suitable parallelized programme on FPGA.

In future, we also wish to incorporate an external RAM memory chip which may be used to further enhance performance for memory intensive computing problems.

Motivation behind the Project

Our project can is aimed at gaining experience with the PCI and FPGA based design. The benefits can be seen in 2 fold

1. The product can be used as a low cost generic FPGA board with external memory able to communicate using pci protocol. This PCI card can be used for education/protoyping purpose.
2. A custom build specialized accelerator card with multiple FPGAs and desired memory bank can be built cultivating on our experience with the project.

PCI bus and its successor has always been known for its high throughput and minimum latency as compared to any other external interfacing protocol. The successor PCIe v2.0 has a capacity as much as 8 GB/s. On the other hand FPGAs over the last decade have matured enough that its performace are being compared to that of ASICs. Supplementing the computational power of FPGA with communication capability of PCI variants protocol and storage capacity of fast RAMs can result in one of the best hardware accelerator, if properly synched.

At University of Tokyo, the GRAPE (GRAvity PipE)[1] project achieved a processing speed of 1TeraFLOPS and fetched Gorden Bell award for proving best price-performance of their custom designed supercomputer at a stagrring $7/MegaFLOPS way back in 1999[2]. Their machine was built of several pluggable board with scores of ASIC specifically designed for simulating N body problem with gravity. Currently a similar project for molecular dynamic known as MDGRAPE[3], is under development.

Our project can be seen as a preliminary step towards making such high performance machine for accelerating different scientific computing problem by mapping their kernel on FPGA/ASIC.

Product Design Process

Our design process is split up into two stages that will proceed in sequential order. The first stage aims at building the basic blocks of final product and hence has a lot of redundant work which won't be utilized in final product but is present simply to supplement the understanding and workability of different blocks. The second stage aims at putting all these block together to make the final product. The final application development will be done at the end.

Stage I: It has following sub-blocks
1)JTAG board
2)Atmega128 application board
3)FPGA board
4)PCI board with microcontroller

Stage II:
1)PCI Accelerator Card
2)CFD Kernel Demo Software

Stage I

1.JTAG board

JTAG board is a standalone pcb board which connects to PC through USB at one end and has JTAG interface at the other. JTAG is the industry standard protocol for test, debug and programming of programmable devices. We will be using it for programming FPGAs.

1.1 Hardware Design

The pcb for JTAG dominantly contains a usb-pci bridge IC FT2232H and pheripheral active and passive components. An external EEPROM AT93B46 is used to store the configuration of usb end-controller like the device name and critical information regarding the JTAG interface. The board is USB power driven. Adjustable linear voltage regulator LM317 is used to generate 3.3V from 5V USB bus potential. Figure 2 shows the schematic diagram of circuit.

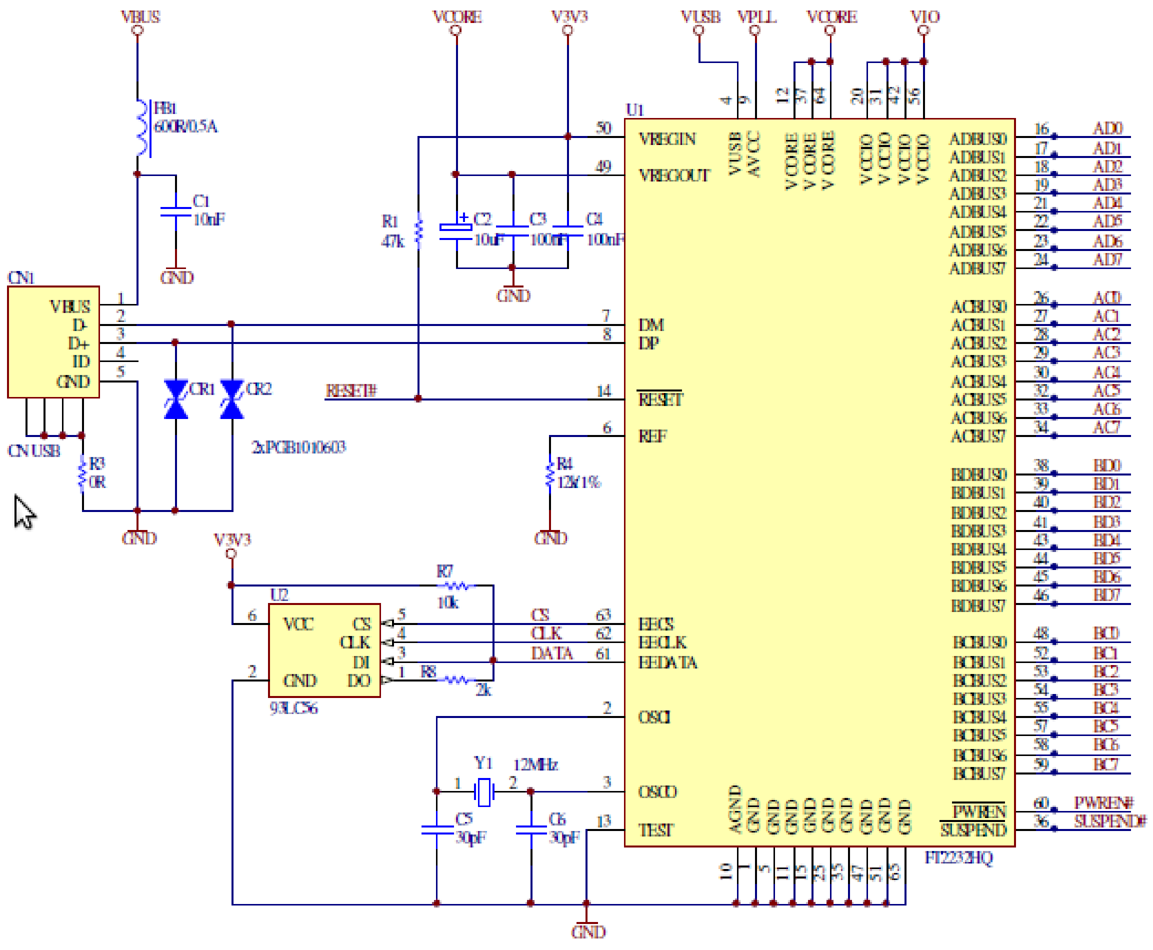

Figure 1:Schematic of JTAG board

1.2 Software Design

The manufacturer of bridge IC, i.e, FTDI Inc provides a user friendly driver which handles the usb device and can virtually map the usb port of PC to JTAG port of our device. This virtual port can then be utilised by different softwares to talk with end device, i.e, device under test/program. Some of the free and open source software that support programming using FTDI 2232H bridge IC are openOCD and urjtag. Currently, we are using urjtag as it has better device support and more user friendly.

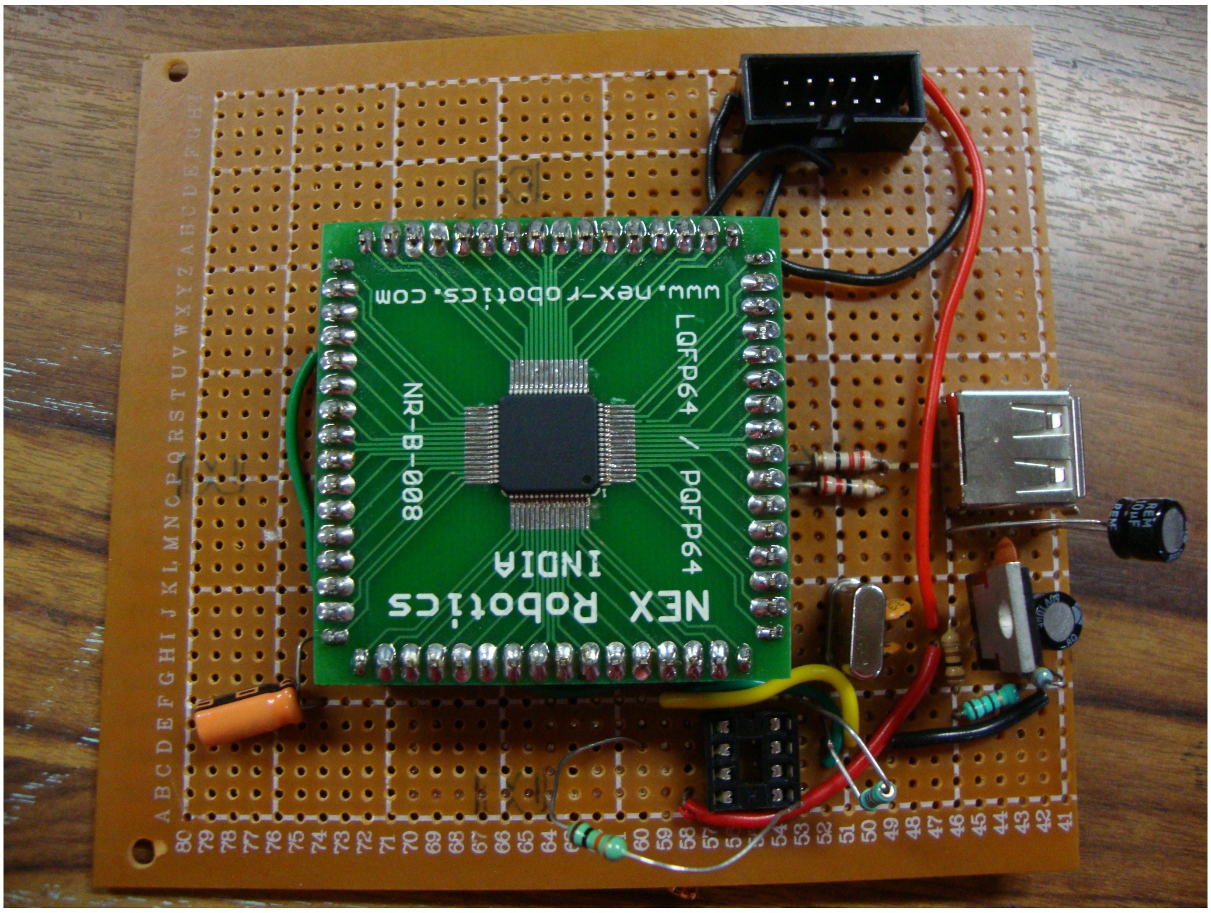

Figure 2: Prototype of JTAG board

2. Atmega128 application board

We needed a target processor which is simple to use, programmable by JTAG interface, does not have practical issues like availability, solderable package, etc., and above all, should be supported by target application, which in our case is urjtag. Atmega series is widely available and very simple to program using other interfaces as well. Currently it is most widely used processor by electronics enthusiasts and we too have some prior experience with series. Importantly urjtag supports just Atmega128 and openOCD none in the series. So we finally bet on Atmega128 for JTAG testing purpose.

2.1 Hardware Design

The board has least number of components and simplest of all. Power to drive circuit will be extracted from JTAG interface through its Vcc-Gnd pins which ultimately will bank on power supplied by USB. All the required Vcc-Gnd pins are connected to corresponding JTAG Port's pin. The JTAG interface has 4 control and data signals,viz, TMS, TCK, TDI, TDO which are connected to respective pins on microcontroller directly. A LED has also been included on pin6 of PORTB for testing and debugging process.

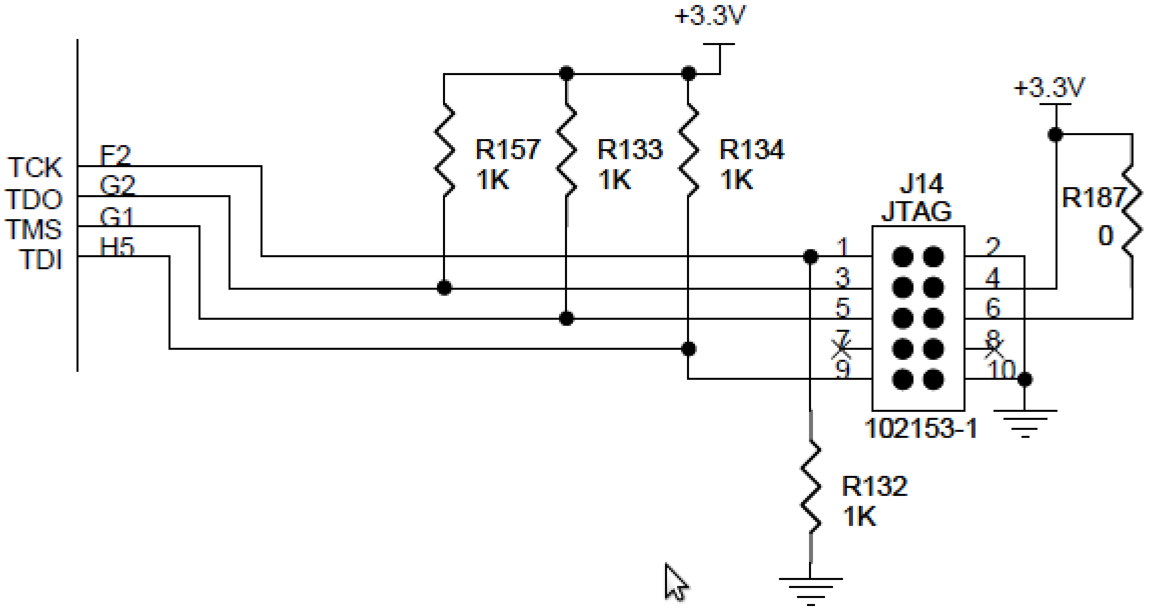

Figure 3: Schematic of Atmega128 application board

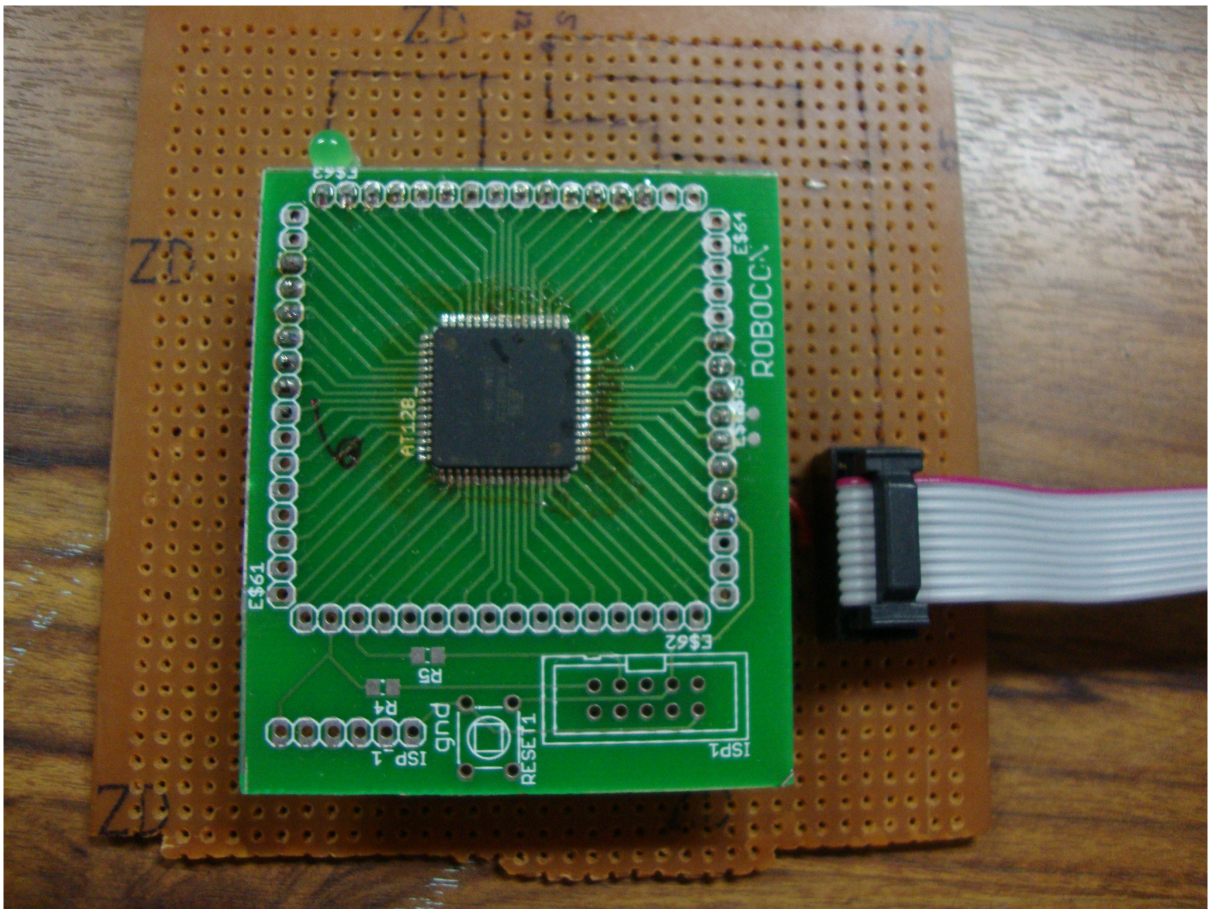

Figure 4: Prototype of Atmega128 Application Board

2.2 Software Design

Urjtag will be used for programming the microcontroller via JTAG. Urjtag has the device support buildup for Atmega128 and hence not much is required from the user side. The programme to be flashed in microcontroller will be generated using Atmel AVR compiler and supporting package avrdude.

3. FPGA Board

As FPGA device is a new concept, not only for our EDL group but most likely for the entire university, we decided upon building a seprate FPGA board with the simple purpose of programming it at the first hand and porting some very simple application to it. This board can be looked back whenever we get struck with the main accelerator card to be build up in next stage.

Choosing a specific FPGA was not a difficult task as there are just 2 major player in the market, Xilinx and Altera. We preferred to go for Altera beacause of its more openess nature, good forum support and excellent academic alliance support. In HPC lab, IIT Bombay, there is a gradual shift happening from Xilinx based designs to Altera based design and we have a new development board arriving any time soon. A mediocre level Cyclone II series device was finally chosen as the series has quite matured and help over the net can reasily be expected. Besides the upcoming development board is also based on Cyclone II series which might act as a good benchmark platform and the applications developed for one can be tested on other. Now given the budget constraint, we went on with a low end device of same series, i.e, EP2C8. It has 8k logic elements, 36 4Kbits RAM blocks, 18 multipliers and 2PLL. The multipliers are 18*18 bits multiplier which can be used as 2 9*9 multiplier. The PLLs with the help of 8 clock signals pins can be used to generate different clock frequencies often required for different part of the application. The device also has 182 I/O pins which can be used for interfacing external RAMs to enhance performance for memory intensive applications. Other pratical considerations were availabilty of chip in non BGA package, per unit cost within INR 1-2K, device support for urjtag and availability in local market or trusted global vendors.

The FPGA board also has an external 32MB SDRAM chip from Micron Technologies having 16 bit wide data or word line. The RAM can be accessed synchronously with an access time of 5.4ns and can be clocked upto 133 Mhz. This memory chip is added just to enhance the performance for memory critical application.

3.1 Hardware Design

The design of PCB is centered around FPGA chip EP2C8. Most of the I/O connection are with external SDRAM. The clock is generated using an external 12MHz crystal and CDCS502,a crystal oscillator based clock generator from TI. This clock signal is fed to both the on-board PLLs. The device is powered from the Vcc-Gnd lines of JTAG port. EP2C8's low voltage differential signal buffer needs 1.2V and hence a low dropout regulator TLV70012-DC from TI is used to get 1.2V from the input 3.3V. A PCB Schematic and Layout was drawn using Eagle cadsoft PCB making CAD software. Though we wanted to do manual routing for entire chip, but due to lower chip count, complexity and tight time constraints, we utlized the auto routing feature in Eagle which often does not give a good result. It was optimized with different DRCs and component placement.

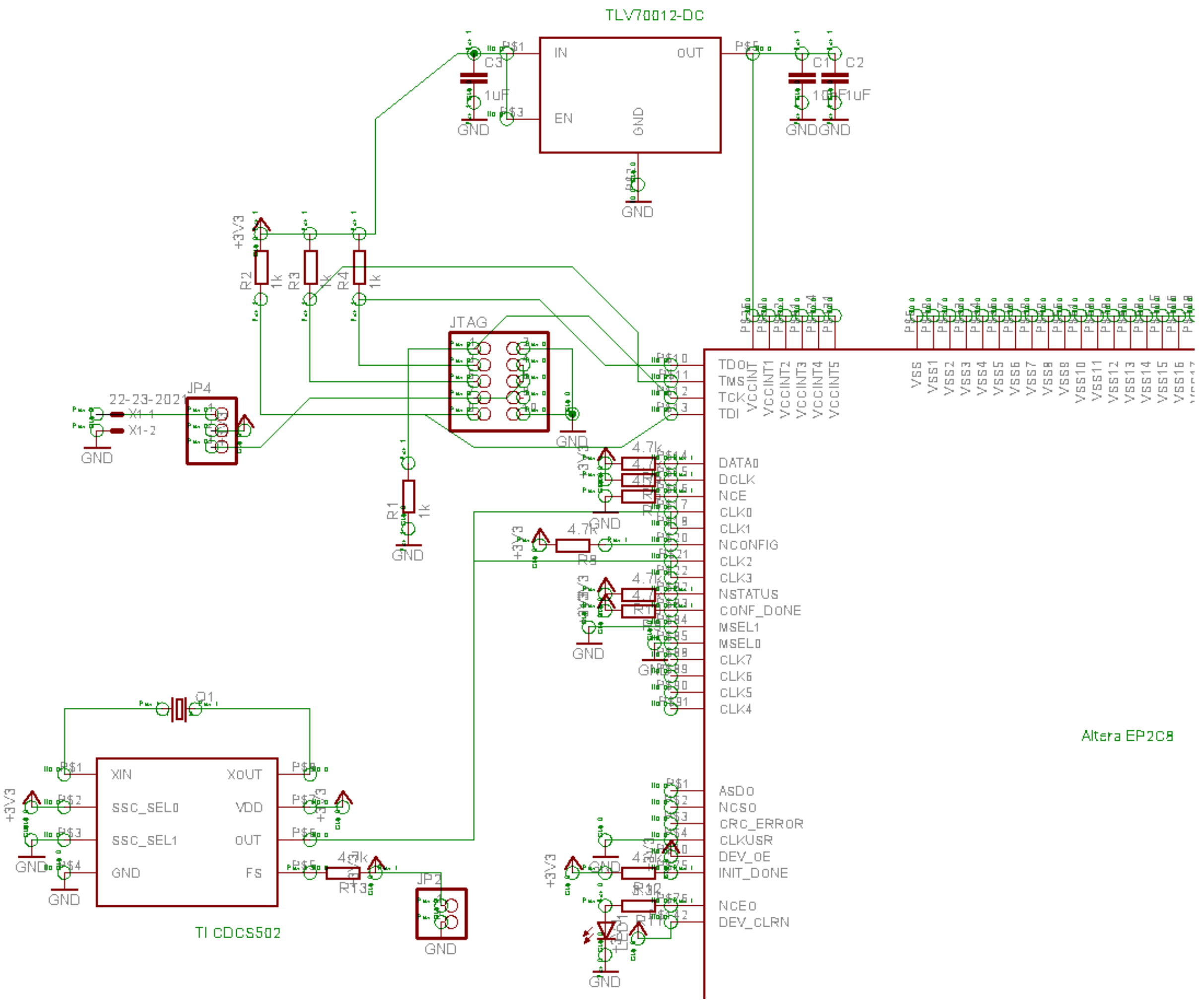

Figure 5: Schematic of FPGA board

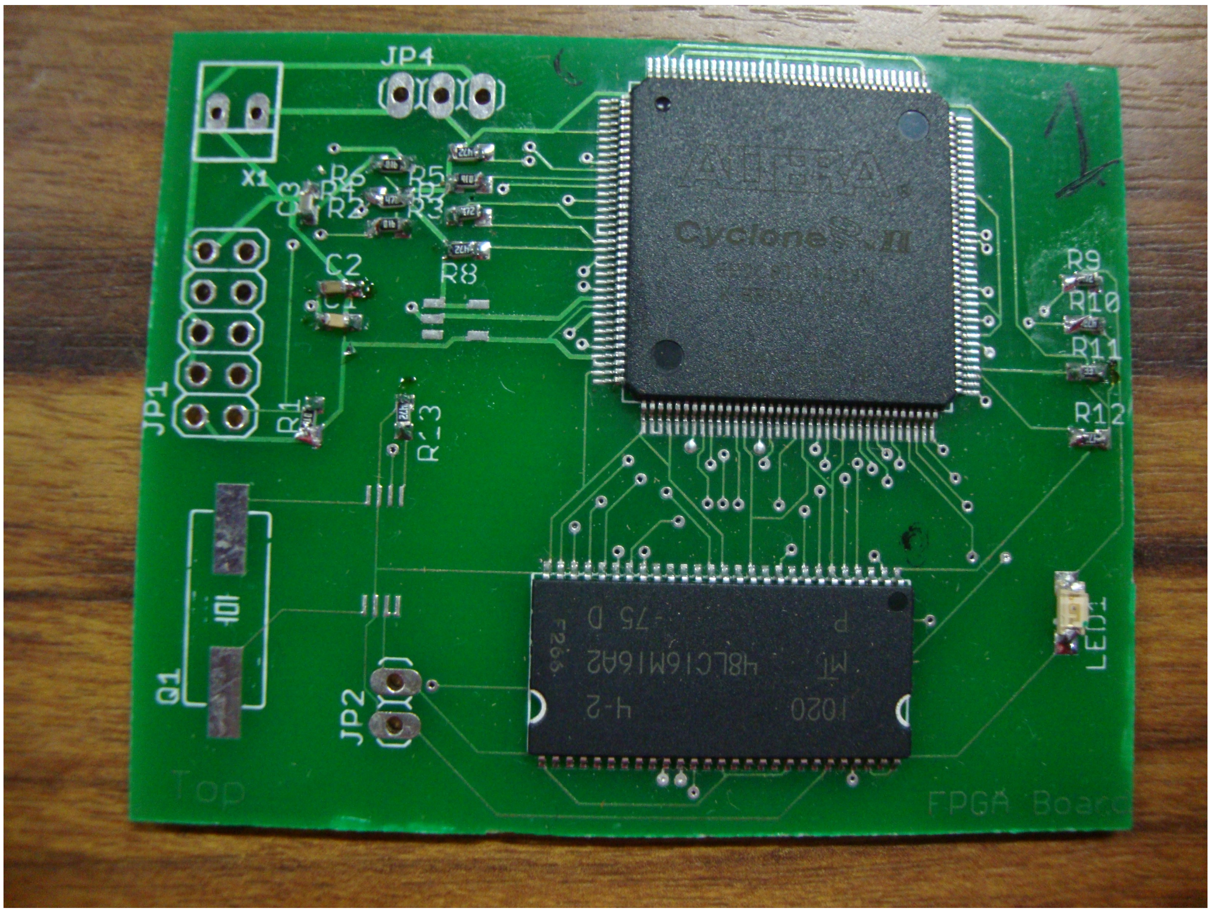

Figure 6: Prototype for FPGA Board

Specification of Layout parameters which is given by PCB manufacturer (Yashna Circuits, Andheri in our case) were

- Minimum width of conducting track – 6mil
- Minimum distance between any conducting track/via/pad – 6mil
- Minimum drill size - 12mil
- Minimum diameter of via/pad - 28mil

3.2 Software Design

A small application will be used to test the workability of programming interface. This application can be written in VHDL or Verilog and then compilation and synthesis can be done using the free web edition of Quartus, Altera's design suite software. A library will also have to be developed for using the external RAM which will follow the required SDRAM protocol.

4. PCI board with microcontroller

The purpose of this board is to get familiarize and establish the PCI communication between the PC and an end device. For testing and debugging purposes, a microcontroller is being chosen as an end device which would be replaced by the FPGA in the later stage.

4.1 Introduction to PCI (Peripheral Component Interconnect)

It is a computer bus with 32-bit width for attaching hardware devices in a computer. Due to its fast speed of data transfer, 33MHz, PCI remains a – protocol in the high performance field to communicate with the external peripherals like FPGA programmed computing devices.

PCI provides separate memory and I/O port address spaces for the x86 processor family, 64 and 32 bits, respectively. Addresses in these address spaces are assigned by software, which is the PCI driver. A third address space, called the PCI Configuration Space, which uses a fixed addressing scheme, allows software to determine the amount of memory and I/O address space needed by each device. The PCI-configuration space is of 256-bytes.

Its signalling can be in either 5V or 3.3V. In a typical system, the firmware (or operating system) queries the PCI buses at startup time (via PCI Configuration Space) to find out what devices are present and what system resources (memory space, I/O space, interrupt lines, etc.) are required by each. It then allocates the resources and tells each device what its allocation is.

We have chosen 32-bit memory address space and 5-V signalling. In the PCI communication we have the PC as initiator and the microcontroller NXP LPC2212 as the target.

4.2 Hardware Design

The PCI control signals and Data lines are connected to the uC from the 2 busses as shown in the schematic in Fig. 5. Components on the board :-

1) NXP LPC2212 :- This is the 32-bit microcontroller chosen as the end device for PCI communication. The device was selected on for following reasons :-
    a) PCI works at 33MHz. Hence any controller working at 33MHz or above would qualify as controller for our PCI board. The maximum clock frequency at which LPC2212 works from the external clock source or crystal using on-chip PLL is 60MHz.
    b) The chosen controller device should be easily programmable with a programmer that can be built in-house and the programming tools should preferably be available for free. Most of the NXP's controller can be programmed using very simple UART protocol using freely available FlashMagic software from NXP. Although we plan to program the controller using JTAG once the JTAG board is tested successfully.

c) One of the teammate had successful past experience using this device.

d) Other features like on-chip cache, flash memory, number of i/o pins were sufficeintly available on this device.

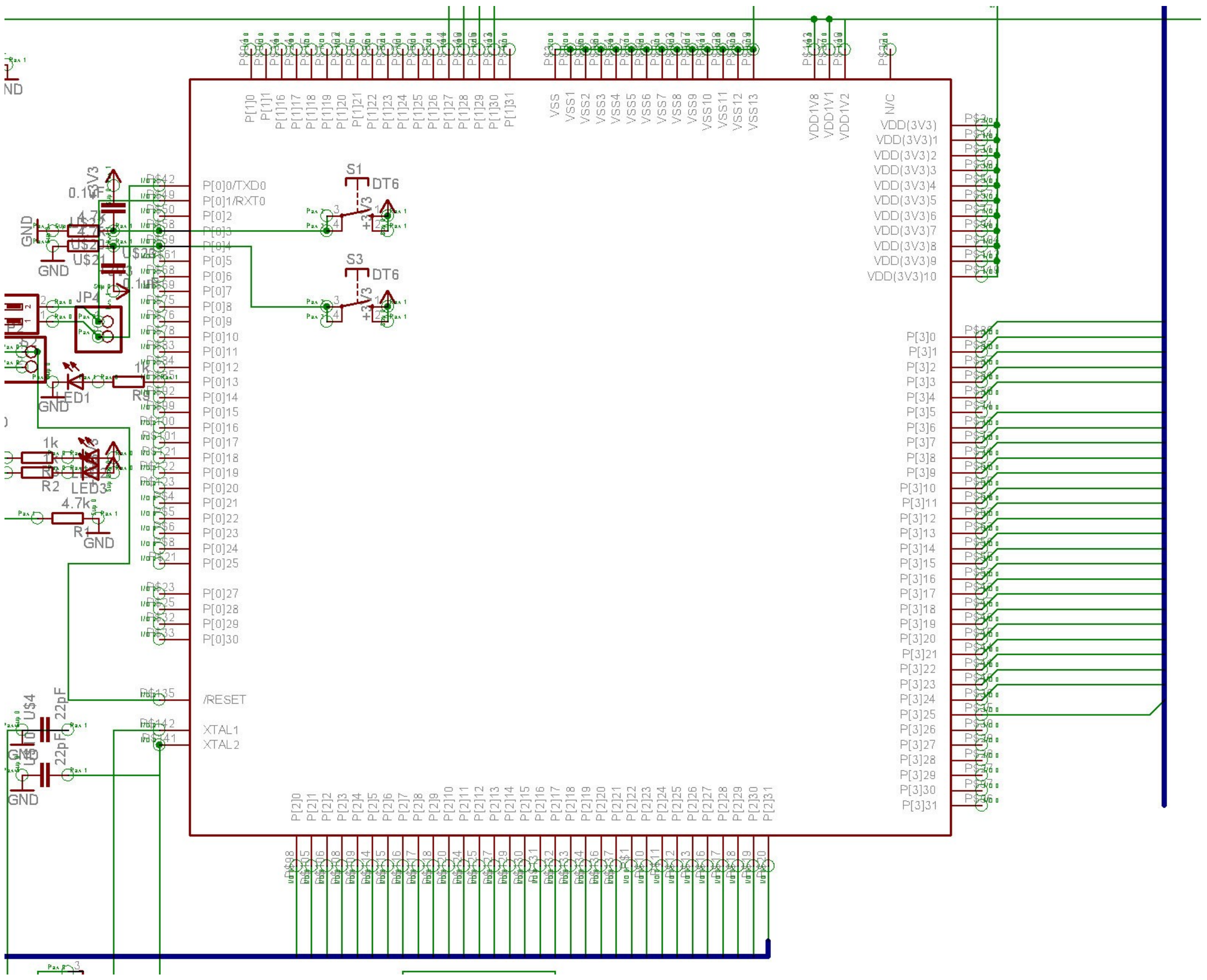

Figure 7. Schematic showing connections around microcontroller, LPC2212

2) FT232 :- We have provided the feature of programming the uC on-board from the RS232 protocol. FT232 is being used for this purpose.

3) LM1117 :- The uC works requires supply of 3.3V as well as 1.8V. Hence this voltage regulator is being used. This was selected due to its stable output as compared to other regulators like LM317.

4) JTAG interfacing :- This is an extension feature on the board to program the uC from JTAG, as shown in Fig. 3.

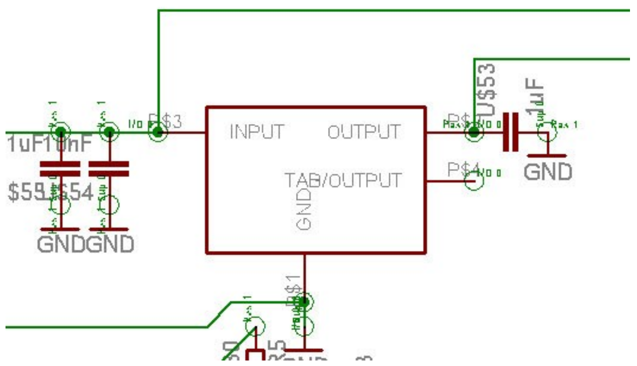

Figure 8. Schematic showing signals aroung voltage regulator, LM1117

5) SN74HC125 :- This buffer is connects the FRAME# signal from the PCI slot to the uC. We have provided an option to either clock the uC through a 12MHz crystal, or through the PC.

   Following are the reasons to use this buffer :-

   a) Non inverting logic was required.

   b) Its rise time and fall time were 9.5ns which is in accordance with the requirement of 15nsec

   c) Supply voltage requirement is within range. (2V-5.5V)

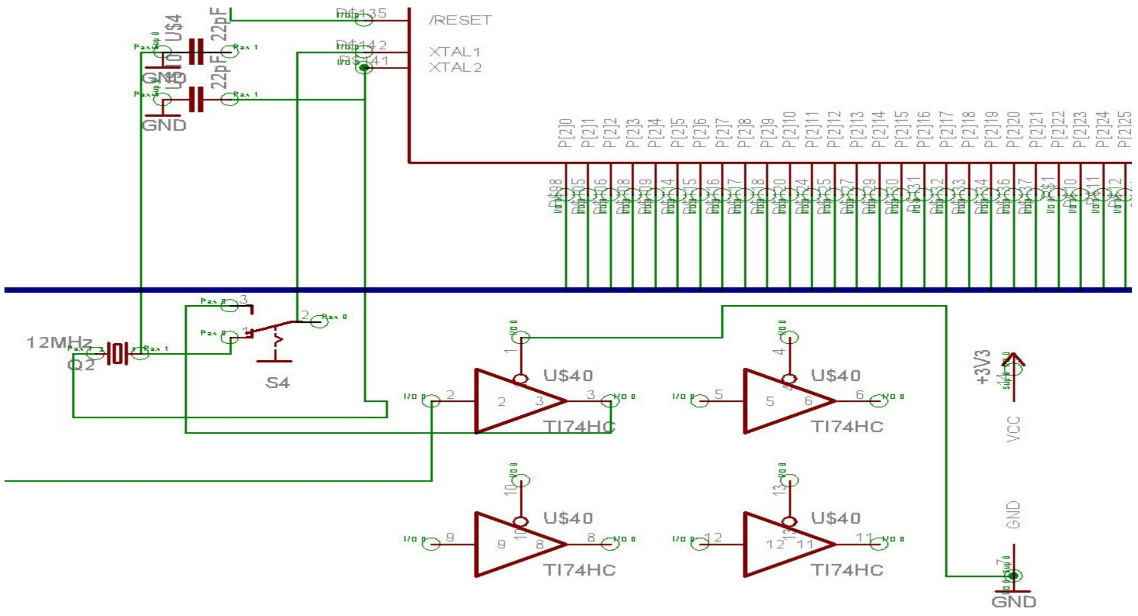

Figure 9. SN74HC125 providing clock to uC from FRAME# signal of PCI

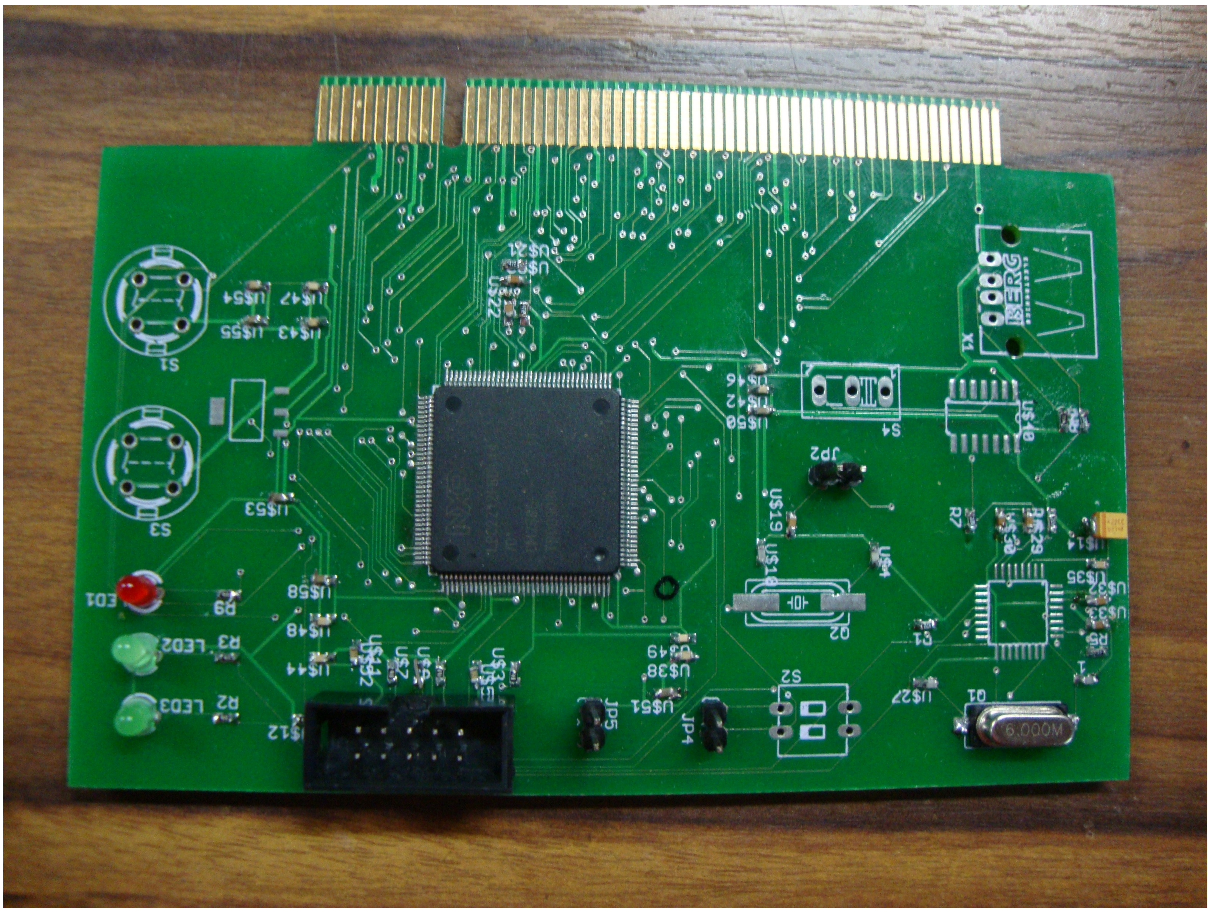

Figure 10: Prototype of PCI board (Not completely populated)

Software Design :-

1) LPC2212 programming: Keil will be used to program the LPC2212 from the Atmega128 programmer. The code is in C and its function would be to behave as a PCI target and assert the necessary signals as and when required. After receiving the R/W command from the initiator, it will also do the operations accordingly.

2) PCI Driver (Plug & Play): This is written in the kernel of the operating system, which is ubuntu 9.10, in our case. It is a platform dependent driver. The function of this driver would be to find if a PCI device is being connected to the PC, and then enable it. After this, it will allocate the configuration space of the PCI device in the PC memory when the device is inserted in PCI slot. Later, it sees the PCI end device as memory or I/O mapped registers to be written or read from. Kernel version used :- 2.6

Stage II

1. PCI Accelerator Card

Our final hardware product would be this PCI Accelerator Card containing FPGA and external SDRAM(if time permits). This will involve logically combining all the blocks discussed in stage I. The card will also have an USB device port for configuring the FPGA through JTAG. To avoid configuration on every power booting of board, we plan to include external flash/EEPROM from where the configuration data can be loaded on every power boot of device.

All the softwares, be it driver, library, application software or CAD tool are either open sourced or freely available for non-commercial use or have been build by us.

2. CFD Kernel Demo Software

Though any potentially parallelizable application could be ported to FPGA and speedup can be demostrated, we chose a simple 2D Euler CDF Kernel to be implemented on FPGA. A supplementary reason is Nikhil's seminar topic being “Hardware parallelization of CFD Kernel” under guidance of Prof. S. Patkar.

The CFD software can be separated into parts, the control part and inner loops. The control part comprises of everything else the inner loops including variable declaration, header initialization, loop control, conditional statements, etc. This part of code is good for cpu as instructions are required to operate sequentially. The contents inside inner loop generally deals with solving the N-S or 2D Euler equation at a node given the values for neighbouring nodes are known. These codes often run for a few thousand to million times for a single programme execution depending upon grid structure and is the most time consuming step of programme. Speedup can be obtained by implenting the N-S or 2D Euler euation on hardware which can operate within a very low number of cycle. More greedly these hardware programs can be multiplied many times and executed concurrently till we run out of resource on FPGA/ASIC. The result of each step can be locally stored on RAM blocks within FPGA locally and data with cpu can be exchanged on demand using the PCI interface. If external memory is present, the need for frequent communication with cpu can further be brought down.

Status of the project:

JTAG Board:
Design , manufacturing, testing and debugging was done.
The board is working successfully.

ATMEGA128 Board
Design , manufacturing, testing and debugging was done.
We established successful communication wth the JTAG Board.

FPGA Board:
Completed the design and manufacturing of the board.
We were not able to debug the system correctly.

PCI Card:
1.Completion of design, manufacturing and component population of the pci end controller was done
2. The driver development for PCI on pc's side was done
3. Program for lower level interface signals generation from end controller

Future Works:

1.Hardware program of a CFD kernel to be ported on FPGA and supporting serial programme for cpu.
2. Integration of PCI, FPGA and JTAG on a single board could not be done.
We shall provide the interface between FPGA and an external SRAM.

References

1.http://en.wikipedia.org/wiki/Gravity_Pipe
2.http://www.sc2000.org/bell/pastawrd.htm
3.http://www.research.ibm.com/grape/grape_mdgrape2.htm
4.Datasheets for following chips:
- FT2232H
- Atmega128
- Altera's EP2C8
- Micron Tech's MT48LC16M16A2
- TI's CDCS502
- TI's TLV70012-DC
- NXP's LPC2212
- FT232BL
- TI's TLV1117-18

5.Web sources :
- http://en.wikipedia.org/wiki/Conventional_PCI
- http://www.fpga4fun.com/PCI.html
- http://electrofriends.com/articles/computer-science/protocol/introduction-to-pci-protocol/

6.Reference Projects
- Abhishek Kamath's Aletra's Max CPLD Board, RA, WEL1-2, IIT Bombay
- Darrel Harmon's Single Board Computer (http://dlharmon.com/sbc.html)
- Darrel Harmon's FPGA Board (http://dlharmon.com/fpgacard)

7.Reference Books
- “Linux Device Drivers” by Alessandro Rubini