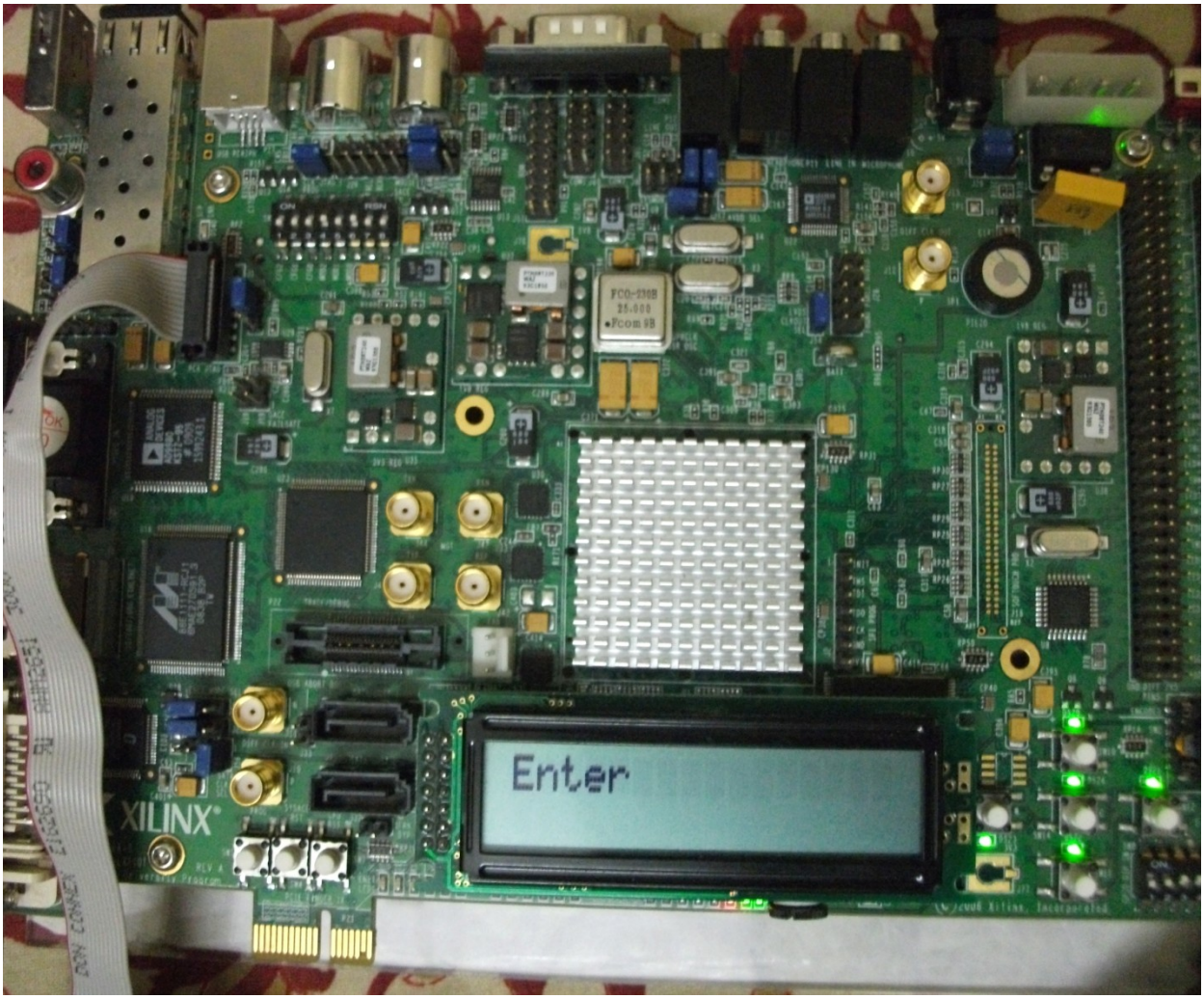
FPGA

(Field Programmable Gate Array)



Team Name:
Electromarvels

Team Members:
Megha Nawhal
Jyotsna Soni
Vijaya Rani

Acknowledgement

Apart from the efforts of me, the success of any project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project.

I would like to show my greatest appreciation to our senior Sushobhan Nayak. Without his encouragement and guidance this project would not have materialized.

The guidance and support received from all the coordinators of our E-Club who contributed and who are contributing to this project, was vital for the success of the project. I am grateful for their constant support and help.

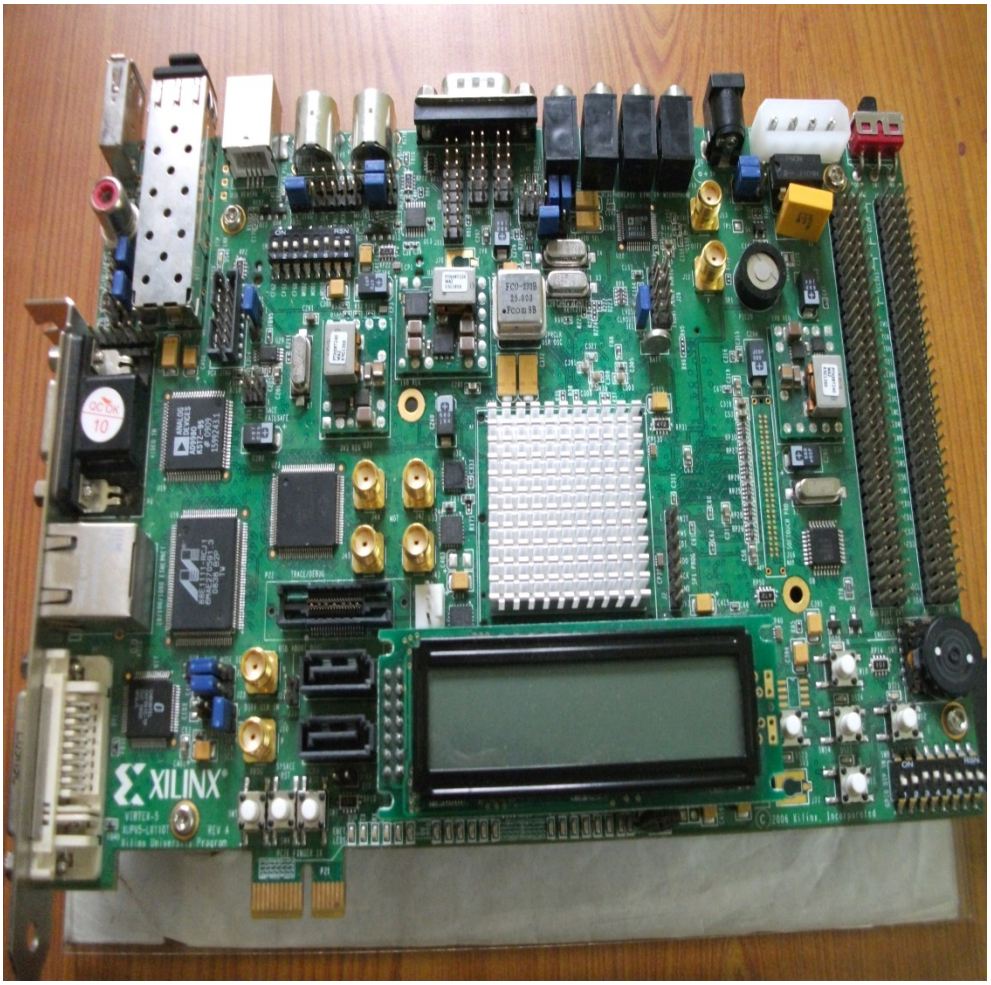
Our Project

This project was an experimental project on FPGAs to learn the application of it in various tasks.

As we have already worked with microcontrollers, we tried to explore other devices which can be used for various purposes in the field of electronics. We were very much impressed by the advantages of FPGA and wanted to know more about its diverse applications.

We tried two things in our project .First, to implement OpenSPARCT1 on our FPGA and design a SDLX processor and carry out some basic mathematical operations using the designed processor.

The below is the picture of the FPGA board which we used in our project.



XILINX VIRTEX-5 XC5VLX110T FPGA BOARD

FPGA Basics

The FPGA is an integrated circuit that contains many (64 to over 10,000) identical logic cells that can be viewed as standard components. The individual logic cells are interconnected by a matrix of wires and programmable switches. A user's design is implemented by specifying the simple logic function for each cell and selectively closing the switches in the interconnect matrix. The array of logic cells and interconnects form a fabric of basic building blocks for logic circuits. Complex designs are created by combining these basic blocks to create the desired circuit.

Field Programmable means that the FPGA's function is defined by a user's program rather than by the manufacturer of the device. A typical integrated circuit performs a particular function defined at the time of manufacture. In contrast, the FPGA's function is defined by a program written by someone other than the device manufacturer. Depending on the particular device, the program is either 'burned' in permanently or semi-permanently as part of a board assembly process, or is loaded from an external memory each time the device is powered up. This user programmability gives the user access to complex integrated designs without the high engineering costs associated with application specific integrated circuits.

The task of individually defining the many switch connections and cell logic functions is handled by special software and the programs are written in a Hardware Description Language (HDL).The software translates a user's schematic diagrams or textual hardware description language code then places and routes the translated design. Most of the software packages have hooks to allow the user to influence implementation, placement and routing to obtain better performance and utilization of the device. (The software we used was Xilinx ISE Design Suite and programs were written in verilog)

Three major vendors provide FPGA hardware—Xilinx Inc., Altera Corp. and Lattice Semiconductor Corp.—as do a host of others. The use of FPGA technology has been supported by increasingly available design tools and modules, many of them application-specific.

FPGA Vs MICROCONTROLLERS:

- FPGA is mainly for programmable logic but microcontroller is mainly for hardcore processing.
- Microcontroller is running sequentially regardless of how fast the controller is. In digital signal processor, the hardcore would enhance the harward architecture by increasing pipelining to certain level of parallel instruction processing. Instead, FPGA is totally hardware based programmable. The parallel processing in FPGA is not depends to the pipelining, but it is hardware based parallel architecture.
- In FPGAs, program memory, ram memory and input/output resources are external to the chip but in case of microcontrollers these are incorporated internally.

OpenSPARC

OpenSPARC *is an open-source hardware project started in December 2005 by Sun Micro systems. The full OpenSPARC T1 system consists of 8 cores, each one capable to execute 4 threads concurrently, for a total of 32 threads. Each core executes instruction in order and its logic is split among 6 pipeline stages.*

FEATURES OF OPENSPARCT1:

The OpenSPARC T1 processor features include the following:

CPU

- SPARC V9 Architecture
- On-chip L2-cache
- 48-bit virtual, 40-bit physical address space

Caches

- 16 Kbyte primary instruction cache per core
- Parity
- 8 Kbyte primary data cache per core
- Parity
- 3 Mbyte unified L2-cache
- Error correction code (ECC)
- 12-way set-associative, 4 banks

Integration

- 8 cores, 4 threads per core
- Four 144-bit DDR2-533 SDRAM interfaces
- Quad error correction, octal error detection, chip kill ECC
- Optional 2-channel operation mode
- J-Bus Interface
- 2.56 Gbyte/sec peak effective bandwidth
- 128-bit address or data bus
- 150–200 MHz operation

Installation Instructions for OpenSPARC T1 design and verification database

1. Download OpenSPARCT1.tar.bz2 file to your directory.

2. Unzip downloaded file by using following command:

   bunzip2 OpenSPARCT1.tar.bz2

   This step will create OpenSPARCT1.tar file.

3. Extract files from tar file by using following command:

   tar xvf OpenSPARCT1.tar

4. Setup environment variables by editing OpenSPARCT1.cshrc file.

5. Source the environment variable file.

6. The OpenSPARC T1 Design/Verification package now comes with 3 environments: thread1, core1 and chip8.

   - The thread1 environment consists of single thread implementation of one SPARC CPU core (without Stream Processing Unit), Cache, Memory, Crossbar etc. without I/O sub-system.

   - The core1 environment consists of one SPARC CPU core, Cache, Memory, Crossbar etc. without I/O sub-system.

   - The chip8 environment consists of full OpenSPARC T1 chip including all 8 cores, Cache, Memory, Crossbar, I/O sub-system.

   Each environment has mini regression and full regression.

SDLX Processor

INTRODUCTION:

The DLX is a RISC processor architecture designed by John L. Hennessy and David A. Patterson, the principal designers of the MIPS and the Berkeley RISC designs (respectively),the two benchmark examples of RISC design.RISC design is a CPU design strategy based on the insight that simplified instructions can provide higher performance if this simplicity enables much faster execution of each instruction .

The SDLX processor is a 32-bit architecture. It can perform operations on 32-bit data in the most natural manner. For example, the SDLX processor can add two 32-bit numbers using a single instruction. All instructions in the SDLX processor are 32-bit (4 bytes) wide. The SDLX processor implements only the integer operations on the data values. The floating-point instructions are not used in the SDLX. Complex integer instructions such as mul or div are also excluded from the instruction set of the SDLX.

SDLX has 32-bit memory addressing capabilities. It supports byte addressable memory. Thus the size of the address space of the SDLX processor is 32 bytes. The operands larger than one byte are stored in multiple memory locations. The order of storage in such cases is big-endian. The most significant byte is stored first in the lower address. The least significant byte is stored last in the higher address.

SDLX REGISTER SET:

- The SDLX processor has 32 general-purpose registers, each 32-bit wide in size. Registers are named R0 to R31.
- Register R0 is hardwired to 0. A value that can ever be read from this register is 0. This register can never be used to hold any value as a value written in this register is lost.
- Register R31 of this processor has one special function in addition to the usual use. This register can be used as link register.
- The processor has one 30-bit program counter register (PC). It is used to read the instructions from the memory.

The two least significant bits of the instruction addresses are always 0. Therefore it is enough to store just 30-bits of the instruction address in the PC register. The PC register therefore stores instruction address divided by 4.

SDLX INSTRUCTIONS:

SDLX instructions are categorized in the following three categories.

- R-type instructions are pure register instructions, with three register references contained in the 32-bit register.

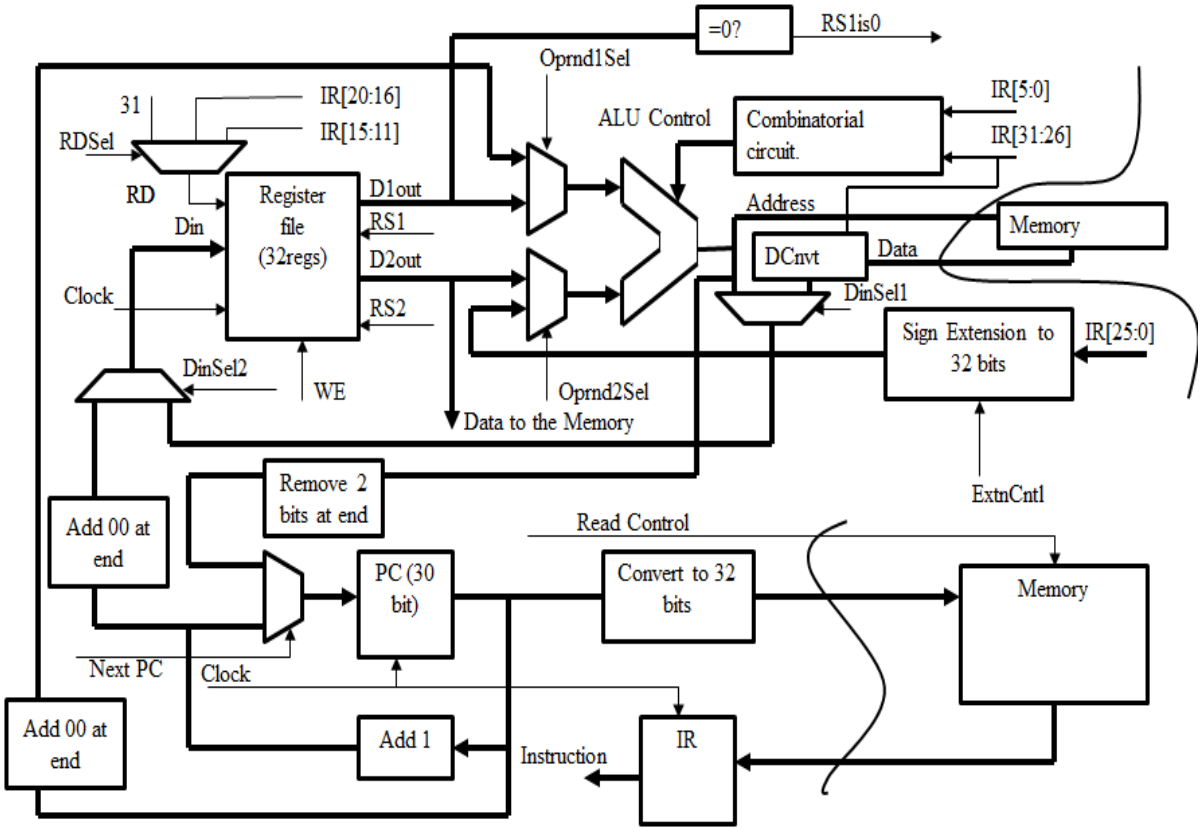
- I-type instructions specify two registers, and use 16 bits to hold an immediate value.
- J-type instructions are jumps, containing a 26-bit address.

INSTRUCTION FORMATS:

Format	Bits					
	31:26	25:21	20:16	15:11	10:6	5:0
R-type	0x0	RS1	RS2	RD	*unused*	*opcode*
I-type	*opcode*	RS1	RD	*immediate*		
J-type	*opcode*	*value*				

Table 4.1: Types of Instructions

SDLX INSTRUCTION EXECUTION DATAPATH:



In the execution of the SDLX instructions, there may be up to three registers involved where two of them are read and the third one is updated. Therefore, a register file in the SDLX must have at least three ports (two read and one write). For registers we shall assume a circuit based on the flip-flops that use a positive edge of the clock to load the value. A 32-bit ALU in the SDLX is used to perform the operations on the input operands and to produce the output.

The processor has a five stage instruction pipeline .The main steps involved are listed as follows:

- On every positive edge on the clock input, which signifies the beginning of a new instruction cycle, the instruction is first read from memory. The address of the instruction is provided by the Program Counter (PC).
- After the fetch, the PC is incremented by the size of the instruction, and the instruction is stored in the Instruction Register (IR).
- Subsequently, the instruction is executed on the basis of the instruction format.
- Memory address mentioned in the IR is accessed and then finally the output is written on the destination register of the IR.

This was a brief introduction of the SDLX Processor and once the processor gets implemented we can make changes in the program memory and data memory module and can perform different tasks such as displaying any text on LCD or making a calculator or displaying fibonacci series .

Completion of our project

Our first step was to get accustomed with the software Xilinx ISE Design Suite and for that we wrote basic modules in verilog and implemented on our FPGA board .When we were clear with the basics we started with the implementation of the OpenSPARCT1 processor o our board . We were successful in booting Open Solaris but the operating system did not have any firm wares (drivers) installed so we could not attach any external device to our board and use it .We also tried booting Linux but due to certain reasons we were not able to do that.
Next, we started working on DLX Processor and had the target of displaying fibonacci numbers on the lcd along with the user interface with the board . We faced some issues on creating the user interface with the board and finally we were successful in doing so.