IMPLEMENTATION OF MICROPROCESSOR AND DIGITAL WAVEFORM GENERATOR ON FPGA USING VERILOG HARDWARE DESCRIPTION LANGUAGE

A REPORT SUBMITTED TO

RAJA RAMANNA CENTRE FOR ADVANCED TECHNOLOGY, INDORE

SUBMITTED BY

MEHER ADBOL

(B. E. 4TH YEAR, ELECTRONICS AND INSTRUMENTATION ENGINEERING, IET DAVV, Indore)

Under The Supervision of:

Dr. M. S. ANSARI

(SO/H, HEAD, LPSD, RRCAT Indore)

ACKNOWLEDGMENT

I take this opportunity to express my sincere and heartfelt gratitude to **Dr. M. S. Ansari**, Scientific Officer/H, Head, Laser Power Supplies Division (LPSD), RRCAT Indore for insight practical knowledge and his constant support in my project work. In spite of heavy commitment and busy schedule, he had been there for me with his invaluable guidance and support to help me. His persistent encouragement, perpetual motivation, everlasting patience and valuable technical inputs during the progress of my project work, have benefited me to an extent, which is beyond expression.

My gratitude will not be complete without mention of all the members of Laser Power Supplies Division for their support, encouragement and giving me the opportunity for this project work.

I also wish to thank he Project Placement Committee of RRCAT and Institute of Engineering and Technology for giving me this opportunity to undertake my internship at Laser Power Supplies Division.

I also thank other sir / madam and officers for giving me their valuable suggestions.

I sincerely wish to express my gratefulness to all the staff members, my family and my friends who have been a source of support and guidance, directly or indirectly for me.

Finally, I thank the almighty God who has been my guardian and a source of strength and hope in this period.

Meher Adbol

DECLARATION

Meher Adbol, a student of Bachelor in Engineering from the Department of Electronics and Instrumentation Engineering, **Institute of Engineering & Technology, DAVV, Indore (MP)**, hereby declare that the work presented in the report entitled ***" IMPLEMENTATION OF MICROPROCESSOR AND DIGITAL WAVEFORM GENERATOR ON FPGA USING VERILOG HARDWARE DESCRIPTION LANGUAGE "*** is carried out by me at Raja Ramanna Centre for Advanced Technology, Indore under the supervision of **Dr M. S. Ansari, Scientific Officer/H, Head, Laser Power Supplies Division (LPSD), RRCAT Indore.**

Meher Adbol
Institute of Engineering & Technology,
DAVV, Indore (MP)

सत्यमेव जयते
Goverment Of India

राष्ट्र की सेवा में परमाणु
राराप्रौके
RRCAT
ATOMS IN THE SERVICE OF THE NATION

GOVERNMENT OF INDIA
DEPARTMENT OF ATOMIC ENERGY
RAJA RAMANNA CENTRE FOR ADVANCED TECHNOLOGY
INDORE (M.P.) 452 013

CERTIFICATE

This is to certify that **Mr. Meher Adbol** a student of fourth year Bachelor in Engineering at Institute of Engineering & Technology, DAVV, Indore has carried out internship training at Laser Power Supplies Division, Raja Ramanna Centre for Advanced Technology (RRCAT), Indore during the period from 1st June 2022 to 18th August 2022 and completed the project **"Implementation of microprocessor and digital waveform generator on FPGA using Verilog Hardware Description Language".** We recommend this work towards the partial fulfilment for the award of the degree of Bachelor in engineering at IET, DAVV. We wish Meher Adbol a successful professional career.

(Dr. M. S. Ansari)
Head LPSD, RRCAT

Dated:

CONTENTS:

List of Figures

List of Tables

Chapter 1

Abstract:

This report discusses the design of a simple microprocessor and a digital waveform generator on FPGA board using Verilog HDL. Microprocessors are found in almost every electronic device that is available today. The microprocessor contains the arithmetic, logic, and control circuitry required to perform the functions of a computer's central processing unit. Microprocessors and microcontrollers enabled the modern world to produce devices that make the life of people easy and going. FPGAs are widely used in the semiconductor industry to design, test and verify integrated circuits before the final manufacturing. The microprocessor in this project has an 8-bit Arithmetic & Logic Unit (ALU) which is capable of performing operations on 8-bit inputs and producing 8-bit outputs and a carry (if calculation results in one). The processor is also capable of storing the inputs, outputs and control signals in the registers designed. The microprocessor hardware and architecture are designed using Verilog HDL and implemented on FPGA. The digital waveform generator is capable of generating square waves with variable duty cycles, saw-tooth waves and triangular wave which are obtained as 8-bit digital signals. These 8 bit signals can be converted to analogue signal using 8-bit Digital-to-Analog Converter IC and can be observes using Digital Storage Oscilloscope. During this project duration I also got hands-on experience with Programmable Logic Controllers (PLC) and Supervisory Control and Data Acquisition (SCADA) system which is also mentioned further in the report.

Keywords: FPGA, microprocessor, Verilog, HDL, ALU, registers, digital waveform generator, IC, PLC, SCADA

Introduction:

The work presented in this report describes the application of digital circuit design using Verilog HDL. It is focused on designing of a simple 8-bit microprocessor and a digital waveform generator on FPGA board. A Microprocessor is a computer processor where the data processing logic and control is included on a single integrated circuit, or a small number of integrated circuits. The microprocessor contains the arithmetic, logic, and control circuitry required to perform the functions of a computer's central processing unit. The integrated circuit is capable of interpreting and executing program instructions and performing a number of arithmetic operations. The microprocessor is a multipurpose, clock-driven, register-based, digital integrated circuit that accepts binary data as input, processes it according to instructions stored in its memory, and provides results (also in binary form) as output. Hardware Description Language (HDL) is a specialized language used to describe the structure and behaviour of the digital circuits. FPGAs are pre-fabricated silicon devices that can be electrically programmed in almost a digital circuit. FPGAs are computing platforms that offer re-configurability. These are reprogrammable when required in the field. FPGAs need specialized programming tools such as Xilinx Vivado design suite, Intel Altera Compiler, etc.

FPGA design flow requires synthesis and implementation (place and route) that at times is a time consuming process based on the size of design. The microprocessor is a multipurpose, clock-driven, register-based, digital integrated circuit that accepts binary data as input, processes it according to instructions stored in its memory, and provides results (also in binary form) as output. Microprocessors contain both combinational logic and sequential digital logic, and operate on numbers and symbols represented in the binary number system.

FPGAs provide a number of advantages over Application Specific Integrated Circuits (ASICs). ASIC generally take months to fabricate and are expensive. FPGAs are configured within days and can be reconfigured if required and are cost effective than ASICs.

Important highlights of this project are:

(a) Design of digital circuits using Verilog HDL
(b) Design of ALU
(c) Design of Microprocessor
(d) Testing and verification of Microprocessor
(e) Digital Waveform Generation
(f) Hands-on experience of PLC and SCADA

Chapter 2

FPGA

2.1. Introduction to FPGA:

Field Programmable Gate Arrays (FPGAs) are digital ICs (Integrated Circuits) that enable the hardware design engineer to program a customized Digital Logic as per his/her requirements. The term "Field Programmable" implies that the Digital Logic of the IC is not fixed during its manufacturing (or fabrication) but rather is programmed by the end-user (designer).

Field Programmable Gate Arrays or FPGAs in short are pre-fabricated Silicon devices that consist of a matrix of reconfigurable logic circuitry and programmable interconnects arranged in a two-dimensional array. The programmable Logic Cells can be configured to perform any digital function and the programmable interconnect (or switches) provide the connections among different logic cells.

Using an FPGA, you can implement any custom design by specifying the logic or function of each logic block and setting the connection of each programmable switch. Since this process of designing a custom circuit is done in the field rather than in a fab, the device is known as "Field Programmable".

FPGAs have revolutionized the way prototyping and designing is done in the world. The flexibility offered by reprogrammable FPGAs has enhanced the design process.

2.2. FPGA Basics:

FPGAs are ICs that contain an array of identical logic blocks with programmable interconnections. The user can program the functions realized by each logic block and the connections between the blocks. FPGAs are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. FPGAs can be reprogrammed to desired application or functionality requirements after manufacturing. This feature distinguishes FPGAs from Application Specific Integrated Circuits (ASICs), which are custom manufactured for specific design tasks. Although one-time programmable (OTP) FPGAs are available, the dominant types are SRAM based which can be reprogrammed as the design evolves.

Components of an FPGA

Let us now take a closer look at the structure of an FPGA. Typically, an FPGA consists of three basic components. They are:

- Programmable Logic Cells (or Logic Blocks) – responsible for implementing the core logic functions.
- Programmable Routing – responsible for connecting the Logic Blocks.
- IO Blocks – which are connected to the Logic Blocks through the routing and help to make external connections.

Logic Block

The Logic Block in Xilinx based FPGAs are called as Configurable Logic Blocks or CLB. A CLB is the basic component of an FPGA, which provides both the logic and storage functionalities. The basic logic block can be anything like a transistor, a NAND gate, Multiplexors, Look-up Table (LUT), a PAL like structure or even a processor. Both Xilinx and Altera use Look-up Table (LUT) based logic blocks to implement the logic as well as the storage functionalities.

A Logic Block can be made up of a single Basic Logic Element or a set of interconnected Basic Logic Elements, where a Basic Logic Element is a combination of a Look-up table (which is in turn made up of SRAM and Multiplexors) and a Flip-flop. The logic blocks in this type of FPGA are organized in a matrix-like fashion, as illustrated in Figure. Most Xilinx FPGAs belong to this category.

Basic Logic element

A LUT with 'n' inputs consist of 2n configuration bits, which are implemented by SRAM Cells. Using these 2n SRAM Bits, the LUT can be configured to implement any logical function. Many look-up-table–based FPGAs use a 4-variable look-up table plus a flip-flop as the basic element and then combine several of them in various topologies. The 4-variable look-up tables is often denoted by the short form LUT4. The LUT4 can also be called a 4-variable function generator since it can generate any function of four variables.

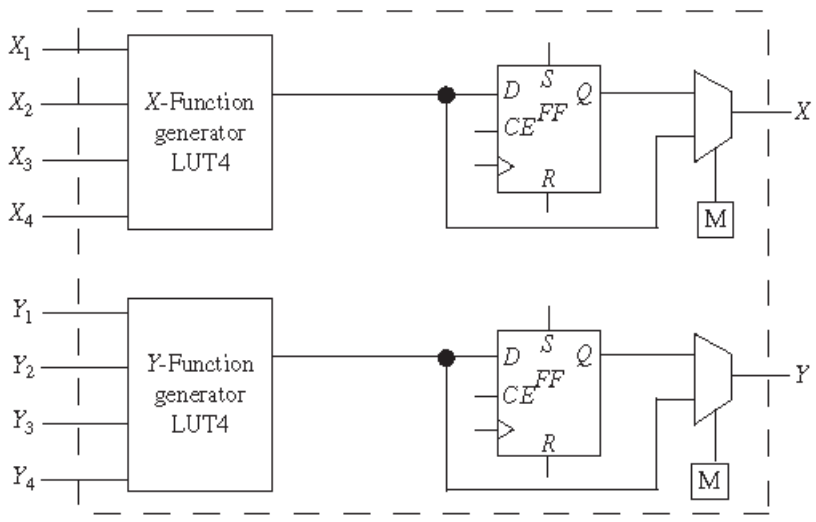


Figure 2.1: Logic Element Example

Consider the Figure 2.1. There are two 4-variable look-up tables and two flip-flops in this programmable logic block. The two LUT4s can generate any two functions of four variables. The inputs to the *X*-function generator are called *X1*, *X2*, *X3*, and *X4*, and the inputs to the *Y*-function generator are called *Y1*, *Y2*, *Y3*, and *Y4*. The functions can be steered to the output of the block (*X* and *Y*) in combinational or latched form. There are two D flip-flops in the logic block. The D flip-flops are versatile in the sense that they have clock enable, direct set, and direct reset inputs. A multiplexer selects between the combinatorial output and the latched version of the output.

Routing

If the computational functionality is provided by the Logic Blocks, then the programmable routing network is responsible for interconnection these logic blocks. The Routing Network provides interconnections between one logic block to other as well as between the logic block and the IO Block to completely implement a custom circuit.

Basically, the routing network consists of connecting wires with programmable switches, which can be configured using any of the programming technologies.

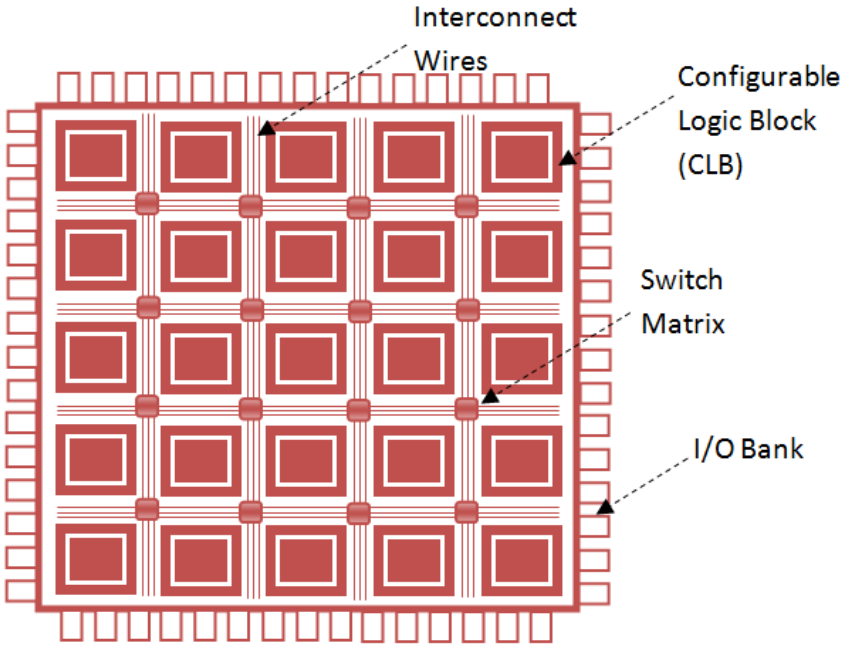


Figure 2.2: FPGA Architecture

FPGAs are designed to be configured by a customer or a designer after manufacturing – hence the term field-programmable. FPGAs contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects allowing blocks to be wired together. Logic blocks can be configured to perform complex combinational functions, or act as simple logic gates like AND and XOR. In most FPGAs, logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory. Many FPGAs can be reprogrammed to implement different logic functions, allowing flexible reconfigurable computing as performed in computer software.

FPGAs are extremely versatile. They allow developers to test any number of variables after the board is built. When changes are required, new configuration files are transferred onto the device that make new functionality available. This flexibility allows OEMs to ship systems earlier in the design process. Developers design prototypes on FPGA to incrementally mature the design before finalizing it at tape-out. FPGAs are often used in commercial applications where there's a need for parallel computing and the requirements are dynamic, such as for telecoms and avionics.

Table 2.1

Customer Benefits by Using FPGAs

Benefit	Details
Flexibility	• FPGA functionality can change upon every power-up of the device. So, when a design engineer wants to make a change, they can simply download a new configuration file into the device and try out the change. • Often, changes can be made to the FPGA without making costly PC board changes. • ASSPs and ASICs have fixed hardware functionality that can't be changed without great cost and time.
Acceleration	Get products to market quicker and/or increase your system performance. • FPGAs are sold "off the shelf" vs. ASICs (which require manufacturing cycles taking many months). • Because of FPGA flexibility, OEMs can ship systems as soon as the design is working and tested. • FPGAs provide off-load and acceleration functions to CPUs, effectively speeding up the entire system performance.
Integration	Today's FPGAs include on-die processors, transceiver I/O's at 28 Gbps (or faster), RAM blocks, DSP engines, and more. More functions within the FPGA mean fewer devices on the circuit board, increasing reliability by reducing the number of device failures.
Total Cost of Ownership (TCO)	• While ASICs may cost less per unit than an equivalent FPGA, building them requires a non-recurring expense (NRE), expensive software tools, specialized design teams, and long manufacturing cycles. • Intel FPGAs support long lifecycles (15 years or more), avoiding the cost of redesigning and requalifying OEM production equipment if one of the electronic devices on-board goes end of life (EOL). • FPGAs reduce risk, allowing prototype systems to ship to customers for field trials, while still providing the ability to make changes quickly before ramping to volume production.

2.3. Specifications of FPGAs used in the project:

1. Artix-7 series based BASYS 3 FPGA board is used for this project.

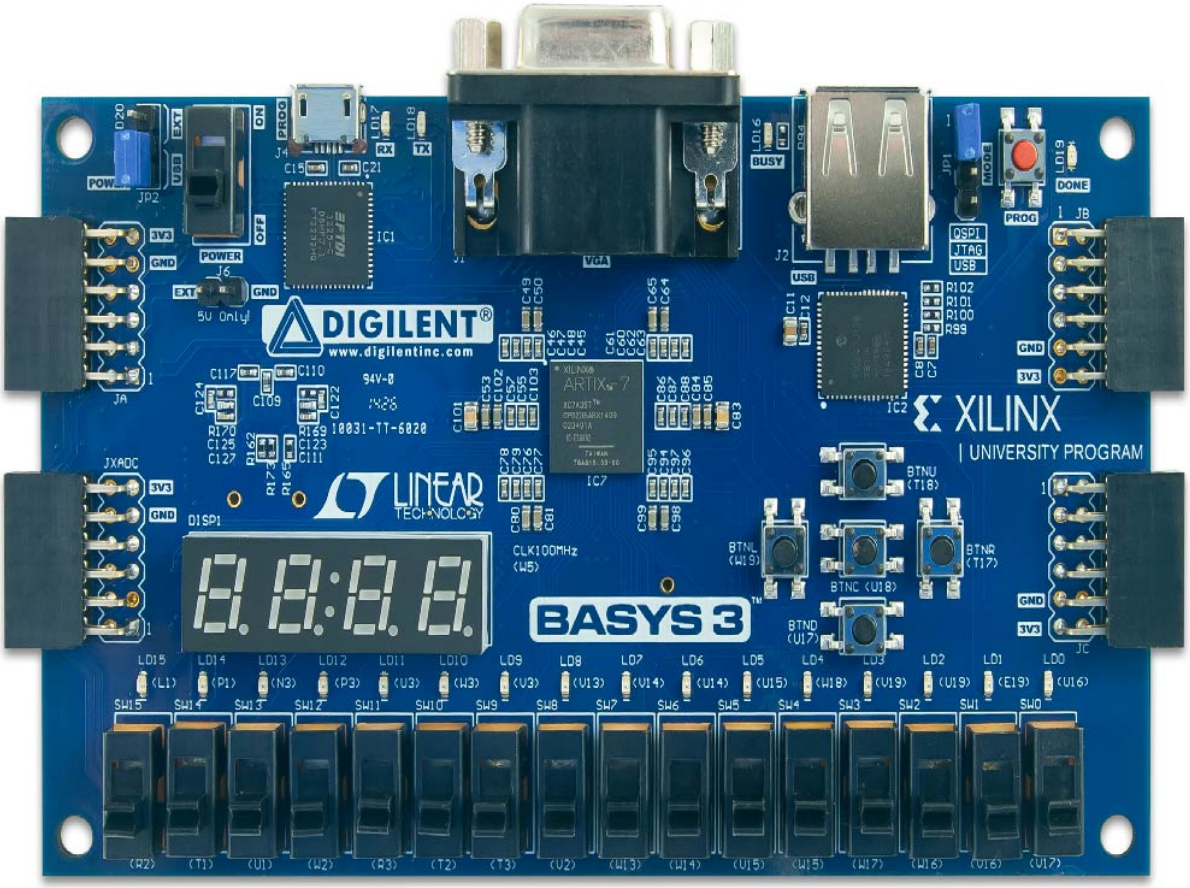


Figure 2.3: Digilent Basys 3 board

Key Features and Benefits:

- 4 Pmod (Peripheral Module) ports: 3 Standard 12-pin Pmod ports, 1 dual purpose XADC signal / standard Pmod port
- 4-digit 7-segment display
- 5 user pushbuttons
- 16 user LEDs
- 16 user switches
- USB HID Host for mice, keyboards and memory sticks
- 12-bit VGA output
- USB-UART Bridge
- Serial Flash
- Digilent USB-JTAG port for FPGA programming and communication
- On-chip analog-to-digital converter (XADC) 1,800 Kbits of fast block RAM
- 33,280 logic cells in 5200 slices (each slice contains four 6-input LUTs and 8 flip-flops)
- Features the Xilinx Artix-7 FPGA: XC7A35T-1CPG236C.
- The Basys 3 board includes a single 100MHz oscillator connected to pin W5.

2. Zynq-7000 series based Zybo Z7 FPGA board

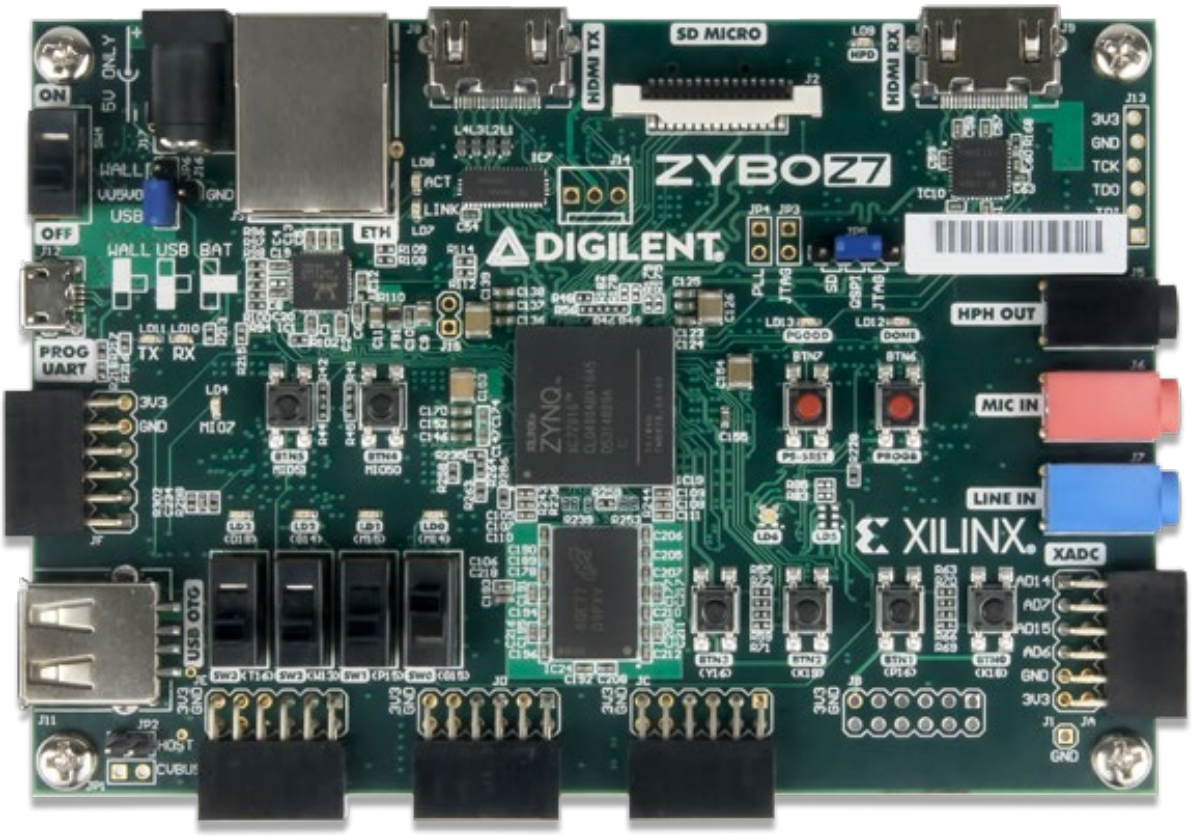


Figure 2.4: Digilent Zybo Z7 board

- ZYNQ Processor
  - 667 MHz dual-core Cortex-A9 processor
  - DDR3L memory controller with 8 DMA channels and 4 High Performance AXI3 Slave ports
  - High-bandwidth peripheral controllers: 1G Ethernet, USB 2.0, SDIO
  - Low-bandwidth peripheral controllers: SPI, UART, CAN, I2C
  - Programmable from JTAG, Quad-SPI flash, and micro SD card
  - Programmable logic equivalent to Artix-7 FPGA
- Memory
  - 1 GB DDR3L with 32-bit bus @ 1066 MHz
  - 16 MB Quad-SPI Flash with factory programmed 128-bit random number and 48-bit globally unique EUI-48/64™ compatible identifier
  - micro SD slot
- Power

        - Powered from USB or any 5V external power source
- USB and Ethernet
    - Gigabit Ethernet PHY
    - USB-JTAG Programming circuitry
    - USB-UART bridge
    - USB 2.0 OTG PHY with host and device support
- Audio and Video
    - Pcam camera connector with MIPI CSI-2 support
    - HDMI sink port (input) with/without* CEC
    - HDMI source port (output) with CEC
    - Audio codec with stereo headphone, stereo line-in, and microphone jacks
- Switches, Push-buttons, and LEDs
    - 6 push-buttons (2 processors connected)
    - 4 slide switches
    - 5 LEDs (1 processor connected)
    - 2 RGB LEDs (1*)
- Expansion Connectors
    - 6 Pmod ports (5*)
    - 8 Total Processor I/O
    - 40 Total FPGA I/O (32*)
    - 4 Analog capable 0-1.0V differential pairs to XADC

Software support:

The boards are compatible with Xilinx Vivado Design Suite. This tool set combines FPGA logic design and embedded ARM software development into an easy-to-use, intuitive design flow. It can be used for designing systems of any complexity. It runs faster, allows better use of FPGA resources, and allows designers to focus their time evaluating design alternatives.
In this project, the Vivado Web pack edition is used which is the free version of the complete Vivado Design Suite which supports both the boards used in the project.

Chapter 3

HDL

3.1. Introduction to HDL:

HDL or Hardware Description Language is a specialized computer language used to describe the structure and behavior of electronic circuits, and most commonly, digital logic circuits.

HDLs allowed the designers to model the concurrency of processes found in hardware elements. Hardware description languages such as Verilog HDL and VHDL became popular. Verilog HDL originated in 1983 at Gateway Design Automation. Later, VHDL was developed under contract from DARPA. Both Verilog and VHDL simulators to simulate large digital circuits quickly gained acceptance from designers.

Digital circuits could be described at a register transfer level
(RTL) by use of an HDL. HDLs also began to be used for system-level design. HDLs were used for simulation of system boards, interconnect buses, FPGAs (Field Programmable Gate Arrays), and PALs (Programmable Array Logic). A common approach is to design each IC chip, using an HDL, and then verify system functionality via
simulation.

3.2. HDL Basics:

HDLs can describe a digital system at several different levels—behavioral, dataflow, and structural. For example, a binary adder could be described at the behavioral level in terms of its function of adding two binary numbers without giving any implementation details. The same adder could be described at the data flow level by giving the logic equations for the adder. Finally, the adder could be described at the structural level by specifying the gates and the interconnections between the gates that comprise the adder.

HDLs lead naturally to a top-down design methodology, in which the system is first specified at a high level and tested using a simulator. After the system is debugged at this level, the design can gradually be refined, eventually leading to a structural description closely related to the actual hardware implementation. HDLs are designed to be technology independent. If a design is described in HDL and implemented in today's technology, the same HDL description could be used as a starting point for a design in some future technology.

Importance of HDLs:

HDLs have many advantages compared to traditional schematic-based design.

- Designs can be described at a very abstract level by use of HDLs. Designers can write their RTL description without choosing a specific fabrication technology. Logic synthesis tools can automatically convert the design to any fabrication technology. If a

new technology emerges, designers do not need to redesign their circuit. They simply input the RTL description to the logic synthesis tool and create a new gate-level netlist, using the new fabrication technology. The logic synthesis tool will optimize the circuit in area and timing for the new technology.

- By describing designs in HDLs, functional verification of the design can be done early in the design cycle. Since designers work at the RTL level, they can optimize and modify the RTL description until it meets the desired functionality. Most design bugs are eliminated at this point. This cuts down design cycle time significantly because the probability of hitting a functional bug at a later time in the gate-level netlist or physical layout is minimized.
- Designing with HDLs is analogous to computer programming. A textual description with comments is an easier way to develop and debug circuits. This also provides a concise representation of the design, compared to gate-level schematics. Gate-level schematics are almost incomprehensible for very complex designs.

3.3. Verilog Basics:

The Verilog HDL language was first developed by Gateway Design Automation in 1983 as a hardware modelling language for their simulator product. At that time, it was a proprietary language. Verilog HDL is a hardware description language that can be used to model a digital system at many levels of abstraction ranging from the algorithmic level to the gate-level to the switch-level. The digital system can be described hierarchically and timing can be explicitly modelled within the same description.

The Verilog HDL language includes capabilities to describe the behavioral nature of a design, the dataflow nature of a design, a design's structural composition, delays and a waveform generation mechanism including aspects of response monitoring and verification, all modelled using one single language. Verilog is a general-purpose hardware description language that can be used to describe and simulate the operation of a wide variety of digital systems, ranging in complexity from a few gates to an interconnection of many complex integrated circuits.

The Verilog HDL language includes capabilities to describe the behavioral nature of a design, the dataflow nature of a design, a design's structural composition, delays and a waveform generation mechanism including aspects of response monitoring and verification, all modelled using one single language.

Capabilities of Verilog HDL:

- Primitive logic gates, such as “and”, “or” and “nand”, are built-in into the language.
- Flexibility of creating a user-defined primitive (UDP). Such a primitive could either be a combinational logic primitive or a sequential logic primitive.
- Switch-level modeling primitive gates, such as pmos and nmos, are also built-in into the language.
- A design can be modeled in three different styles or in a mixed style. These styles are: behavioral style - modeled using procedural constructs; dataflow style -

modeled using continuous assignments; and structural style - modeled using gate and module instantiations.

- Verilog HDL is non-proprietary and is an IEEE standard.
- A design can be described in a wide range of levels, ranging from switch-level, gate-level, register-transfer-level (RTL) to algorithmic-level, including process and queuing-level
- The same single language can be used to generate stimulus for the design and for specifying test constraints, such as specifying the values of inputs.
- Verilog HDL can be used to perform response monitoring of the design under test, that is, the values of a design under test can be monitored and displayed. These values can also be compared with expected values, and in case of a mismatch, a report message can be printed.
- At the behavioral-level, Verilog HDL can be used to describe a design not only at the RTL-level, but also at the architectural-level and its algorithmic-level behavior.
- At the structural-level, gate and module instantiations can be used.
- Verilog HDL also has built-in logic functions such as & (bitwise-and) and I (bitwise-or); high-level programming language constructs such as conditionals, case statements, and loops are available in the language.

Types of coding/modelling styles in Verilog HDL:
Verilog language has the capability of designing a module in several coding styles. Depending on the needs of a design, internals of each module can be defined at three level of abstractions.

1. Behavioral level
2. Dataflow level
3. Gate level or Structural level

1. Behavioral level:
   - This is the highest level of abstraction provided by Verilog HDL.
   - A module can be implemented in terms of the desired design algorithm without concern for the hardware implementation details.
   - It specifies the circuit in terms of its expected behavior.
   - It is the closest to a natural language description of the circuit functionality, but also the most difficult to synthesize.

2. Dataflow level
   - At this level, the module is designed by specifying the data flow.
   - Looking towards this design, one can realize how data flows between hardware registers and how the data is processed in the design.
   - This style is similar to logical equations. The specification is comprised of expressions made up of input signals and assigned to outputs.
   - In most cases, such an approach can be quite easily translated into a structure and then implemented.

3. Gate level or Structural level
    - The module is implemented in terms of logic gates and interconnections between these gates.
    - It resembles a schematic drawing with components connected with signals.
    - A change in the value of any input signal of a component activates the component. If two or more components are activated concurrently, they will perform their actions concurrently as well.
    - A structural system representation is closer to the physical implementation than behavioral one but it is more involved because of large number of details. Since logic gate is most popular component, Verilog has a predefined set of logic gates known as primitives. Any digital circuit can be built from these primitives.

Chapter 4

Design Flow

Design Flow for FPGAs

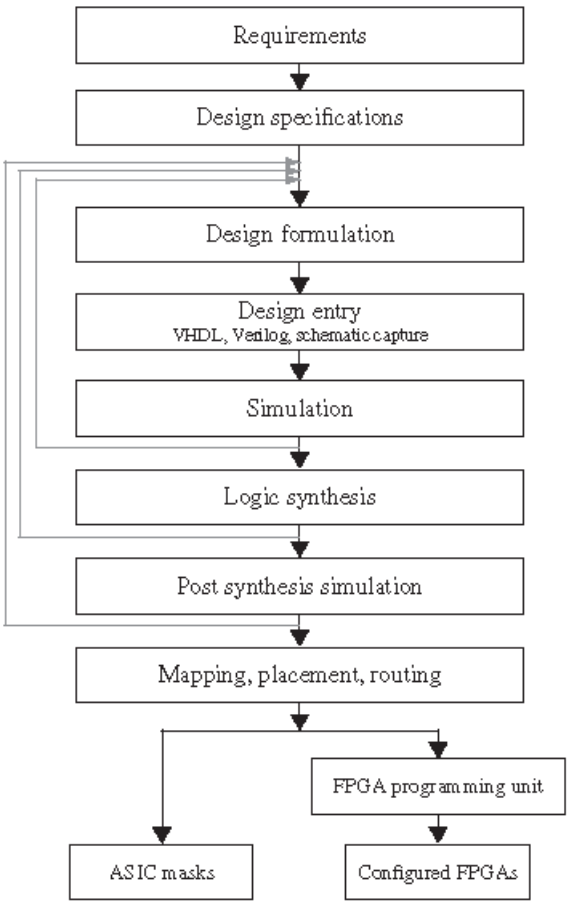


Figure 4.1: Design Flow

The design flow shown in Figure 4.1 is typically used by designers who use HDLs.

Sophisticated CAD tools are available to assist with the design of systems using programmable gate arrays. Designs can be entered in many ways. In the early days of FPGAs, designs were entered using schematic entry or even lower levels of design entry tools. Low-level design entry means less abstraction, whereas high-level means entering designs at a higher level of abstraction (e.g., behavioral Verilog/VHDL description). Early FPGA tools allowed low-level utilities to enter logic equations, Karnaugh maps, and other objects into specific logic blocks in the FPGA. Schematic capture technique means that the designer develops a schematic of the design. Schematic diagrams utilizing standard hardware components are created and entered into the CAD software.

Currently, automatic synthesis tools are available that will take a Verilog description of the system as an input and generate an interconnection of gates and flip-flops to realize the system. Behavioral models can be translated into design implementations reasonably

efficiently. Synthesis tools have advanced significantly in the last decade. The most common method of designing a digital system with an FPGA uses the following steps:

1. Create a behavioral, RTL, or structural model of the design in a hardware description language such as Verilog or VHDL.
2. Simulate and debug the design.
3. Synthesize the design targeting the desired device.
4. Run a mapping/partitioning program. This program will break the logic diagram into pieces that will fit into the configurable logic blocks.
5. Run an automatic place-and-route program. This will place the logic blocks in appropriate places in the FPGA and will then route the interconnections between the logic blocks.
6. Run a program that will generate the bit pattern necessary to program the FPGA.
7. Download the bit pattern into the internal configuration cells in the FPGA and test the operation of the FPGA.

Steps 3, 4, and 5 are often integrated in modern CAD tools. However, the processes mentioned in the steps are happening whether presented as one step or several steps. This is analogous to how general-purpose compilers have integrated compiling and assembling steps.

Chapter 5

Project Description

Microprocessor

The project includes an 8-bit microprocessor and a digital waveform generator on FPGA using Verilog HDL. The project started with deciding specifications of basic components of microprocessor which included the design of ALU, register memory, interrupts, clock and counter. Then the project started with the design of ALU which is capable of producing 8-bit outputs by accepting two or one 8-bit inputs and selecting the operation via select lines. The ALU can also produce a carry if the operation results. Some basic idea of working of microprocessor is shown below.

A Microprocessor is a computer processor where the data processing logic and control is included on a single integrated circuit, or a small number of integrated circuits. The microprocessor contains the arithmetic, logic, and control circuitry required to perform the functions of a computer's central processing unit. The integrated circuit is capable of interpreting and executing program instructions and performing a number of arithmetic operations. The microprocessor is a multipurpose, clock-driven, register-based, digital integrated circuit that accepts binary data as input, processes it according to instructions stored in its memory, and provides results (also in binary form) as output.

The Verilog code for the project is then tested using behavioral simulation. For simulation another code called “Test Bench” is written in which all the variables are initiated. The clock is defined and set as per the required frequency. The stimulus is added with delays for testing the output waveforms. After successful verification of the outputs in the simulation the code is then synthesized and implemented on the board. With the use of Vivado Design Suite the synthesis and implementation of code is automated. The placing and routing on the board is also automated. The schematic is generated by Vivado Design suite using the components and auto-generated netlist. After this the constraints or mapping of the variable are decided and written in the constraints file. The input and output ports and pins are mapped. Other than the simulation the outputs can also be tested using external hardware which was performed by me which resulted in successful testing and verification. All the diagrams are shown below; the diagram list and table includes:

1. Flowchart of the working of microprocessor
2. Block diagram of the microprocessor
3. Implemented design on the chip
4. Function table of the ALU
5. Simulation result
6. Schematic of the microprocessor

Figure5.1:
Flow chart for working of the microprocessor:

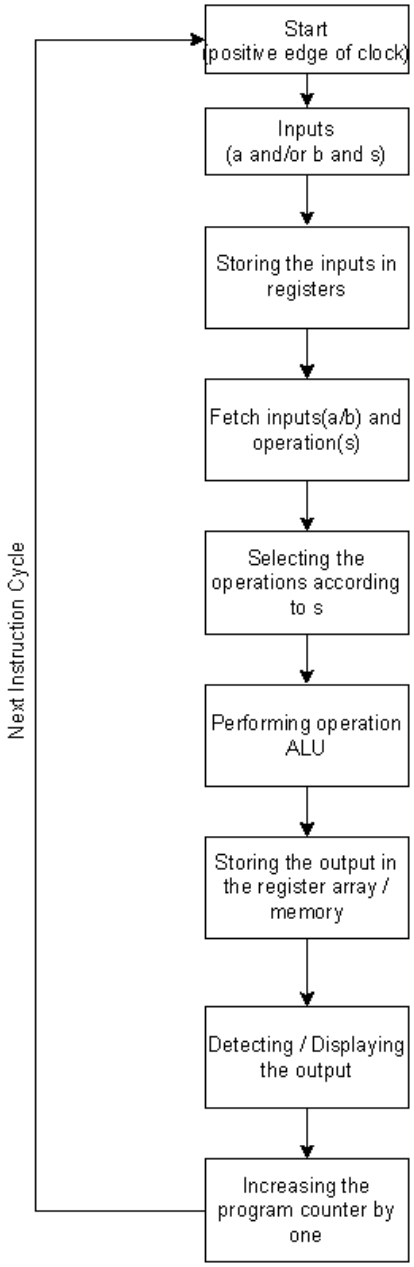


Figure 5.2:
Block Diagram of Microprocessor:

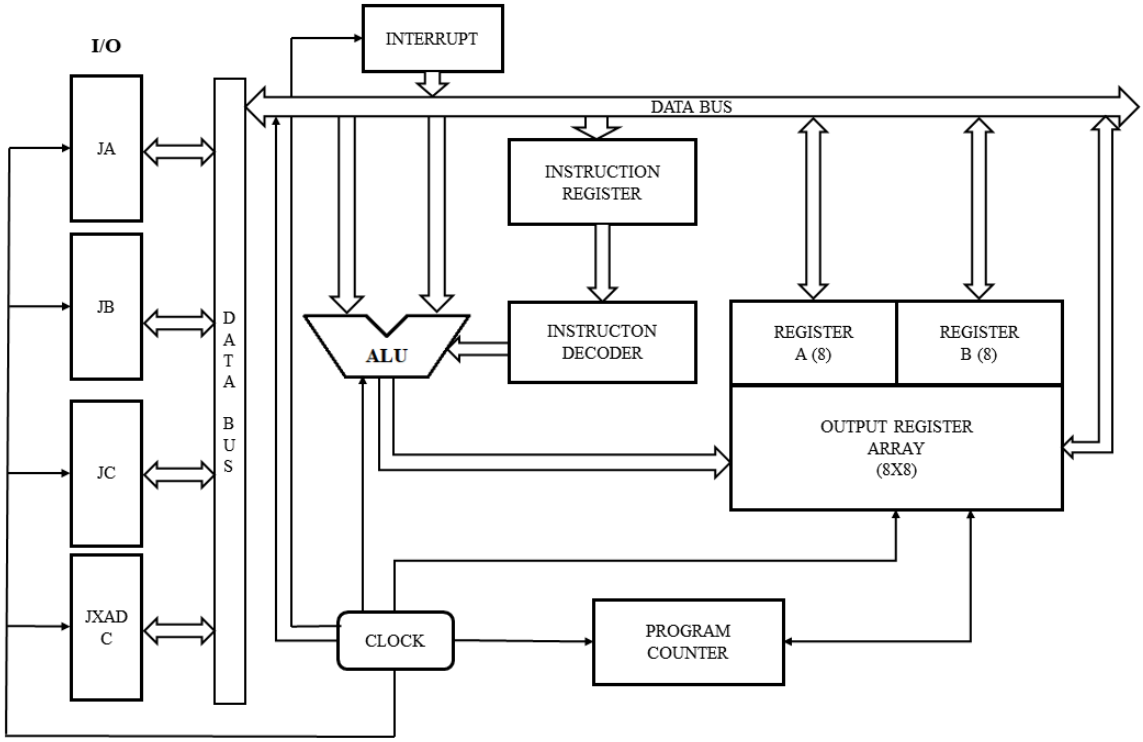


Figure 5.3:

Design of microprocessor on FPGA chip:

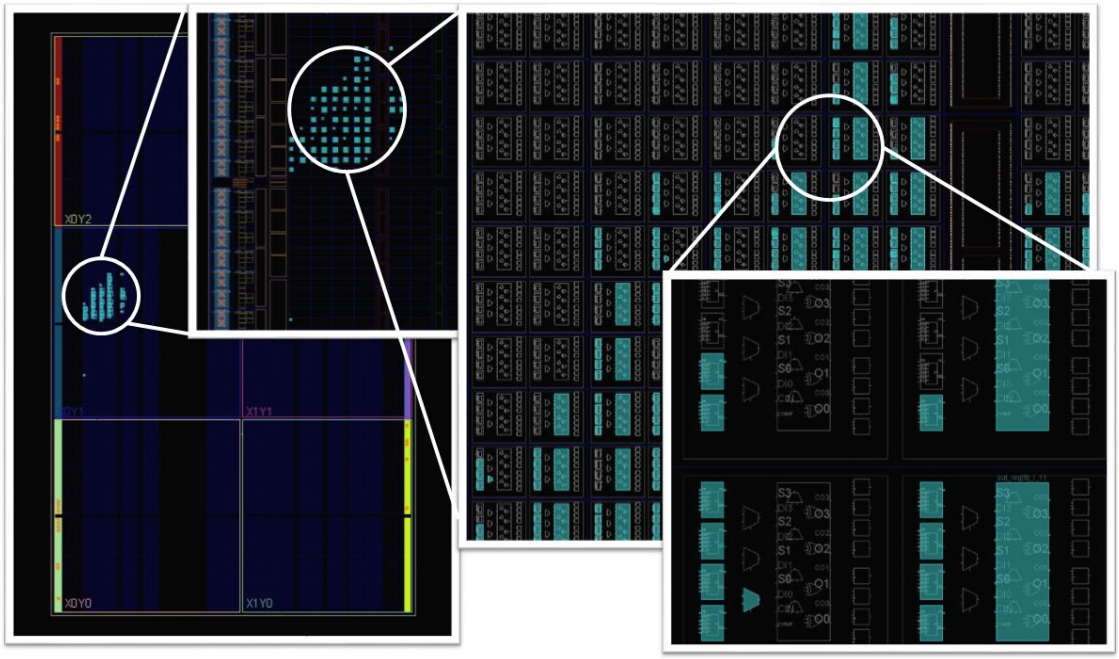


Table5.1

Arithmetic Logic Unit's Function Table						
	Select Lines				Arithmetic and Logic Functions	
S. No.	S_3	S_2	S_1	S_0	Mode = 1	Mode = 0
01.	0	0	0	0	F = ~A	F = A
02.	0	0	0	1	F = ~(A+B)	F = A+B
03.	0	0	1	0	F = ~A.B	F = ~A+B
04.	0	0	1	1	F = 0	F = -1 or 2's Complement
05.	0	1	0	0	F = ~(A.B)	F =A + A.(~B)
06.	0	1	0	1	F = ~B	F =(A+B) + A.(~B)
07.	0	1	1	0	F = A⊕B	F = A-B-1
08.	0	1	1	1	F = A.(~B)	F = A.(~B)-1
09.	1	0	0	0	F = ~A+B	F = A+A.B
10.	1	0	0	1	F = ~(A⊕B)	F = A+B+1
11.	1	0	1	0	F = B	F = (A+(~B))+A.B
12.	1	0	1	1	F = A.B	F = A.B-1
13.	1	1	0	0	F = 1	F =A+(~A)
14.	1	1	0	1	F = A + (~B)	F = (A+B)+(A)
15.	1	1	1	0	F = A+B	F = (A+B)+(B)
16.	1	1	1	1	F = A	F = A-1

Figure 5.4:

Behavioural Simulation Results:

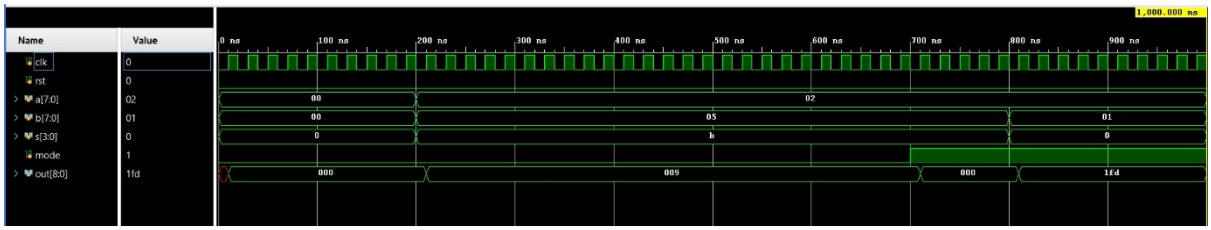
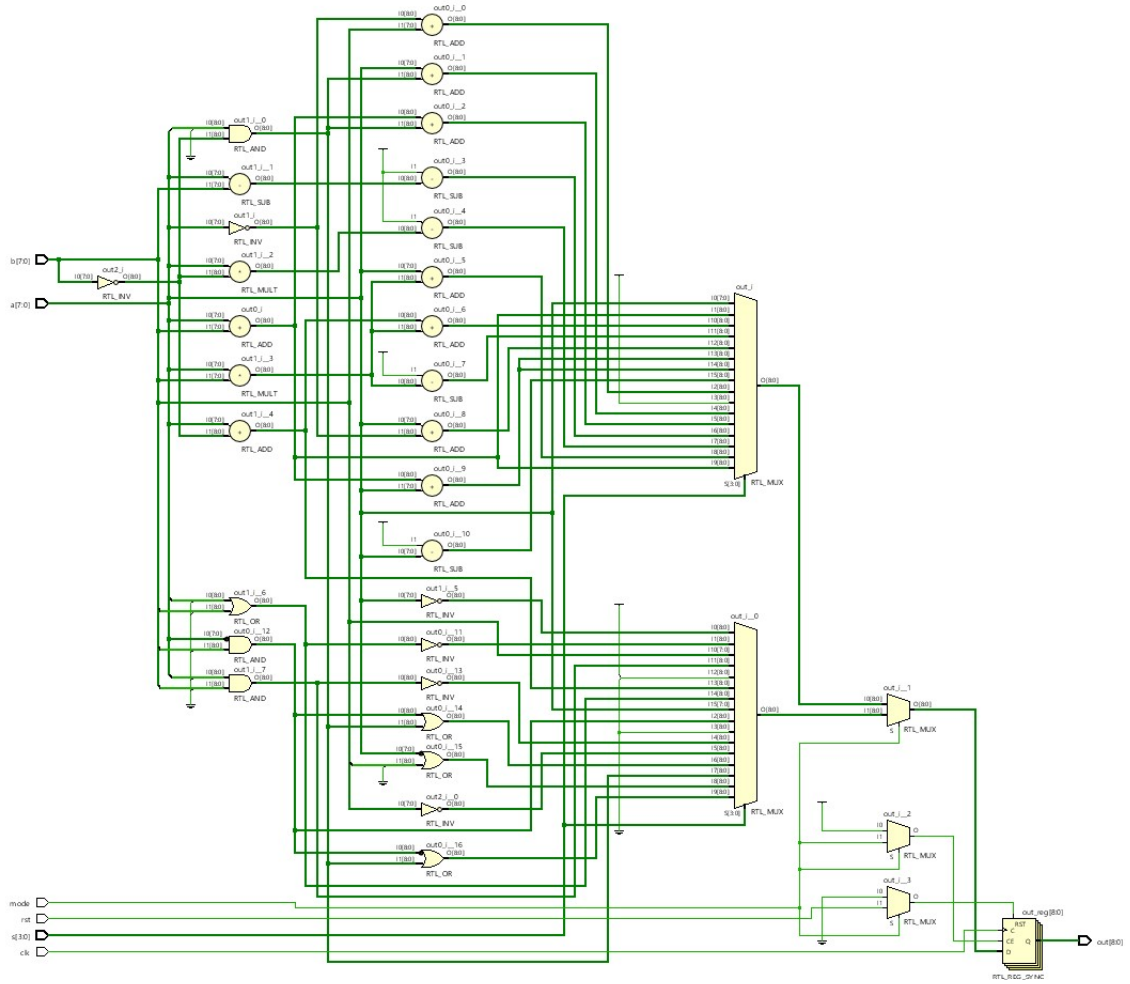


Figure 5.5:

Schematic of the microprocessor:



Digital Waveform Generator

For the digital waveform generator part, three functions square, saw-tooth and triangular functions were decided. Verilog HDL code was written such that the 8-bit output variable can generate 8-bit signal such that the output when connected to DAC will generate given functions. The frequency of the wave is same as the base frequency of the clock of the FPGA board. Eight pins on the board are dedicated for the 8-bit output of the generator. The output is such that the binary or digital output obtained results in waves or functions. The 8-bit output is connected to an 8-bit Digital to Analog Converter (DAC) IC which then converts the digital signal to analog signal which can be observed on the DSO. Test bench for this code is also written and simulated. The simulation results show the functions generated using the Verilog code written. The diagrams shown below are:

1. Simulation results
2. Schematic of the 8-bit output to analog circuit

Simulation results:

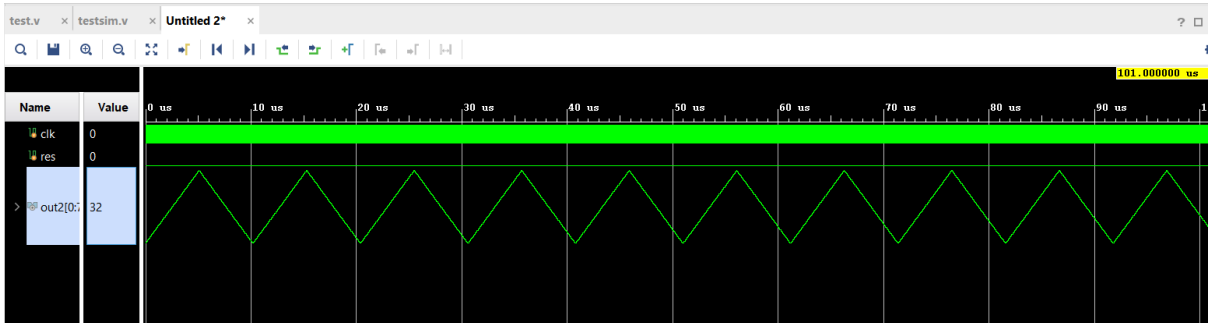


Figure 5.6: Triangular Waveform

Schematic of 8-bit output to analog circuit:

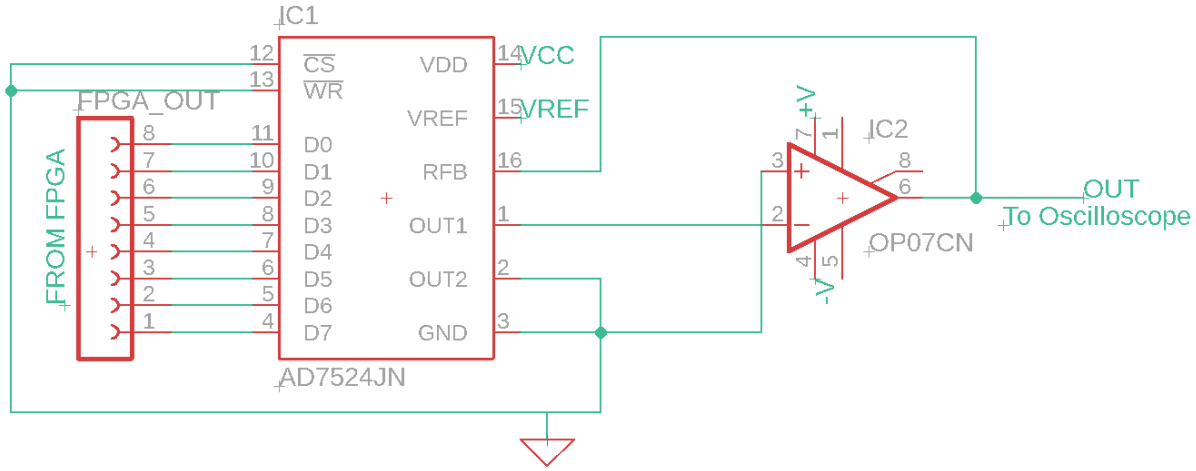


Figure 5.7: Schematic

Chapter 6

PLC and SCADA

Programmable Logic Controller

A Programmable Logic Controller, or PLC for short, is simply a special computer device used for industrial control systems. They are used in many industries such as oil refineries, manufacturing lines, conveyor systems and so on. Where ever there is a need to control devices the PLC provides a flexible way to "softwire" the components together.

The basic units have a CPU (a computer processor) that is dedicated to running one program that monitors a series of different inputs and logically manipulates the outputs for the desired control. They are meant to be very flexible in how they can be programmed while also providing the advantages of high reliability (no program crashes or mechanical failures), compact and economical over traditional control systems.

PLC is programmed using Ladder Logic programming. This is a graphical programming language in which programming is done using logic blocks in the form of a ladder in which the left side of the ladder is V_{cc} and the right side is Ground. Ladder logic programming is an easy type of programming which does not require specialized training in programming languages. Programming can be easily modified.

Working of PLC

A programmable logic controller is a specialized computer used to control machines and processes. It, therefore, shares common terms with typical PCs like central processing unit, memory, software and communications. Unlike a personal computer, the PLC is designed to survive in a rugged industrial atmosphere and to be very flexible in how it interfaces with inputs and outputs in the real world. The components that make a PLC work can be divided into three core areas.

- The power supply and rack
- The central processing unit (CPU)
- The input/output (I/O) section

PLCs come in many shapes and sizes. They can be so small as to fit in your shirt pocket while more involved control systems require large PLC racks. Smaller PLCs (a.k.a. “bricks”) are typically designed with fixed I/O points. For our consideration, we’ll look at the more modular rack-based systems. It’s called “modular” because the rack can accept many different types of I/O modules that simply slide into the rack and plug in.

Figure 6.1: PLC

The Power Supply and Rack

So let's start off by removing all our modules which leaves us with a naked PLC with only the power supply and the rack. The rack is the component that holds everything together. Depending on the needs of the control system it can be ordered in different sizes to hold more modules. Like a human spine the rack has a backplane at the rear which allows the cards to communicate with the CPU. The power supply plugs into the rack as well and supplies a regulated DC power to other modules that plug into the rack. The most popular power supplies work with 120 VAC or 24 VDC sources.

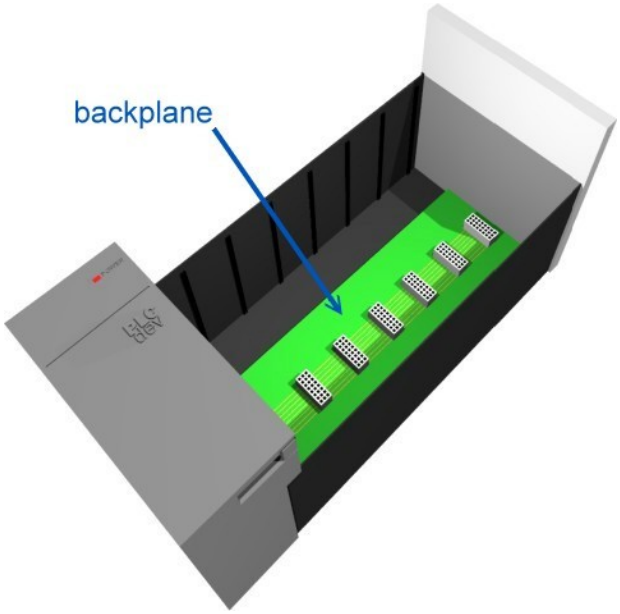


Figure 6.2: Backplane

The CPU

The brain of the whole PLC is the CPU module. This module typically lives in the slot beside the power supply. Manufacturers offer different types of CPUs based on the complexity needed for the system. The CPU consists of a microprocessor, memory chip and other integrated circuits to control logic, monitoring and communications. The CPU has different

operating modes. In *programming mode,* it accepts the downloaded logic from a PC. The CPU is then placed in *run mode* so that it can execute the program and operate the process.
Since a PLC is a dedicated controller it will only process this one program over and over again. One cycle through the program is called a scan time and involves reading the inputs from the other modules, executing the logic based on these inputs and then updated the outputs accordingly. The scan time happens very quickly (in the range of 1/1000th of a second). The memory in the CPU stores the program while also holding the status of the I/O and providing a means to store values.

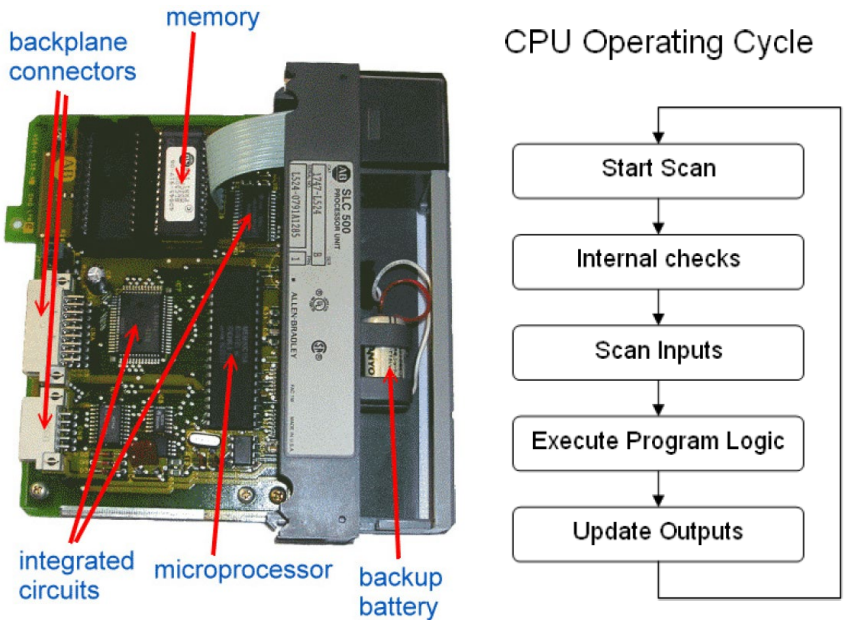


Figure 6.3: CPU and Operating Cycle

I/O System

The I/O system provides the physical connection between the equipment and the PLC. Opening the doors on an I/O card reveals a terminal strip where the devices connect.
There are many different kinds of I/O cards which serve to condition the type of input or output so the CPU can use it for its logic. It's simply a matter of determining what inputs and outputs are needed, filling the rack with the appropriate cards and then addressing them correctly in the CPUs program.

Inputs

Input devices can consist of digital or analog devices. A digital input card handles discrete devices which give a signal that is either on or off such as a pushbutton, limit switch, sensors or selector switches. An analog input card converts a voltage or current (e.g. a signal that can be anywhere from 0 to 20mA) into a digitally equivalent number that can be understood by the CPU. Examples of analog devices are pressure transducers, flow meters and thermocouples for temperature readings

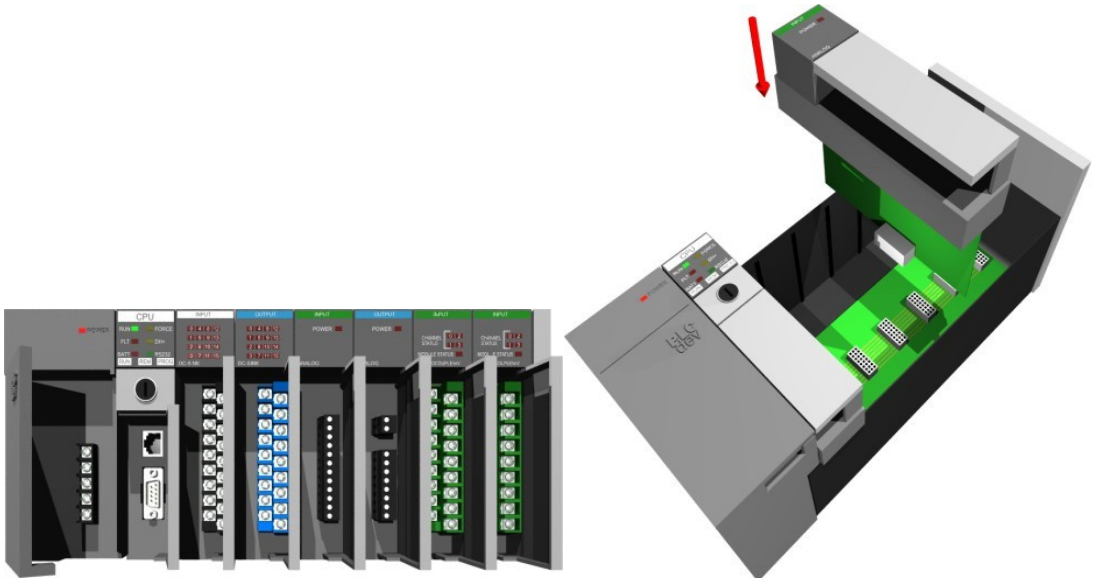


Figure 6.4: I/O System

Outputs

Output devices can also consist of digital or analog types. A digital output card either turns a device on or off such as lights, LEDs, small motors, and relays. An analog output card will convert a digital number sent by the CPU to it's real world voltage or current. Typical outputs signals can range from 0-10 VDC or 4-20mA or 24VDC depending upon the PLC and are used to drive mass flow controllers, pressure regulators and position controls.

Supervisory Control and Data Acquisition

SCADA (supervisory control and data acquisition) is a category of software applications for controlling industrial processes, which is the gathering of data in real time from remote locations in order to control equipment and conditions. SCADA provides organizations with the tools needed to make and deploy data-driven decisions regarding their industrial processes.

One of the most commonly used types of industrial control system, SCADA can be used to manage almost any type of industrial process. SCADA systems include hardware and software components. The hardware gathers and feeds data into field controller systems, which forward the data to other systems that process and present it to a human-machine interface (HMI) in a timely manner. SCADA systems also record and log all events for reporting process status and issues. SCADA applications warn when conditions become hazardous by sounding alarms.

SCADA uses high quality graphics for monitoring. It has large number of inventory of symbols and graphics which can resemble real world equipment and systems. Sample of SCADA is shown below.

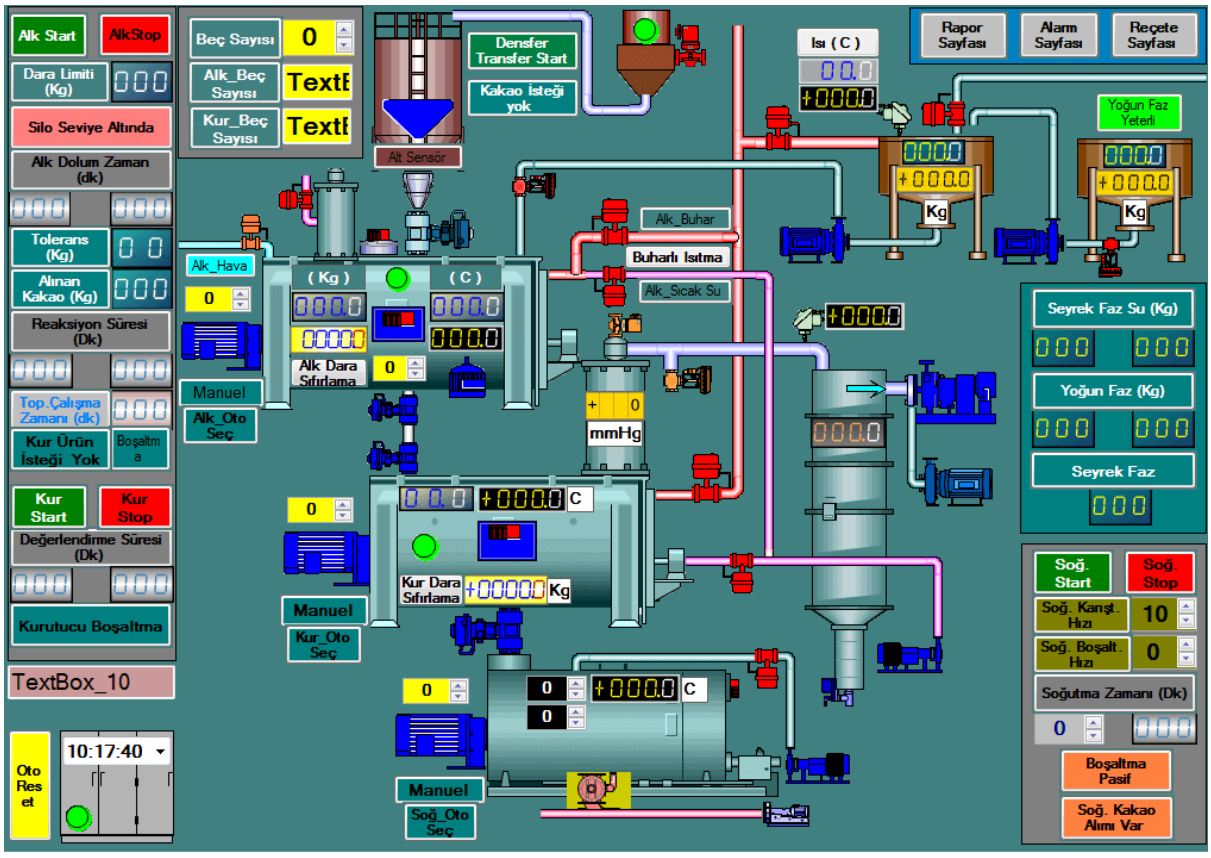


Figure 6.5: SCADA Screen

Features of SCADA systems

Although SCADA systems may include special features for specific industries or applications, most systems support the following features:

- **Data acquisition** is a foundation of SCADA systems; sensors collect data and deliver it to field controllers, which, in turn, feed data to the SCADA computers.
- **Remote control** is achieved through the control of field actuators, based on the data acquired from field sensors.
- **Networked data communication** enables all SCADA functions. Data collected from sensors must be transmitted to SCADA field controllers, which, in turn, communicate with the SCADA supervisory computers; remote control commands are transmitted back to actuators from the SCADA supervisory computers.
- **Data presentation** is achieved through HMIs, which represent current and historical data to the operators running the SCADA system.
- **Real-time and historical data** are both important parts of the SCADA system, as they enable users to track current performance against historical trends.
- **Alarms** alert SCADA operators to potentially significant conditions in the system. Alerts can be configured to notify operators when processes are blocked, when systems are failing, or when other aspects of SCADA processes need to be stopped, started or adjusted.
- **Reporting** on SCADA system operations can include reports on system status, process performance and reports customized to specific uses.

During the project training duration, I got hands-on experience working with PLC and SCADA systems. I learned and implemented ladder logic programming on a given PLC and designed a basic screen on **Unity Pro XL** software. Worked with digital and analog inputs and outputs using the I/O modules. Also generated a square wave with varying frequency and duty cycle and observed the latter on an LED. Designed SCADA for the previous PLC programs and acquired data graphically and controlled the physical parameters using graphical screen used. Used **Citect Vijeo** for SCADA. Both PLC and SCADA was provided by **Schneider Electric.**

PLC used:

- CPU – Modicon M580
- Power Supply – Schneider Electric BMXCPSS500
- Rack - BMEXBP1200 X80 - 12 slots - Ethernet backplane
- Digital Input Module – Modicon X80 I/O BMXDDI1602
- Digital Output Module - Modicon X80 I/O BMXDDI1602
- Analog Input Module - Modicon X80 I/O BMXAMI0810
- Analog Output Module - Modicon X80 I/O BMXAMO0410

Chapter 7

Conclusion and Future Scope

In RRCAT, this internship/project training has been a knowledgeable and rewarding experience. I can conclude that there has been a lot I've learnt from my work at the training in this research center.

I have Designed and Developed a simple 8-bit microprocessor and a digital waveform generator on FPGA board using Verilog language which involved the knowledge of digital circuits, microprocessors and their architecture, analog circuits and signals.
The design has been successfully tested using a test bench and simulations as well as using external hardware.

The final goal of overall work is to study the flow of design and work in the VLSI and automation industry and get hands-on experience with FPGAs, PLC and SCADA.

The other goal of this project is to learn to design digital circuits using Verilog HDL, learn the steps to implement the design on FPGA, and learn the basics of automation which include ladder logic, and control using PLC and SCADA.

Future Scope:

The function table of pre-defined instructions can be increased for operations. The microprocessor implemented can be paired with external memory. The microprocessor can optimize storage and data flow using designated addresses for storage slots and instructions. The microprocessor's instruction cycle could be optimized for better utilization of resources.

Chapter 8

References

- Microprocessor Architecture, Programming and Applications with the 8085 (Ramesh S. Gaonkar)
- Digital VLSI Systems Design A Design Manual for Implementation of Projects on FPGAs and ASICs Using Verilog (Seetharaman Ramachandran)
- Dedicated Digital Processors Methods in Hardware & Software Co-Design (F. Mayer-Lindenberg)
- Verilog Language Reference, Verilog Modeling Style Guide (CFE), Product Version 3.1
- Digital Systems Principles and Applications, 12th Edition (Ronald J. Tocci, Neal S. Widmer etc.)
- John R. Hackworth, Frederick D. Hackworth - PLC Programming Methods and Applications (, Prentice Hall)
- A Verilog HDL Primer - J. Bhaskar
- Verilog HDL – Samir Palnitkar
- A FPGA based Forth microprocessor, P.H.W. Leong, P.K. Tsang and T.L. Lee, Chinese University of Hong Kong
- *International Journal of Engineering Applied Sciences and Technology, 2018,* DESIGN AND IMPLEMENTATION OF 8 BIT AND 16 BIT ALU USING VERILOG LANGUAGE, MANIT KANTAWALA, Dept. of Electronic & Communication, Global Institute of Technology, Jaipur, Rajasthan, India
- Digital Systems Design Using VHDL (Charles Roth)
- https://digilent.com/reference/programmable-logic/basys-3/reference-manual?redirect=1
- https://digilent.com/reference/programmable-logic/zybo-z7/reference-manual?redirect=1