WAVEFORM GENERATOR USING FPGA

A Project Report Submitted By

SHENOY APOORVA GANAPATI	**4NM21EC129**
SNEVIN LEONEEL DSOUZA	**4NM21EC146**
TEJAS POOJARY	**4NM21EC170**
U. AKASH SHENOY	**4NM21EC172**

Under the Guidance of

Mr. Anil Kumar Bhat

Assistant Professor Gd-II

in partial fulfillment of the requirements for the award of the Degree of

Bachelor of Engineering

in

Electronics and Communication Engineering

from

Visvesvaraya Technological University, Belagavi

NITTE EDUCATION TRUST | NMAM INSTITUTE OF TECHNOLOGY

Nitte-574110, Karnataka, India

November-2024

NITTE EDUCATION TRUST | N.M.A.M. INSTITUTE OF TECHNOLOGY
(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi)
Nitte – 574 110, Karnataka, India
ISO 9001:2015 Certified, Accredited with 'A' Grade by NAAC
☎:08258 - 281039 – 281263, Fax: 08258 – 281265

Department of Electronics and Communication Engineering

Certificate

*Certified that the project work entitled "**Waveform Generator using FPGA**" is a bonafide work carried out by **Shenoy Apoorva Ganapati (4NM21EC129), Snevin Leoneel Dsouza (4NM21EC146), Tejas Poojary (4NM21EC170) & U. Akash Shenoy (4NM21EC172)** in partial fulfillment of the requirements for the award of **Bachelor of Engineering Degree in Electronics and Communication Engineering** prescribed by **Visvesvaraya Technological University, Belagavi** during the year 2024-2025. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the Bachelor of Engineering Degree.*

Signature of the Guide **Signature of the HOD**

Signature of the Principal

Semester End Viva Voce Examination

	Name of the Examiners	Signature With Date
1.	_____	_____
2.	_____	_____

Abstract

The design and implementation of waveform generators play a critical role in various fields, including communication systems, signal processing, and instrumentation. A versatile waveform generator using Field-Programmable Gate Arrays (FPGAs) is developed, capable of generating a diverse range of waveforms such as sine, cosine, triangular, and square waves. FPGAs are chosen for their parallel processing capabilities, high precision, and reconfigurability, making them an ideal platform for real-time signal generation. Central to the waveform generator is the Direct Digital Synthesis (DDS) architecture, which offers precise control over the frequency, and amplitude of the generated signals. The DDS method is particularly effective for applications that require rapid switching between different waveform types and dynamic adjustments to signal parameters. The architecture of the FPGA allows for the integration of multiple DDS cores, each responsible for producing a specific waveform, enabling seamless selection or combination of waveforms as needed.

Acknowledgement

Our project would not have been successful without the encouragement, guidance and support by various personalities. First and foremost, we would like to express our sincere gratitude towards our project guide **Mr. Anil Kumar Bhat**, Assistant Professor Gd-II, Department of Electronics and Communication Engineering, N. M. A. M. Institute of Technology, Nitte, for his guidance, encouragement and inspiration throughout the project work.

We extend our gratitude to **Dr. K. V. S. S. S. S. Sairam**, Professor and Head, Department of Electronics and Communication Engineering, N. M. A. M. Institute of Technology, Nitte, for his encouragement and providing necessary facilities in the department.

We wish to acknowledge the support of **Dr. Niranjan N. Chiplunkar,** Principal, N. M. A. M. Institute of Technology, Nitte, for providing motivational academic environment.

We would like to express our sincere thanks to all the teaching and non-teaching staff of the department of Electronics and Communication Engineering, N. M. A. M. Institute of Technology, Nitte, for their patience and help during our project work.

Finally, we would like to thank all our friends and well-wishers who have helped us whenever needed and supported us.

Shenoy Apoorva Ganapati
Snevin Leoneel Dsouza
Tejas Poojary
U. Akash Shenoy

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 General Introduction

Waveform generators are essential equipment in many fields such as communications, signal processing, and instrumentation. They could potentially create all sorts of waveforms, including sine, square, triangular waveforms, which are often very useful for testing, analysis, and simulation. The waveform generators traditionally use analog circuits or microcontroller based solutions, which often suffer in flexibility, precision, and scalability. The discovery of Field Programmable Gate Arrays (FPGAs) has revolutionized the design and implementation of waveform generators. FPGAs offer an adaptive environment for realtime signal generation due to their parallel processing and high reconfigurability features. It is feasible to design waveform generators using FPGAs that are able to produce a large spectrum of waveforms and adaptable for use in very different application-specific requirements. The DDS technology allows for complete control over waveform characteristics including frequency, amplitude. DDS systems make use of a phase accumulator in generating digital waveforms which is then digitized into an analog signal by the DAC. This approach accounts for less distortion with high accuracy making it suitable for demanding applications. The flexibility of FPGAs allows for the integration of Multiple waveform generation cores within a device. This facilitates the production of different waveforms simultaneously or numerous channels for the same waveform, so the critical demands in processing signals are fulfilled. Moreover, the utilization of hardware description languages like VHDL or Verilog makes it possible to tailor and optimize the design of the waveform generator for specific requirements. In this overview, basic concepts and benefits of using FPGAs in waveform generation are considered. It points to the benefits of DDS technology with advanced features integrated into FPGA platforms. The FPGA based implementation further enhances both performance and variability in wave generators while opening up possibilities in the creation of novel signal generation and processing systems.

1.2 Aim

To design and implement a versatile FPGA-based waveform generator capable of producing various waveforms with high precision.

1.3 Objectives

- To develop a versatile waveform generator capable of producing sine, square, triangular, sawtooth, sinc and cosine waveforms.

- To use the FPGA's parallel processing capabilities to enable real-time waveform generation.

- To implement Direct Digital Synthesis (DDS) in FPGA for high precision and low distortion in waveform generation.

- To integrate hardware blocks such as Phase-Locked Loop (PLL) and Digital-to-Analog Converters (DACs) for signal generation.

- To allow user control of waveform type and frequency through a keypad input and display the selected settings on an LCD screen.

- To provide real-time monitoring and verification of generated waveforms using a logic analyzer like SignalTap and simulation tools like ModelSim.

1.4 Problem Formulation

Waveform generators are one of the important parts in modern electronics and are applied in different fields, such as communications, signal processing, and instrumentation. However, traditional waveform generators based on analog circuits or microcontroller-based solutions do not provide an opportunity to make these types of generators flexible, scalable, and accurate to various waveforms, like sine, square, and triangular waves. In systems requiring real-time generation of signals and fast change of signal type when trying to fulfill requirements from various applications, this situation might be quite serious.

The FPGAs solution therefore offers a flexible platform in that it can handle these issues and produce adjustable waveform generators that will enable the manufacture of many waveforms simultaneously. Using FPGAs, engineers can combine DDS technology with other advanced techniques in order to gain accurate control over wave features like frequency, amplitude, phase, and also over several channels as well as types of waveforms.

1.5 Proposed Method/Technique

The initially proposed FPGA-based waveform generator utilized the Cyclone-V FPGA for its high performance and integrated Phase-Locked Loop (PLL) capabilities. A 4x4 matrix keypad allowed users to input waveform type and frequency selections, while a 16x2 LCD provided real-time feedback. The keypad and LCD were interfaced with the FPGA using System Verilog, where a finite state machine (FSM) manages system operations, including user input, waveform selection, and timing through a 29-bit counter. The PLLs generated 1 MHz, 4 MHz and 100 MHz clock signals, ensuring synchronization across the system. The Direct Digital Synthesis (DDS) method was used for waveform generation, with precomputed digital values for waveforms such as sine, cosine, triangular, and square stored in lookup tables (LUTs). The FPGA retrieved these values based on user inputs, sending them to the AD9708 Digital-to-Analog Converter (DAC) to produce an analog signal, which was then amplified using an AD8066 amplifier and filtered to achieve a clean output. Simulation and verification were conducted in Quartus Prime and ModelSim, with waveform values generated and quantized in MATLAB, ensuring the system produces accurate waveforms in real-time, selectable via user inputs.

1.6 Methodology

The FPGA-based waveform generator is designed using the MAX-10 FPGA which has high performance and is integrated along with a Phase-Locked Loop. The type and frequency of waveforms to be generated are entered through a 4x4 matrix keypad. A 16x2 LCD gives real-time feedback about the selected waveform as well as the frequency. The keypad and LCD are interfaced through System Verilog with the FPGA, while the system operations are controlled with a finite state machine, which controls user input, waveform selection, and timing through a 29-bit counter. Effective system-wide synchronization is achieved with PLLs that produces 1 MHz, 4 MHz, 32 MHz and 100 MHz clock signal.

It employs Direct Digital Synthesis (DDS) to generate waveforms. Inside the FPGA, precomputed digital representations of waveforms include sine, cosine, triangular, and square waves, which are stored within lookup tables (LUTs). Based on user input, the FPGA retrieves the appropriate values from the LUTs and passes them to the AD9708 Digital-to-Analog Converter, which converts those digital values into an analog output.

After this, a signal is amplified by an AD8066 operational amplifier.Simulation and verification processes are carried out using Quartus Prime and ModelSim to validate the correct functionality of the design. Additionally, waveform values are

generated and measured in MATLAB to ensure the required precision. A fully integrated system enables the real-time generation and selection of waveforms, making for one flexible and adaptable tool for signal generation.

1.7 Literature Survey

Shoucheng Ding et al. [1] presents the design of a digital waveform generator using a Cyclone II FPGA with Quartus II as the development platform. The generator produces single waveform via Direct Digital Synthesis (DDS), utilizing a 10-bit phase accumulator for frequency control and an analog multiplier for amplitude modulation. A K8051 microcontroller, programmed in C, manages frequency and amplitude inputs via a 4×4 matrix keypad, with output displayed on an LCD1602 screen. The FPGA design, implemented in VHDL, features ROM for waveform storage and multipliers for amplitude control. The system demonstrates high precision, low distortion, and is well-suited for applications in communication, radar, and scientific experiments, highlighting the advantages of FPGA-based digital signal processing.

Ruan Yue et al. [2] highlights advancements in signal generators with FPGA-based Direct Digital Synthesis (DDS) technology. Unlike early analog signal generators, DDS enables precise and flexible waveform generation. Recent studies emphasize the use of soft-core processors like Nios II within FPGA systems for customizable solutions, enhanced by modern design tools for hardware-software co-design. Integration with high-speed digital-to-analog converters (DACs) allows for higher frequencies and faster update rates. Additionally, SOPC (System on Programmable Chip) design methodologies facilitate multi-function integration on a single chip, improving efficiency, scalability, and adaptability for high-speed, multi-channel applications.

Hongshan Nie et al. [3] discusses the crucial role of waveform generators in various applications, including radar, communication, and electronic warfare. Traditional waveform generators have significantly evolved with advancements in technology, particularly through the use of FPGA-based Direct Digital Synthesis (DDS). DDS technology enhances the precision and flexibility of waveform generation, allowing for the production of signal . High-speed components like digital-to-analog converters (DACs) and operational amplifiers are essential for achieving faster update rates and higher signal frequencies. The integration of multi-stage pipeline phase accumulators and complex clock generation techniques further improves the performance of these systems. Recent studies and developments have focused

on optimizing these components to achieve output rates up to 400MSPS, with accurate frequency generation and minimal error. Simulation and testing confirm that the waveform generator presented in this paper effectively meets the demands of high-speed, making it an indispensable tool in advanced electronic systems.

Zeliang Liu [4] reviews advancements in Direct Digital Synthesis (DDS) technology for waveform generators. Early research by Tierney, Rader, and Gold in 1971 established DDS principles, including phase accumulation and digital-to-analog conversion. Recent developments integrate DDS with System on a Programmable Chip (SOPC) technology using FPGA platforms, enhancing system design and functionality. Research, including studies by Stensby, addresses issues like quantization and phase truncation errors, emphasizing the need for optimizing ROM table size and DAC resolution. The use of Altera's Nios II soft-core processor has improved flexibility and scalability in FPGA-based DDS systems, demonstrating the effectiveness of DDS and SOPC for high-quality waveform generation in modern signal processing.

Qiang Chen [5] discusses a research focusing on the implementation of Direct Digital Frequency Synthesis (DDS) in signal generators, particularly using Field-Programmable Gate Arrays (FPGA). DDS is highlighted as an advanced technique offering high frequency resolution, stability, and low phase noise compared to traditional analog methods. The review contrasts DDS with other frequency synthesis methods like Phase-Locked Loop (PLL), emphasizing DDS's superior flexibility and performance. It also discusses the practical advantages of using FPGAs in DDS systems, such as programmability, speed, and cost-effectiveness, making them ideal for developing versatile, low-cost function generators capable of producing arbitrary waveforms.

Anurag Roy [6] introduces an FPGA-based system that combines a function generator, lock-in amplifier, and PID controller into a single, integrated platform. Leveraging the flexibility of digital implementation via FPGA software processing, the system is capable of generating waveform, performing real-time signal detection, and providing precise control with automatic relocking capabilities. This integration simplifies experimental setups by replacing the need for multiple standalone instruments, making it a versatile tool for a wide range of applications. Its adaptability and efficiency are particularly valuable in both research and industrial environments, where streamlined workflows and reliable performance are essential.

Liu Yanming et al. [7] discusses a versatile signal generator based on an FPGA, specifically ALTERA's EP4CE6F17C8 from the Cyclone IV family, which integrates Phase-Locked Loop (PLL) and Direct Digital Synthesizer (DDS) technologies. Unlike traditional signal generators that rely on discrete devices for each waveform type, this FPGA-based design offers a flexible, and cost-effective solution capable of generating waveform. The integration of PLL ensures precise and stable signal output, while DDS provides rapid frequency switching and high resolution. Experimental results demonstrate the generator's capability to deliver accurate and diverse signal outputs, highlighting its advantages over conventional methods in terms of flexibility and cost efficiency.

Sergiy Lukin et al. [8]discusses the use of FPGA-based devices for generating complex signals in modern noise radars. The authors explore two main methods for signal generation: pre-recorded waveform samples and real-time programmable waveform generation, both of which use the flexibility of FPGAs. The study delves into the basics of Pseudo-Noise (PN) sequences and Linear Congruential Generators (LCG), demonstrating their feasibility for real-time signal generation. The FPGA's ability to generate and process wideband noise signals without fast ADCs is highlighted through the stepped-delay method for noise radar receivers. Experimental results using the AWG472 board draw special attention to the effectiveness of these approaches in producing high-resolution radar measurements, offering significant potential for advancing noise radar technology. The research contributes to the FP-7 Project SCOUT, which aims to enhance radar capabilities through creative signal generation and processing techniques.

1.8 Organization of the Report

Chapter 1 introduces the fundamentals, objectives, and methodologies for designing an FPGA-based waveform generator, emphasizing its advantages over traditional methods in terms of flexibility, precision, and scalability.

Chapter 2 explains the system design of the FPGA-based waveform generator, detailing the block diagram, module functions, and schematic components.

Chapter 3 outlines the hardware requirements for the FPGA-based waveform generator, detailing components like the FPGA, keypad, LCD display, DAC, and amplifier.

Chapter 4 describes the software tools used in designing, simulating, and verifying the FPGA-based waveform generator, including Quartus Prime, SignalTap Logic Analyzer, ModelSim, and MATLAB for generating lookup tables.

Chapter 5 presents the setup, module descriptions, results on the oscilloscope, and conclusions of the FPGA-based waveform generator, demonstrating its successful implementation and testing.

Chapter 2
System Design

2.1 Block Diagram

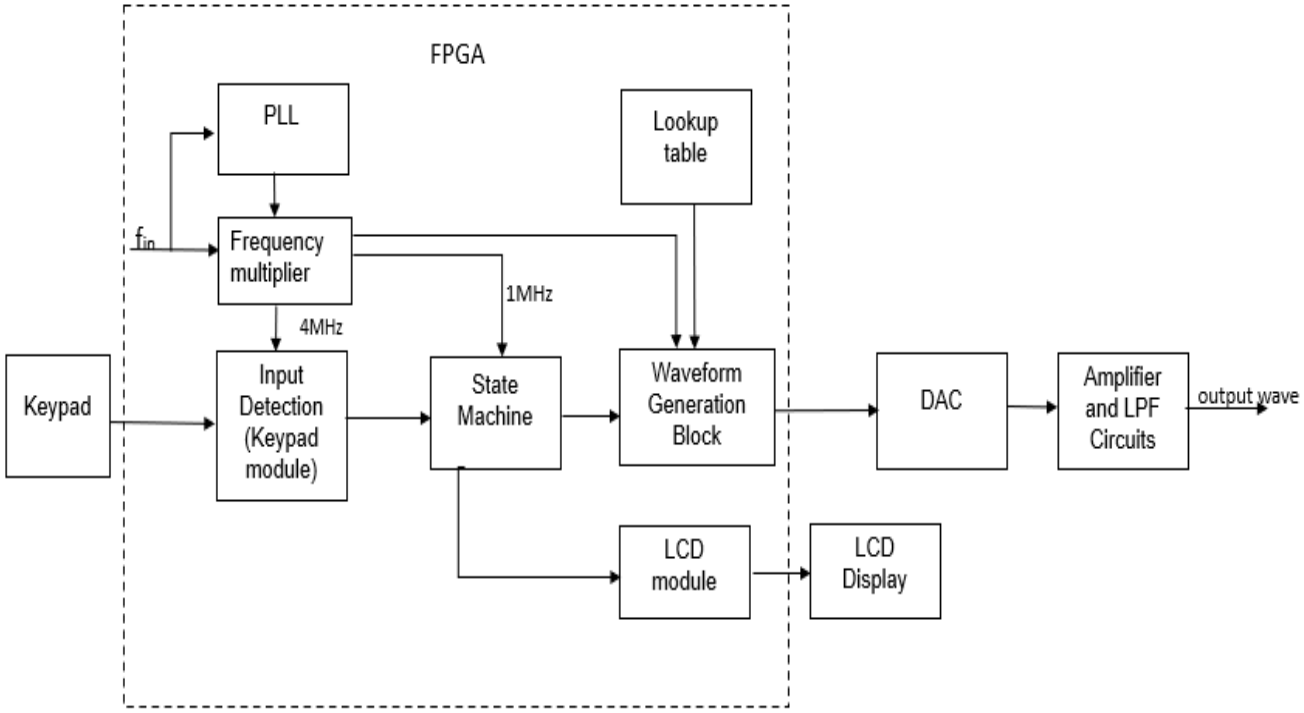

Figure 2.1: Block Diagram of FPGA based waveform generator

Figure 2.1 shows the block diagram of FPGA based waveform generator. The user provides input through the keypad to select the desired waveform type. The keypad module detects the pressed keys and translates them into a digital signal representing the user's selection. This signal is then received by the input detection block within the FPGA, which processes it to identify the selected waveform type. The input detection block also sends a signal to the LCD module, which displays the selected waveform type for user confirmation. The system clock frequency fin which is 50MHz is given as input to the PLL, which multiplies it to generate a higher frequency. The state machine within the FPGA coordinates the overall operation of the waveform generation system. It controls the selection of the appropriate lookup table based on the input from the keypad and ensures the proper sequencing of the waveform generation. The lookup table stores precomputed digital values for different waveforms, and based on the input selection, the

state machine accesses the appropriate lookup table to retrieve the waveform data. The waveform generation block then takes these digital values and uses them to generate the corresponding waveform, with the frequency divider providing an appropriate clock signal to ensure the waveform is generated at the correct frequency. The digital waveform generated is sent to the DAC, which converts it into an analog signal. This analog signal is then passed through an amplifier to increase its strength . The LCD module displays the selected waveform type, providing visual feedback to the user. Finally, the amplified and filtered analog waveform can be observed using an oscilloscope.

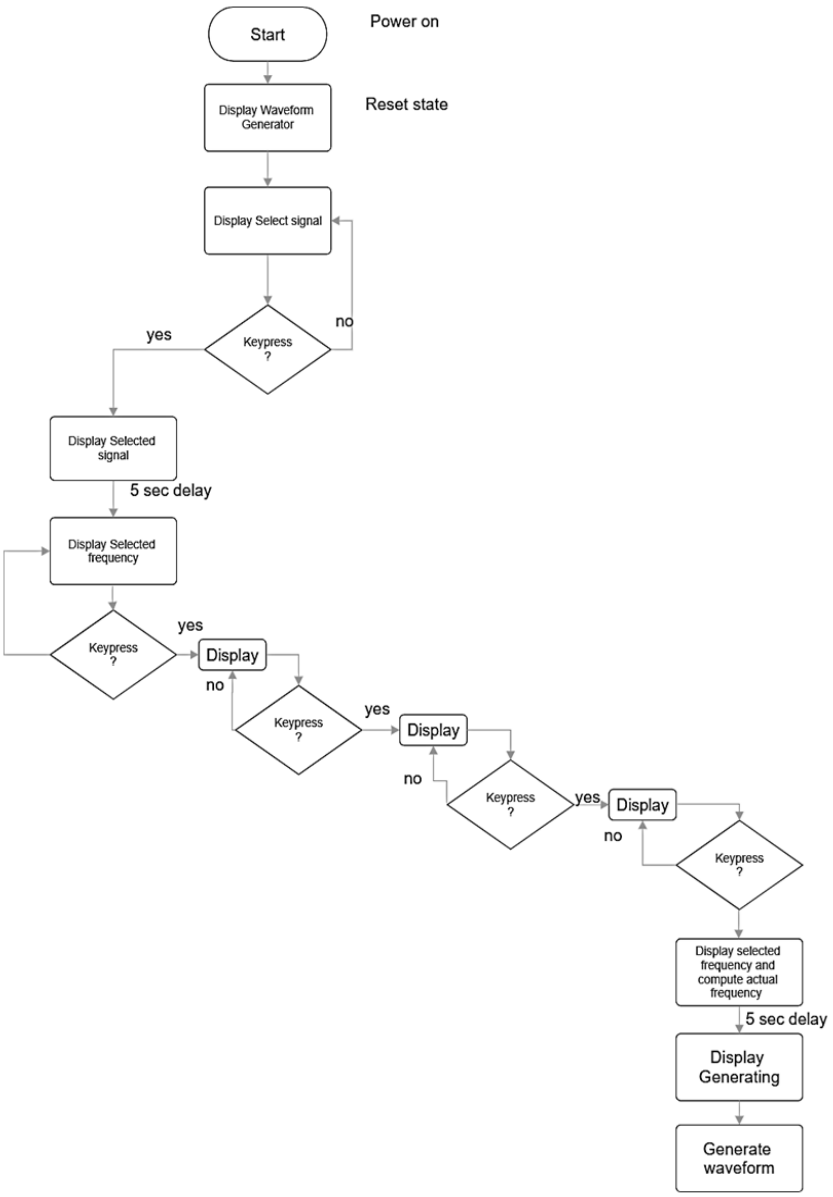

Figure 2.2: Flowchart for FPGA based waveform generator

2.2 Keypad Module

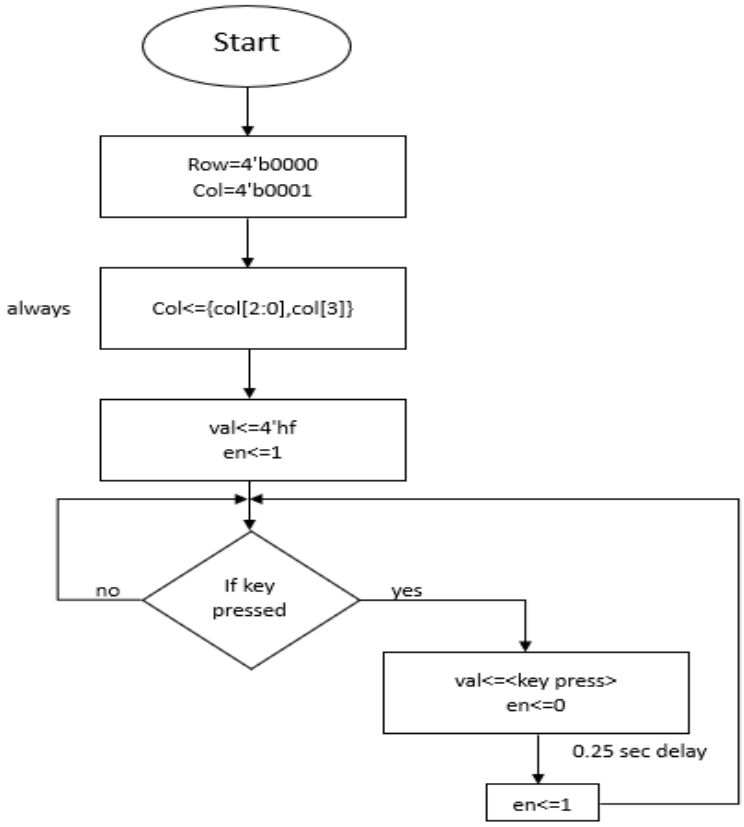

Figure 2.3: Keypad detection

- Figure 2.3 shows the keypad detection. The keypad allows the user to select the desired waveform type. It sends the input to the FPGA, where the selection is detected and used to configure the system, ensuring the correct waveform is generated and outputted.

- The Keypad module scans a 4x4 keypad matrix using a 4 MHz clock, detecting key presses by shifting the col signal across the keypad columns and checking the corresponding row inputs. When a key is pressed, the module outputs a 4-bit code representing the key and sets of flags are triggered and user input is processed. Additionally, a reset signal is triggered when a specific key is pressed, ensuring reliable user input detection and output in the FPGA-based system.

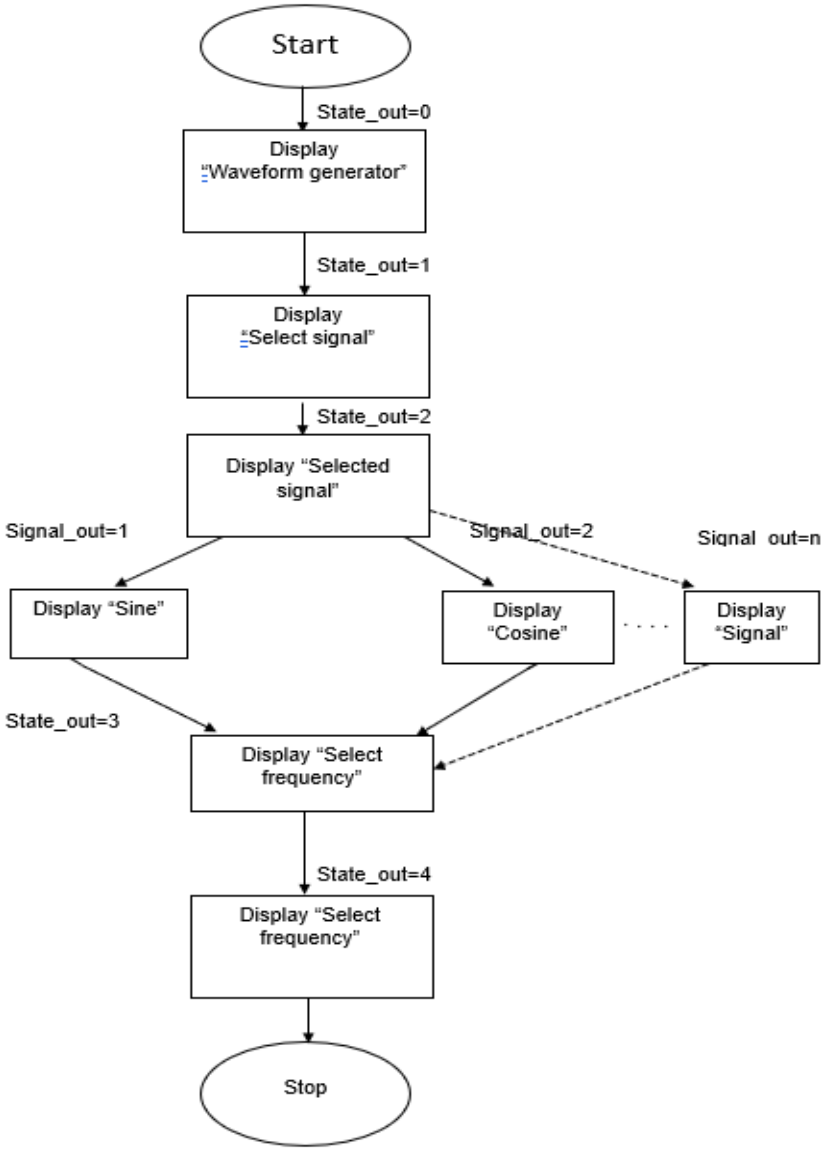

Figure 2.4: Flowchart for LCD module

2.3 LCD Module

- Figure 2.4 shows the flowchart of the LCD module The LCD interface module manages communication between an FPGA and an LCD display, handling the display's initialization and data writing. The module takes a 1 MHz clock signal, a reset signal, and a 2-bit state input, and it produces control signals for the LCD (lcd_e, lcd_rs, lcd_rw) and the data to be displayed.

- lcd_e (Enable) Activates the LCD to latch data or commands. lcd_rs(Register Select) Differentiates between command and data operations. lcd_rw (Read-/Write) Determines whether the operation is a read or write to the LCD.

- The always @(posedge clkdiv) block updates the LCD based on the state. During the s0 state, the module initializes the LCD and writes the predefined

data to it, including display configurations and text strings. The s1 state is for clearing the display and updating it with new data. A delay mechanism ensures data is written correctly by controlling the lcd_e signal (enable) and adjusting the lcd_rs (register select) for command or data writes. The code handles various stages of the data writing process, including delays between write operations to accommodate the LCD's response time.

2.4 State Machine

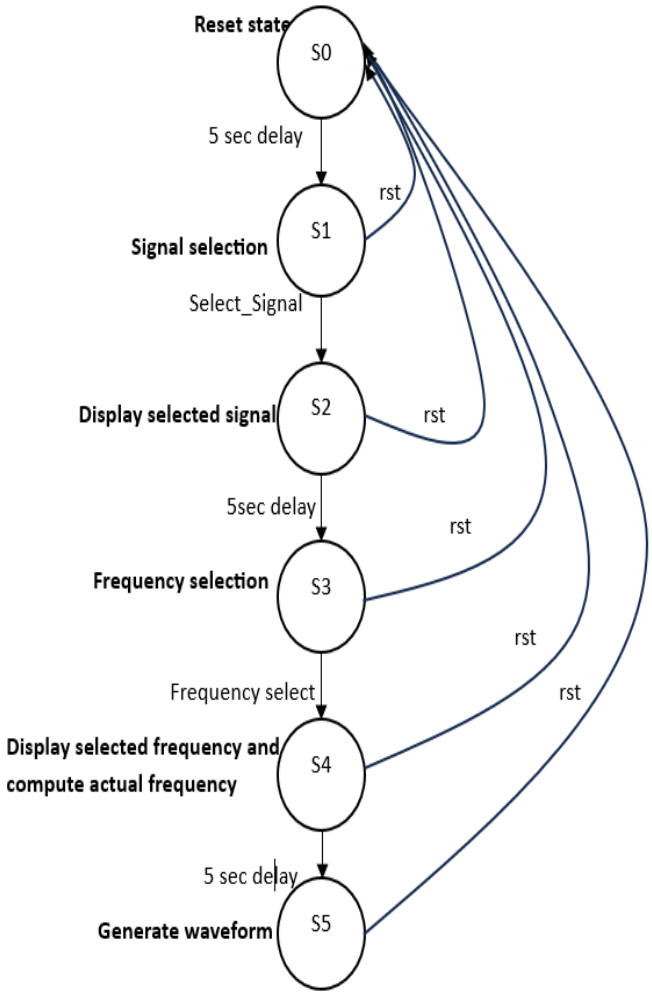

Figure 2.5: State Machine

- The State Machine module in Figure 2.5 defines a finite state machine for managing a waveform generation process through a sequence of defined states. It receives a 1 MHz clock , a reset , and two input signals for selecting a waveform type and frequency . It outputs the current state as a 3-bit signal.

The FSM cycles through six states, each governing a specific part of the process. It begins in S0 , where it waits for a 3-second delay before transitioning to the next state, S1 .In S1, the system waits for the user to choose a waveform type; when signal select flag is asserted, it advances to S2 , where the selected signal is displayed for 5 seconds before moving to S3.

- In S3 , the FSM waits for a frequency choice, transitioning to S4 (Display Selected Frequency) once frequency select flag is asserted. In S4, the selected frequency is displayed for another 5 seconds, after which the FSM moves to S5, the final state. In S5, the FSM remains indefinitely, ready to generate the selected waveform based on the chosen signal and frequency

2.5 Phase Locked Loop

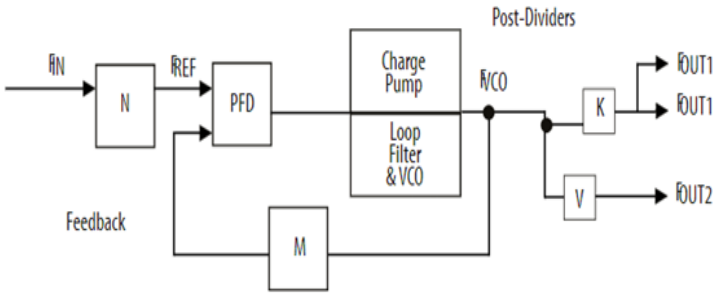

Figure 2.6: Phase Locked Loop

Figure 2.6 shows the block diagram of a PLL. A phase-locked loop (PLL) is a control system that generates an output signal whose phase is fixed relative to the phase of an input signal. Keeping the input and output phase in lockstep also implies keeping the input and output frequencies the same, thus a phase-locked loop can also track an input frequency. Four PLLs are used, and they produce 1 MHz, 4 MHz, 32 MHz and 100 MHz outputs. The PLLs lock onto an input frequency and generate precise clock outputs that drive key parts of the system, including the state machine and input detection modules. These clocks ensure synchronized operation across the FPGA, enabling smooth waveform generation and control. The 1 MHz clock is used for the state machine's timing, while the 4 MHz clock is used in the frequency multiplier and input detection blocks.

2.6 FPGA Design

2.6.1 FPGA Block Diagram

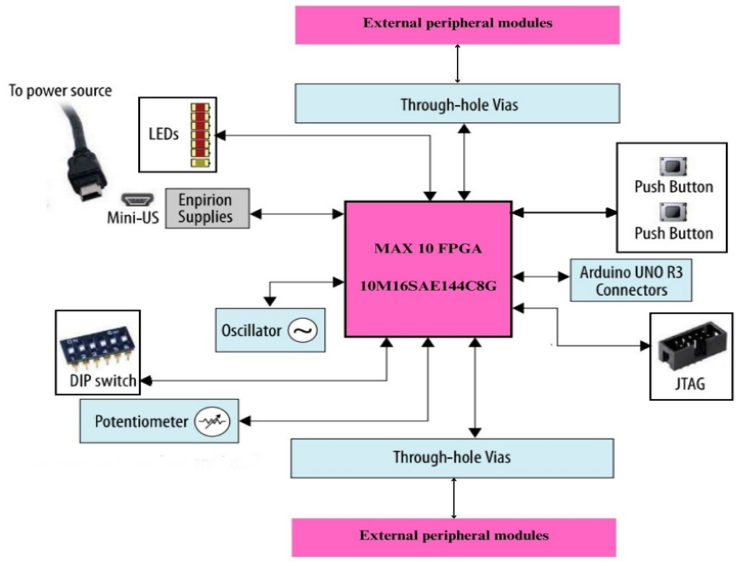

Figure 2.7: FPGA Block Diagram

Table 2.1 gives the MAX 10 FPGA specifications

Table 2.1: FPGA Specifications

Component	Details
FPGA	Intel MAX 10 FPGA (10M08SAE144C8G)
Configuration Circuitry	Supports JTAG headers for programming via Intel FPGA Download Cable, Download Cable II, or Ethernet Cable
Clocking	50 MHz oscillator connected to FPGA global clock input
User I/O	8 Analog I/O, 14 Arduino digital I/O 40 General-purpose I/O
LEDs	5 Red user-defined LEDs, 1 Green LED for USB power status
Push Buttons	SW1: Device-wide reset, SW2: Reconfiguration push button
DIP Switch	User-configurable DIP switch (SW3)
Power Supply	USB-powered from PC or wall jack with green LED (D6) indicating power-on status
Probe Points	TP2 - TP5 for current measurement and power consumption TP1, TP6 - TP9 for voltage verification

2.6.2 FPGA Schematic

The schematic has been partitioned into 7 sections to facilitate easier interpretation due to its large size. They are as follows:

- Main Page:It shows the FPGA (Figure 2.9), specifically the 10M08SAE144C8G IC, along with all the global labels representing the corresponding nets of their respective sections.
- Power Plan: Focused on the power circuitry (Figure 2.10), this area centres around the Empirion step-down switching regulator, namely the EPI5388QI. This component converts the 5V input from the USB to 3.3V (Figure 2.8), which is then distributed to the FPGA and other components on the board.

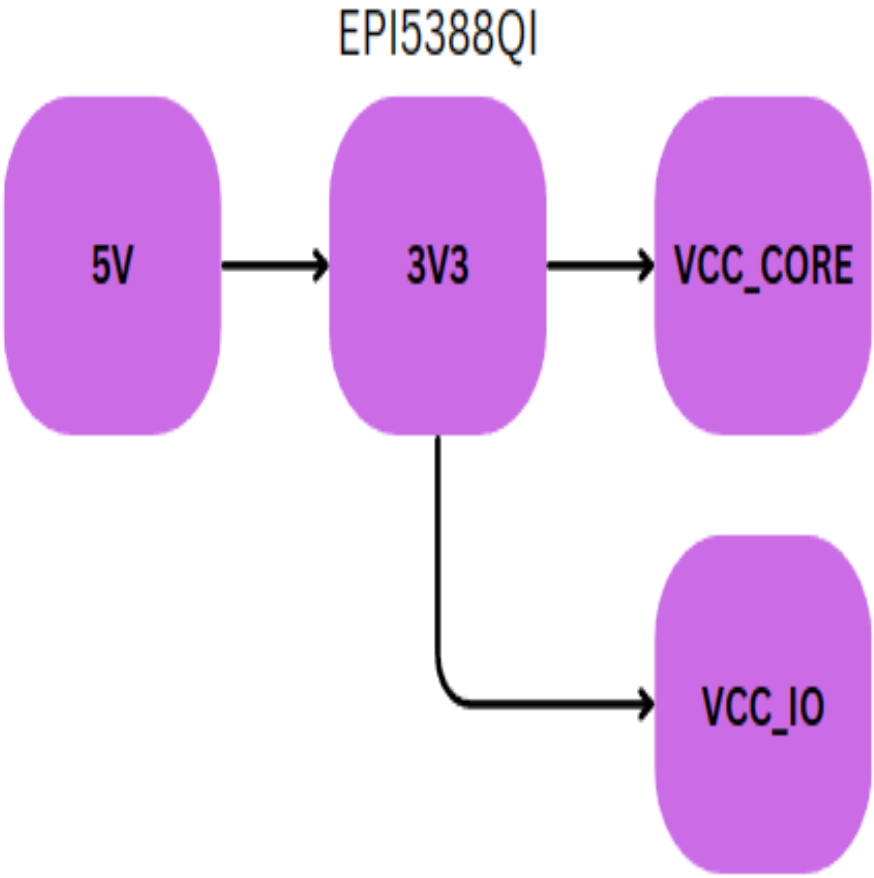

Figure 2.8: Power distribution system

- Analog Section: This segment encompasses analog circuitry centred around two op-amp ICs, the LM2902. These ICs facilitate the reading of analog inputs, which are subsequently supplied to the ADC through a series of RC filters (Figure 2.11).

- Arduino Connection: Here, pin headers are arranged in the form factor of an Arduino Uno, facilitating compatibility and connectivity with Arduino shields and accessories (Figure 2.12).

- Bank 3, Bank 4, Bank 6, and Bank 7 Breakout: This section includes pin headers designed to extend connections from the FPGA to peripheral boards, enhancing versatility and expandability. One general purpose clock with 50MHz oscillator is provided to the FPGA global clock inputs at Bank 2 for general FPGA design (Figure 2.13-2.14).

- Configurations: This area encompasses additional accessories such as the JTAG programming header, push buttons, dip switches, and LEDs, providing essential functionality and control options (Figure 2.15).

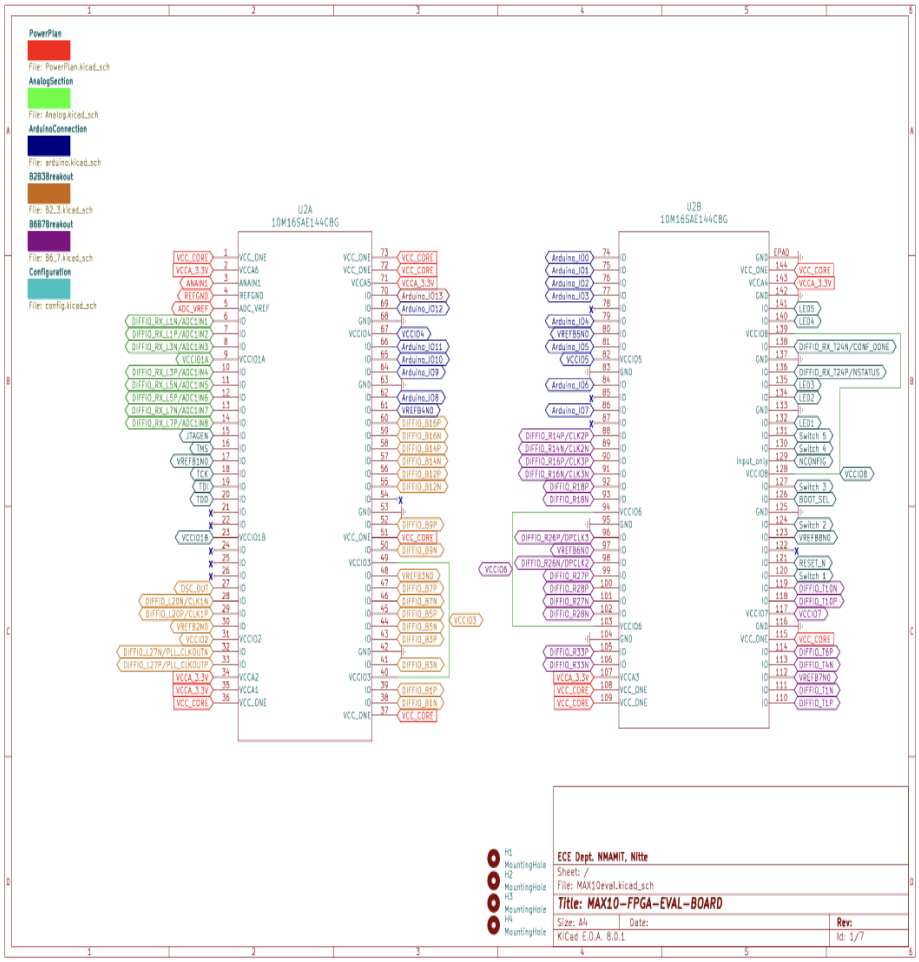

Figure 2.9: Main page schematic

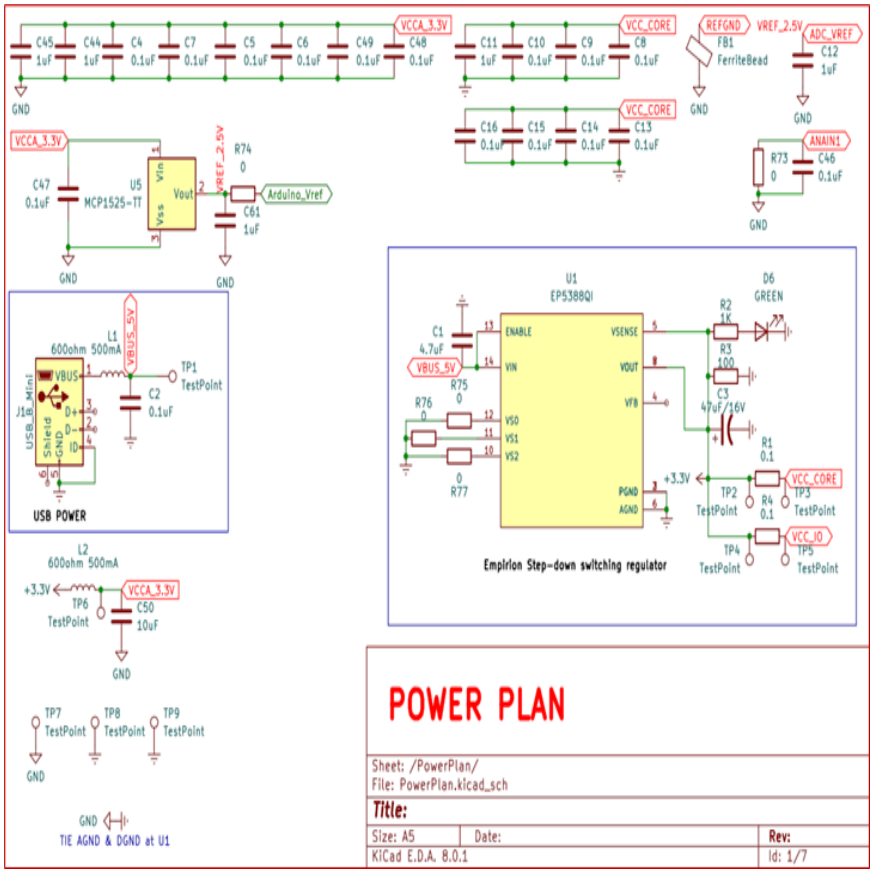

Figure 2.10: Power plan

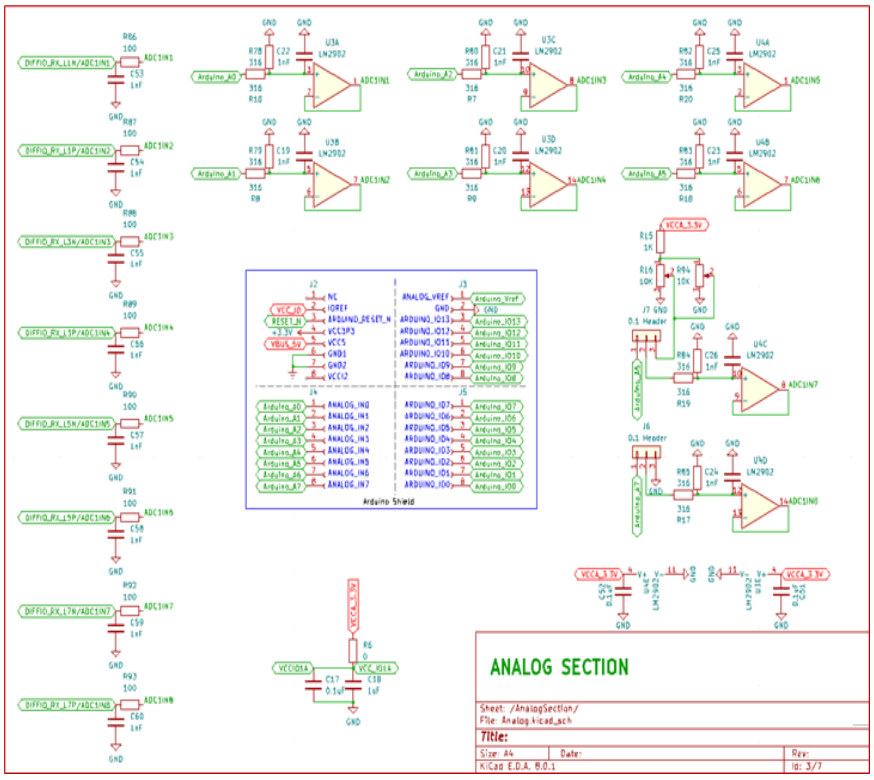

Figure 2.11: Analog section

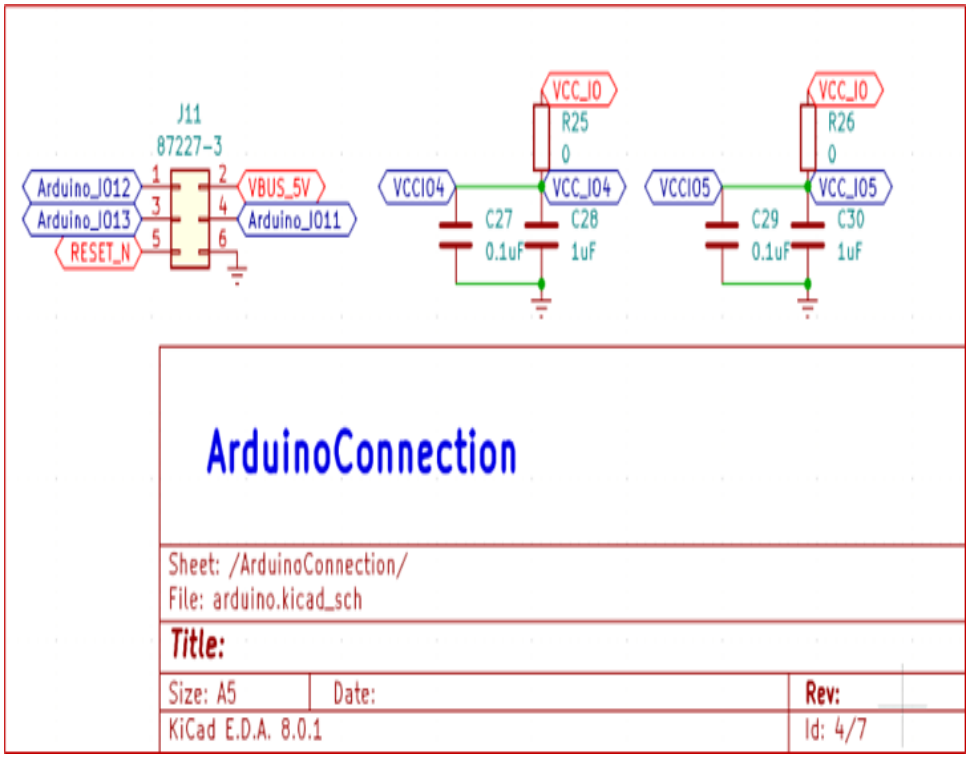

Figure 2.12: Arduino connection

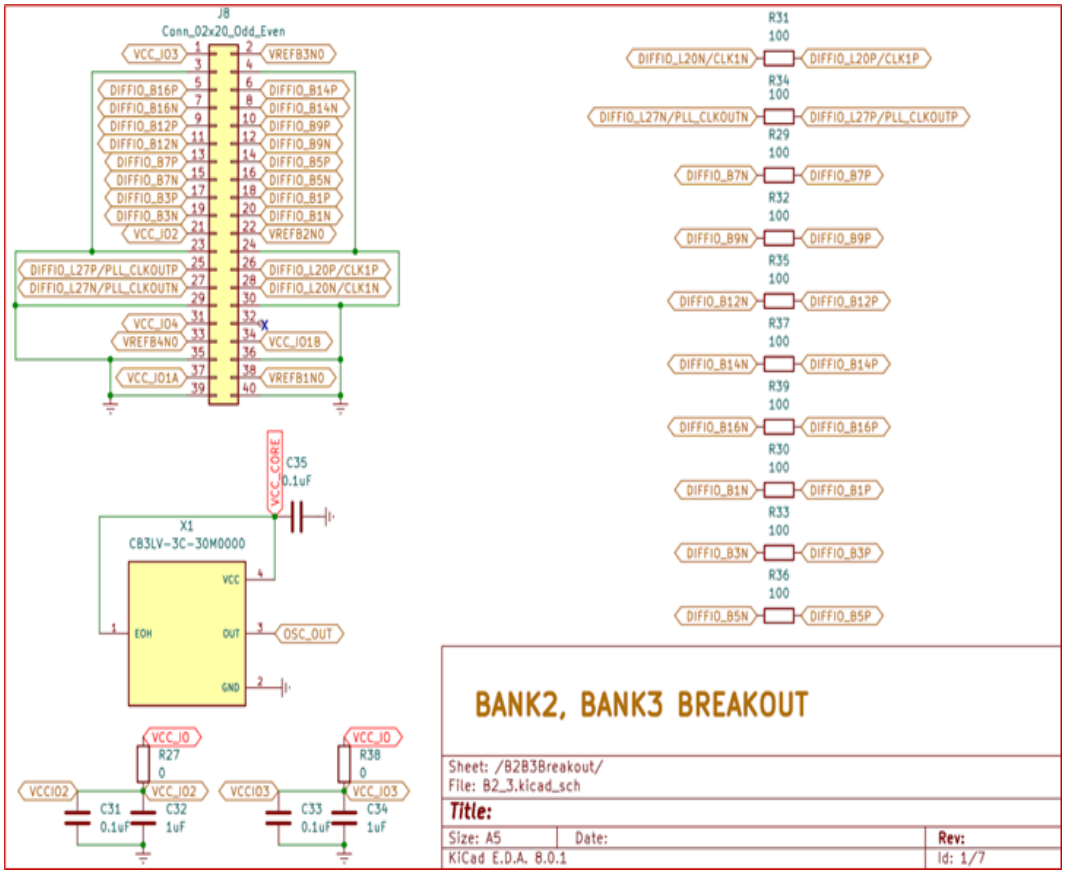

Figure 2.13: Bank 2, Bank 3 Breakout

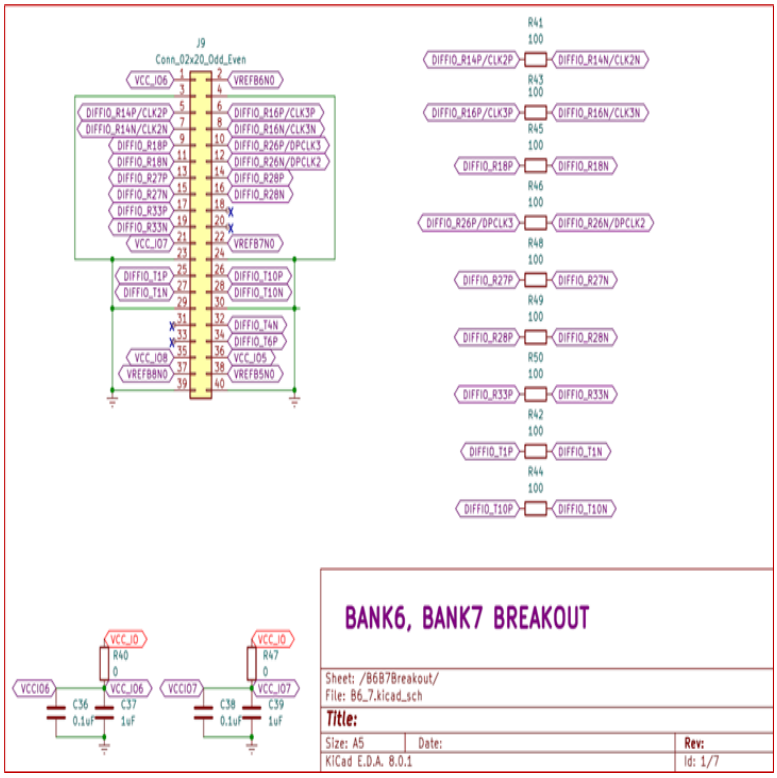

Figure 2.14: Bank 6, Bank 7 Breakout

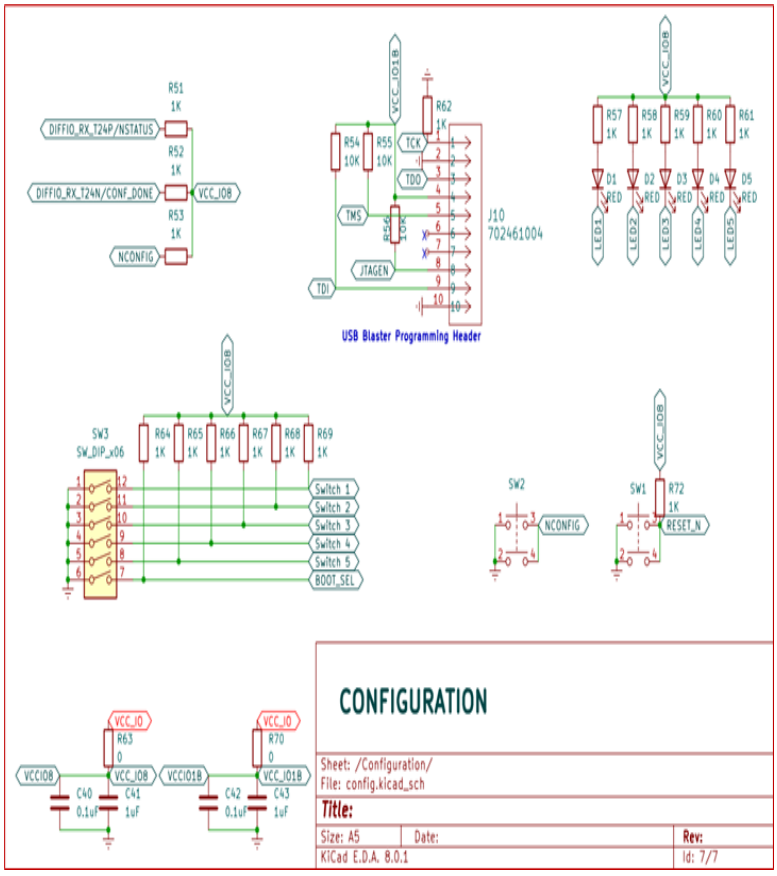

Figure 2.15: Configuration

2.6.3 FPGA Layout

The PCB layout for the above schematic is created (Fig 2.15-2.17). The physical stack-up of the board consists of four copper layers with the following design rules

- Clearance = 0.16 mm
- Track Width = 0.15 mm
- Via Size = 0.625 mm
- Via Hole = 0.3 mm
- µVia Size = 0.3 mm
- µVia Hole = 0.1 mm
- DP Width = 0.2 mm
- DP Gap = 0.25 mm

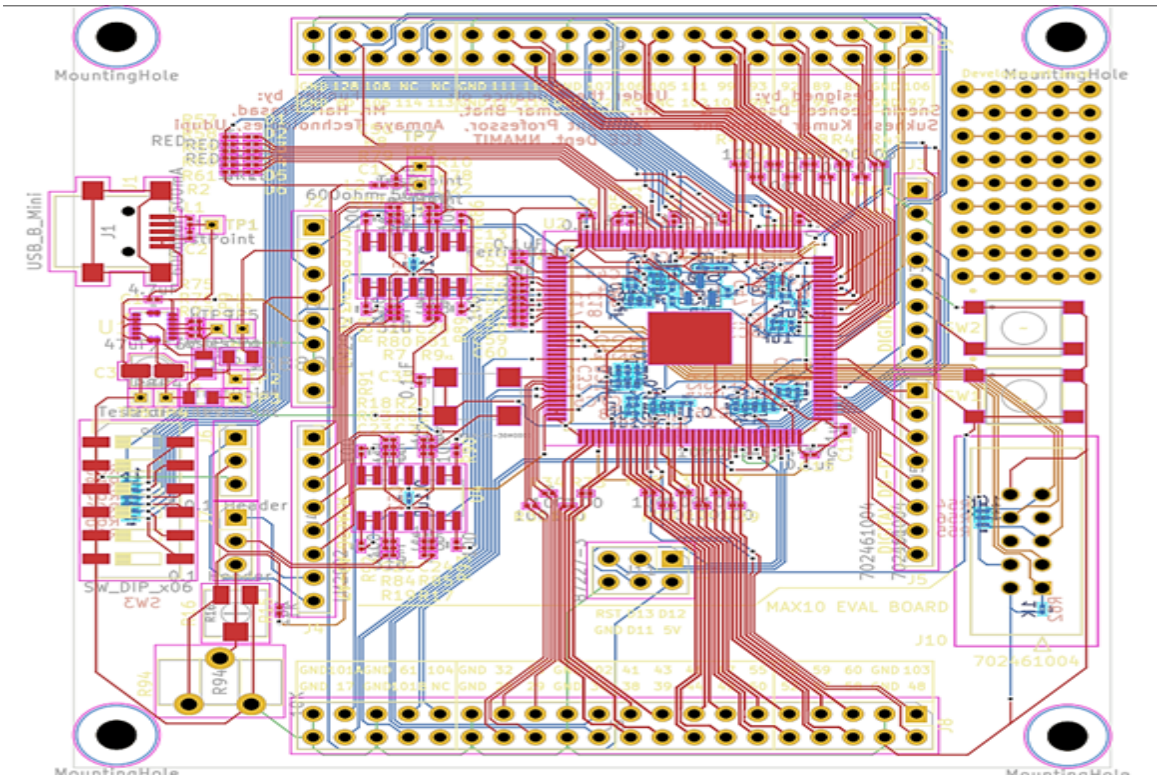

Figure 2.16: Layout

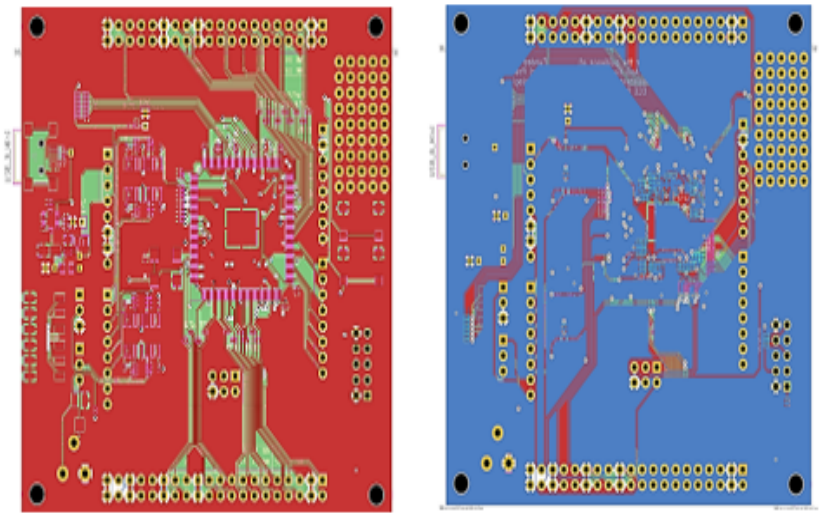

Figure 2.17: Front and Back copper view

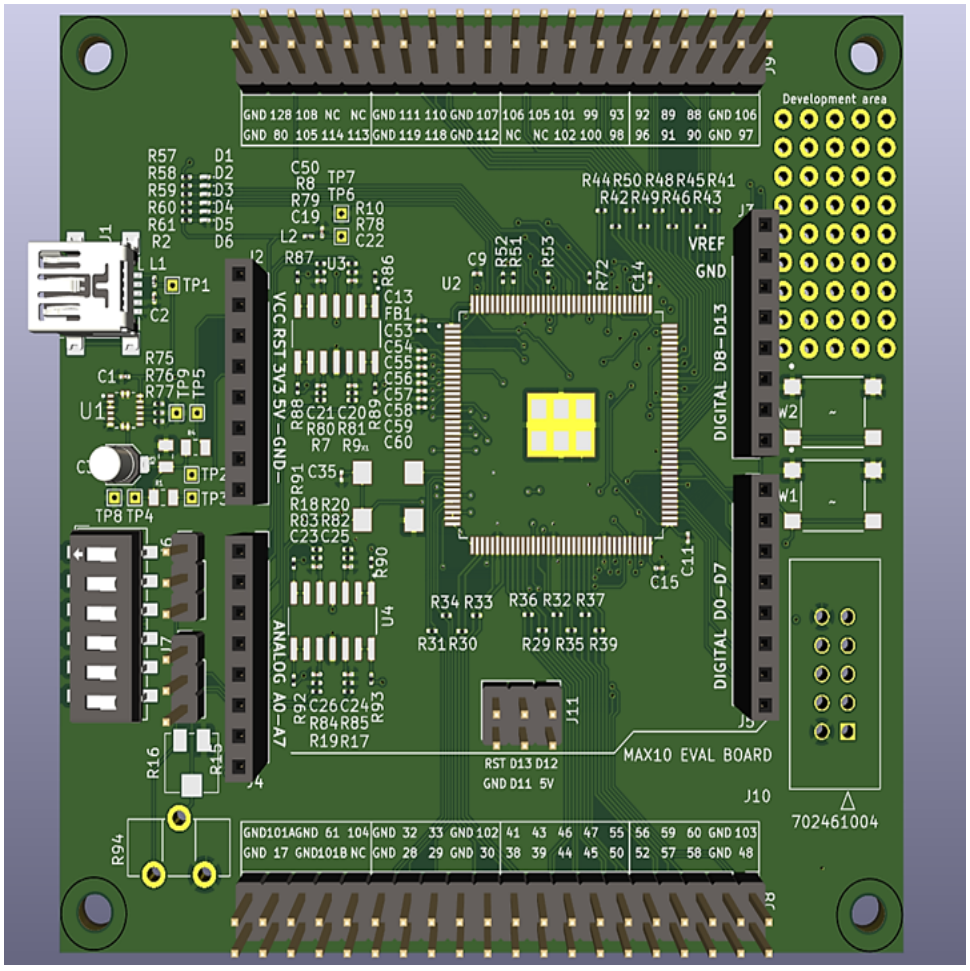

Figure 2.18: 3D Model of PCB

2.7 Waveform Generator Peripheral Board

A peripheral board has been designed using KiCad is shown in Figure 2.27 and it contains the following :

4x4 Keypad Matrix:

Figure 2.19 shows a 4x4 matrix keypad that allows the user to input selections (such as waveform type and frequency). The matrix is constructed with rows and columns connected to switches (S1-S16) that detect key presses.

16x2 LCD Display:

Figure 2.20 shows the interface of the LCD display .The display likely provides real-time feedback to the user about the waveform being generated, frequency, etc. The connections to the LCD include control lines for enabling the display and handling data transmission.

High-Speed DAC (AD9708):

Figure 2.21 shows AD9708 digital-to-analog converter (DAC) that converts digital signals from the FPGA into analog signals. The DAC outputs are used for generating the waveform based on the digital values retrieved from the FPGA's lookup tables (LUTs). The figure 2.22 shows the functional block diagram of DAC which is used to construct DAC circuit. Figure 2.23 shows DC differential coupling using OpAmp. The differential current output of dac is converted to differential voltage output and then AD8066 opamp is used to convert it into a single ended signal, by using the opamp as differential amplifier.

Amplifier:

Figure 2.24 shows the circuit for converting the output current from the DAC into a voltage and amplifying it. The amplifier IC is an AD8066, a high-performance op-amp, which likely ensures the generated analog waveform is sufficiently amplified for the next stage. The figure 2.25 shows Amplifier Circuit .The output from differential amplifier is given to an inverting opamp to control the voltage level to the opamp , Rf is replace by a 50K pot to give an option to vary the gain of the opamp

2x20 Pin Header to Connect to FPGA:

This connector is where the FPGA board interfaces with the peripheral board. It includes various power and signal lines, such as clock signals, control signals, and data lines. Figure 2.26 shows the Layout of Waveform Generator board

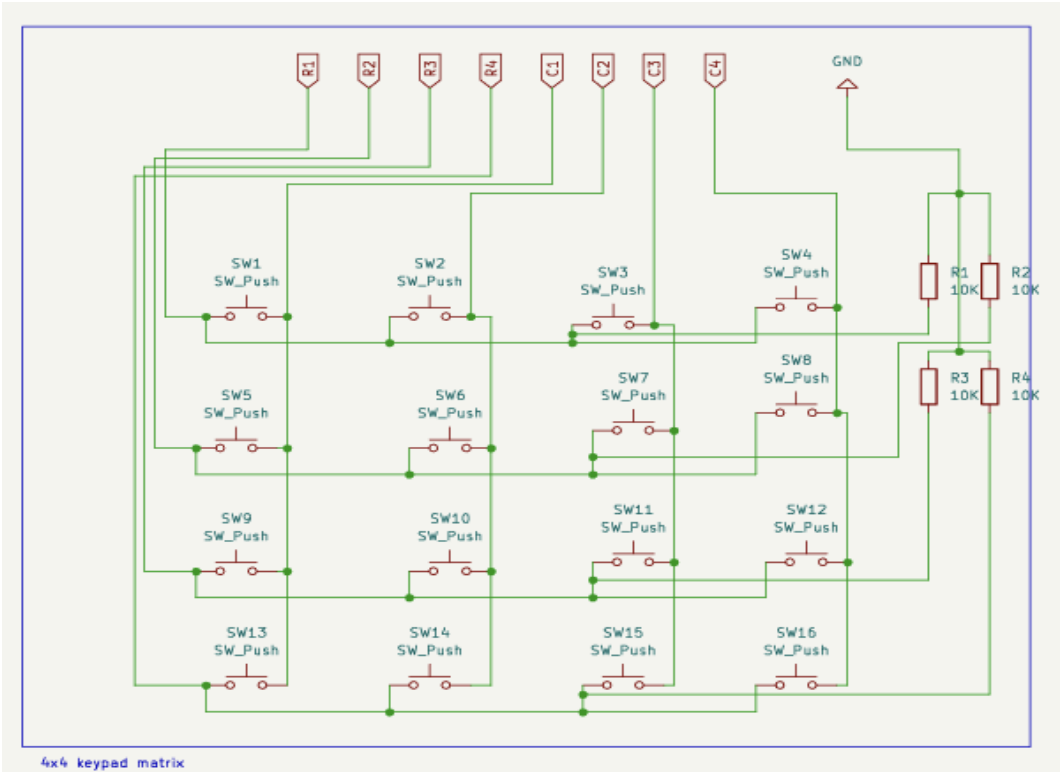

Figure 2.19: 4x4 Keypad Matrix schematic

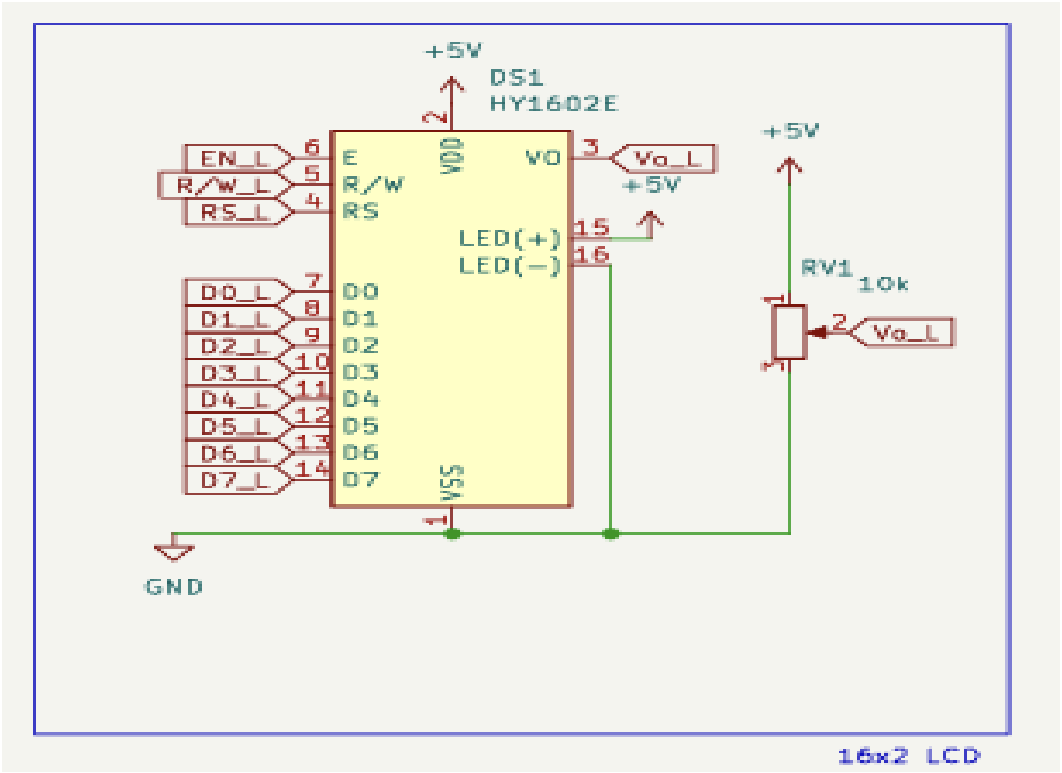

Figure 2.20: 16x2 LCD Schematic

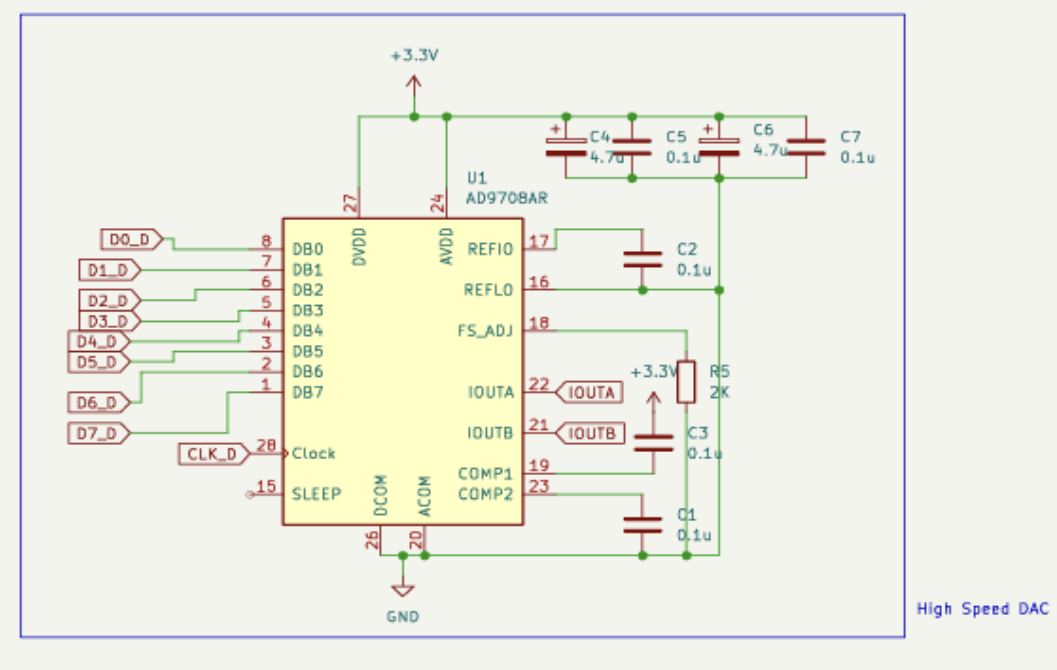

Figure 2.21: High speed DAC

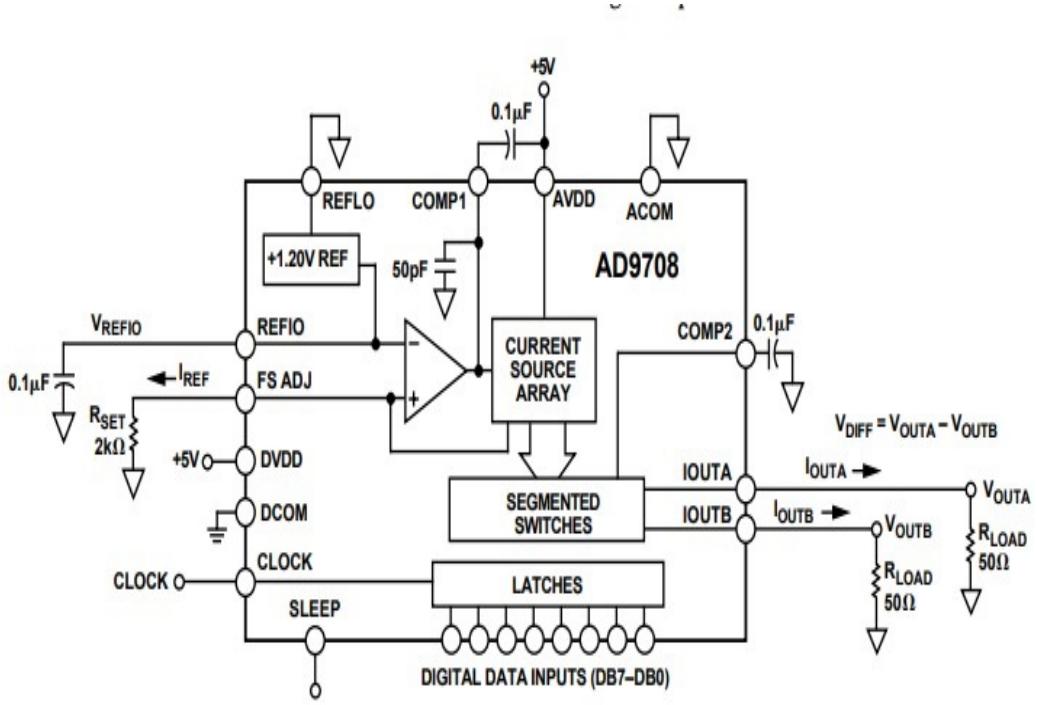

Figure 2.22: Functional Block Diagram of DAC

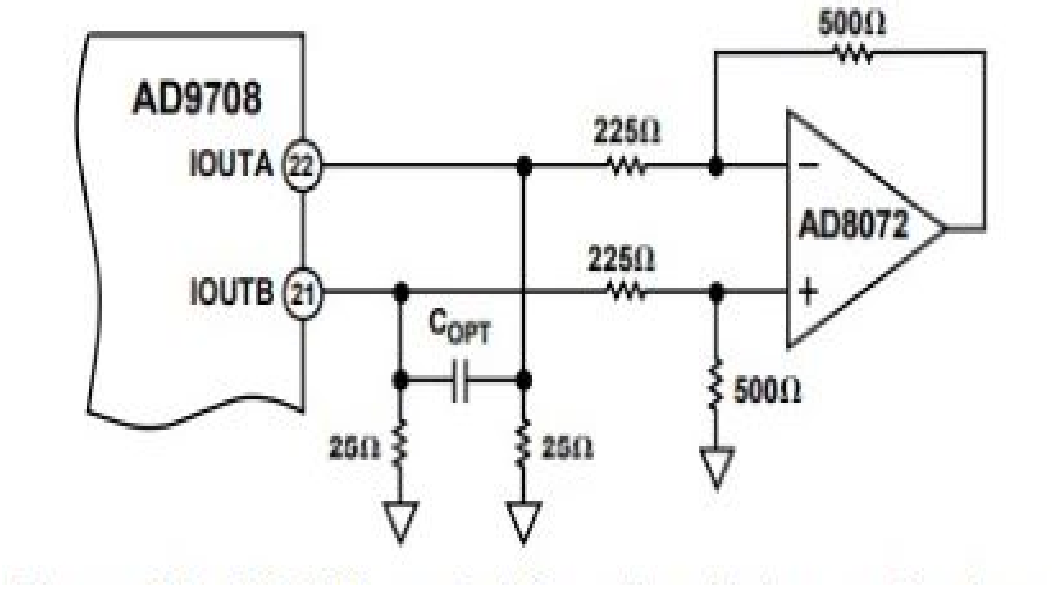

Figure 2.23: DC differential Coupling using OpAmp

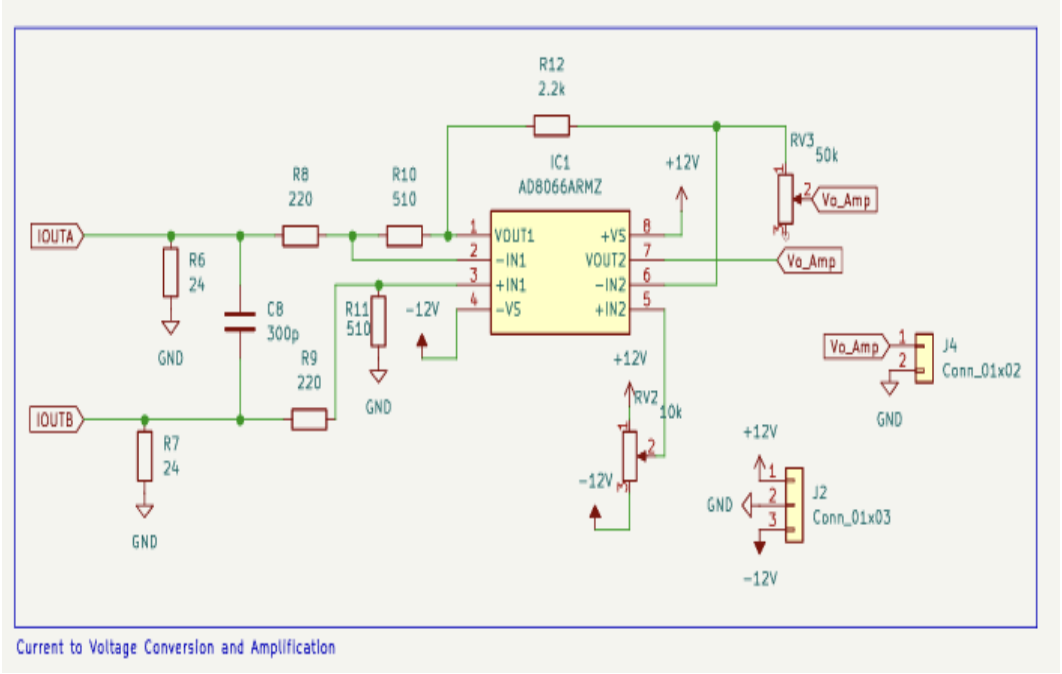

Figure 2.24: Amplifier Schematic

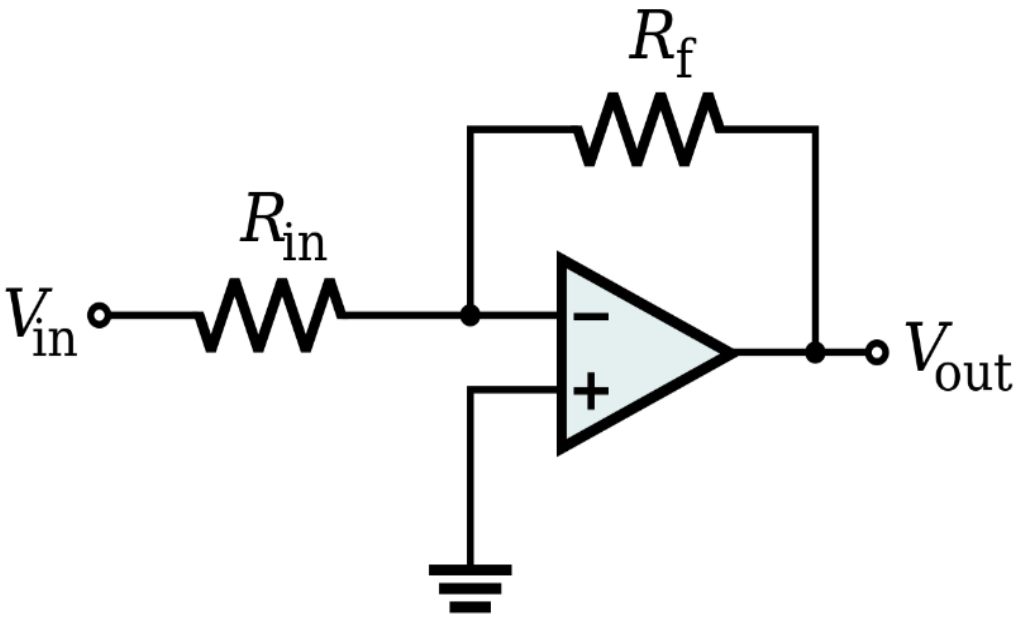

Figure 2.25: Amplifier circuit

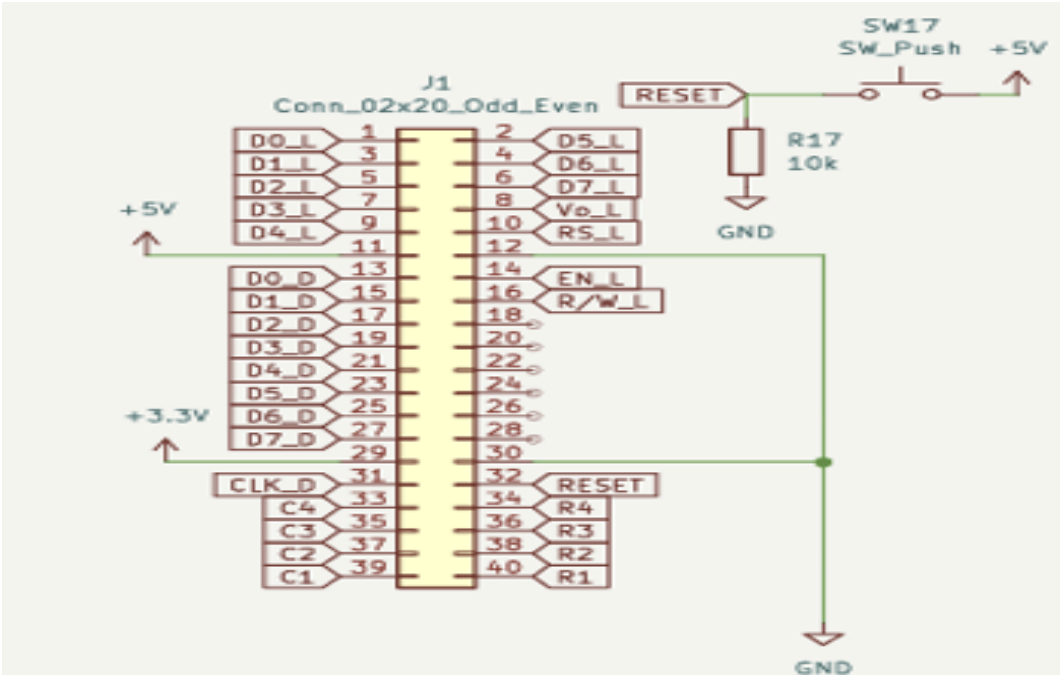

Figure 2.26: 2x20 pin headers schematic

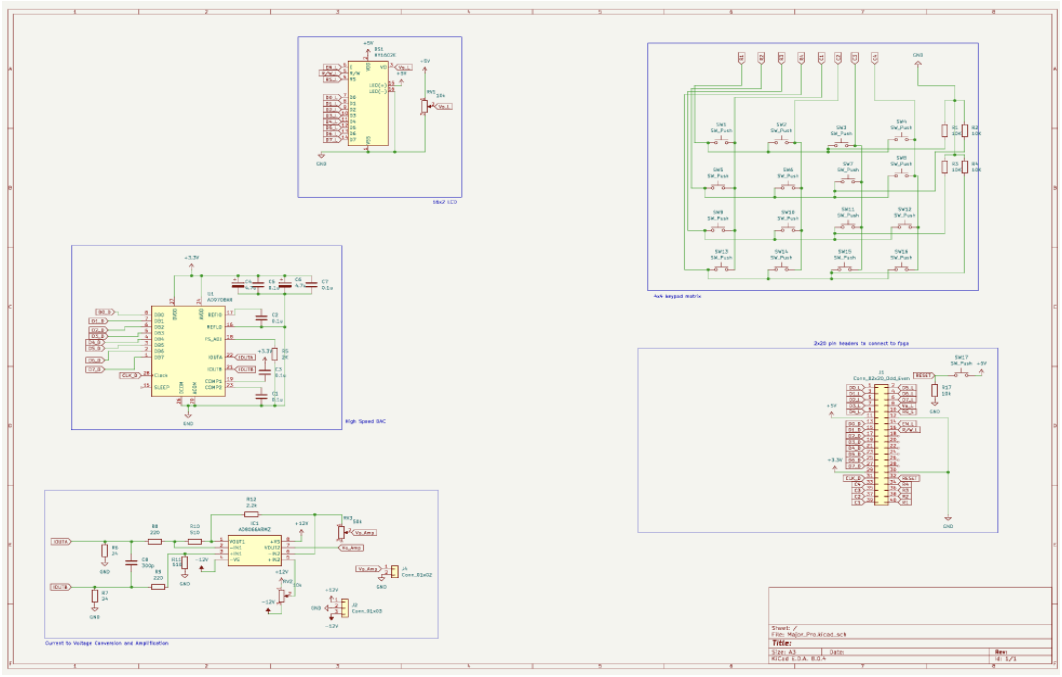

Figure 2.27: Schematic of Waveform Generator Board

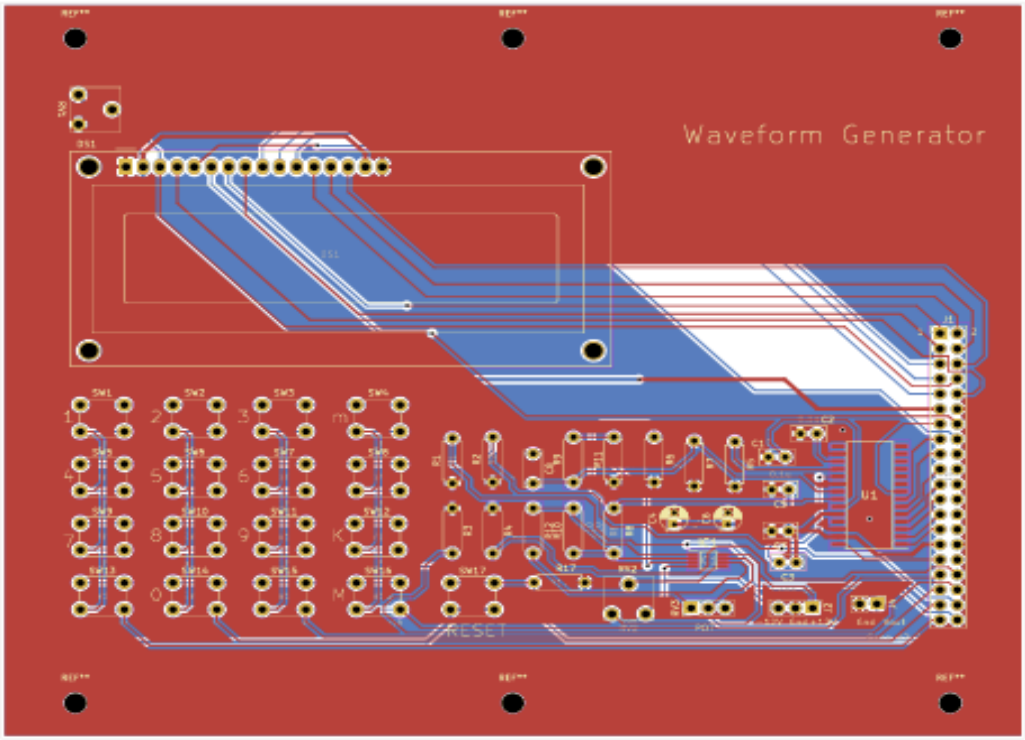

Figure 2.28: Layout of Waveform Generator Board

Chapter 3
Hardware Description

3.1 10M08SAE144C8G FPGA

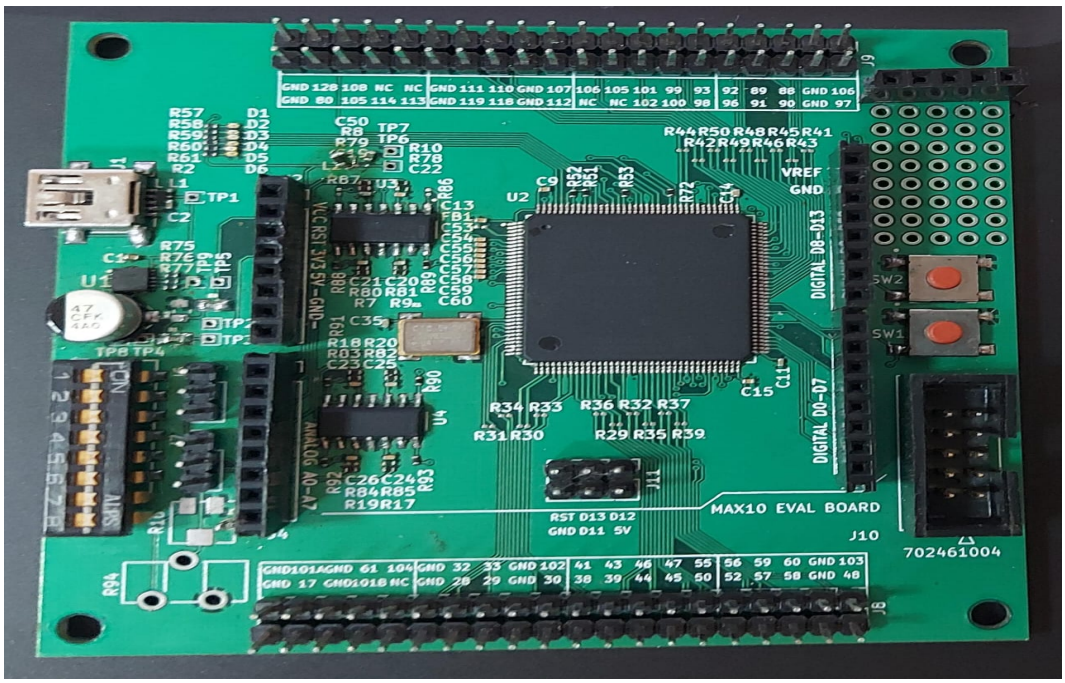

Figure 3.1: 10M08SAE144C8G FPGA

The selected FPGA for the system is the Altera MAX 10 (10M08SAE144C8G), a 144-pin IC with 8000 Logic Elements, 562 Kbits of RAM, and a built-in 12-bit ADC capable of 1 million samples per second. This device is ideal for balancing cost and efficiency in building an evaluation board. The board features the Intel MAX 10 FPGA (10M08SAE144C8G) as the core, with JTAG headers for programming via Intel FPGA Download Cables or Ethernet. A 50 MHz oscillator provides clocking, and the user I/O includes 8 analog inputs, 14 Arduino digital I/Os, and 40 general-purpose I/Os. It also has 5 user-defined red LEDs, a green power LED, push buttons for device-wide reset and reconfiguration, and a user DIP switch. Power is supplied via USB, with probe points available for manual current measurement and voltage verification. These features, along with built-in clocking and configuration circuitry, make this board highly functional for FPGA-based projects.

3.2 4X4 Matrix Keypad

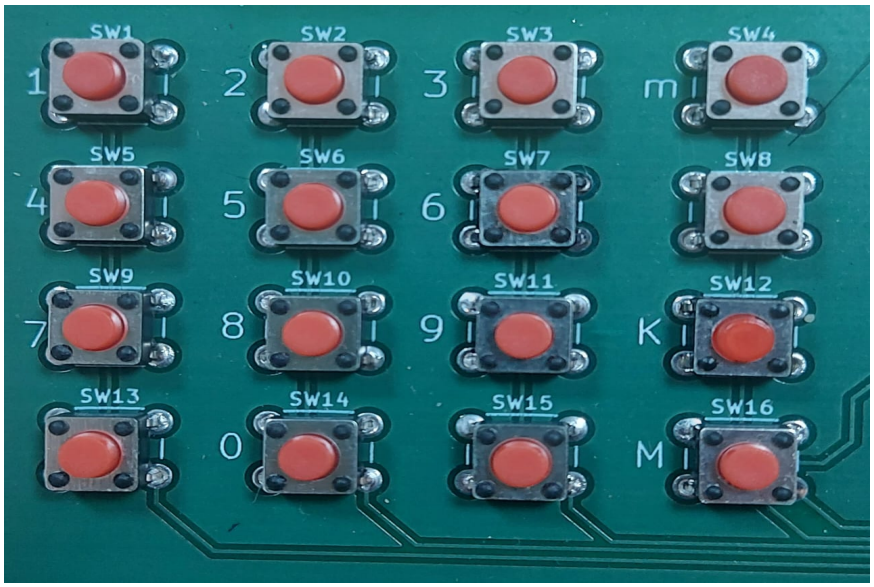

Figure 3.2: 4X4 Matrix Keypad

The 4x4 matrix keypad shown in figure 3.2 serves as the primary input interface for the user to select both the type and frequency range of the waveform. This keypad consists of 16 keys arranged in a grid of four rows and four columns, offering 16 unique input combinations. Each key on the keypad can be assigned a specific function, such as selecting a waveform type (e.g., sine, square, triangle) or adjusting the frequency range, making it versatile for controlling the waveform generator.To detect key presses, the keypad module continuously scans the keypad matrix. This is achieved by applying a scanning technique where the columns of the keypad are sequentially activated (one column at a time) using a 4 MHz clock signal. . When a key is pressed, the connection between the corresponding row and column is detected, allowing the system to identify which key has been pressed.

3.3 16X2 LCD Display

The 16x2 LCD shown in figure 3.3 plays a crucial role in guiding the user through the process of selecting and inputting waveform parameters via the 4x4 matrix keypad. This display provides real-time feedback to the user, showing the current state of the waveform selection and frequency input, making the interaction intuitive and efficient. The LCD module is responsible for managing the communication between the FPGA and the display. It handles tasks such as initialization, data writing, and ensuring the proper functioning of the LCD. The module operates using a 1 MHz clock signal, and it processes user inputs based on a 2-bit state signal that dictates the display's behavior.

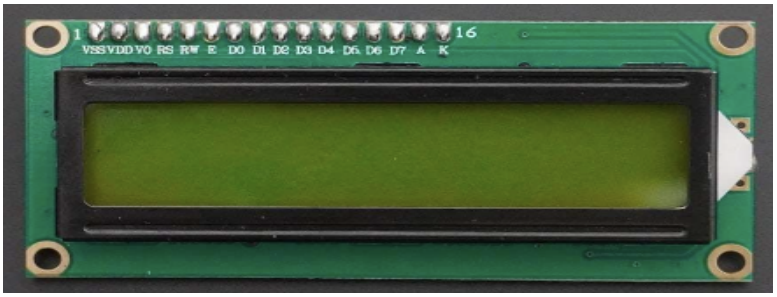

Figure 3.3: 16X2 LCD Display

3.4 AD9708

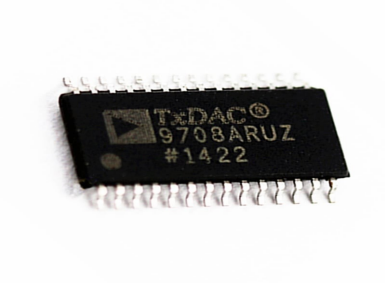

Figure 3.4: AD9708

The AD9708 DAC in this system converts the digital waveform sequences generated by the FPGA (10M16SAE144C8G) into analog signals. Its high-speed conversion capability ensures that the waveforms, whether sine, cosine or others, are accurately translated from digital to analog without compromising accuracy.

3.5 AD8066

Figure 3.5: AD8066

The AD8066 amplifies this signal generated by the DAC to the desired level, enhancing its driving capability. Additionally, it contributes to the low-pass filtering process, which smooths the output waveform by removing high-frequency noise or unwanted components.

Chapter 4
Software Description

4.1 Quartus Prime Design Software

Quartus Prime is a comprehensive FPGA (Field-Programmable Gate Array) design software developed by Intel (formerly Altera) for the design, simulation, synthesis, and verification of digital systems. It supports a wide range of FPGA and CPLD devices, particularly Intel FPGAs such as Cyclone, Arria, and Stratix families.
The Quartus Prime Design Software is used to design, simulate, and implement the logic necessary to interface the 4x4 matrix keypad, control the 16x2 LCD, and manage the waveform generation and frequency selection tasks on the FPGA.

4.2 SignalTap Logic Analyzer

The SignalTap Logic Analyzer is an integrated tool within Intel's Quartus Prime software used for real-time debugging and verification of FPGA designs. It allows you to monitor and capture the internal signals of your FPGA during operation without the need for external test equipment like oscilloscopes or logic analyzers.
This tool is used for monitoring and analyzing signals within the FPGA design allowing engineers to capture waveform data, verify the accuracy of waveform generation, and identify timing issues or logic errors.

4.3 ModelSim

ModelSim is a popular simulation tool used for verifying HDL (Hardware Description Language) designs. It supports Verilog, VHDL, and SystemVerilog, making it a go-to solution for debugging and verifying digital circuits before they are implemented on an FPGA or ASIC. ModelSim is often used alongside Intel Quartus Prime and other FPGA development tools. This simulation tool is used for verifying the functionality of FPGA designs,debug the logic, test the behavior of waveform generation, and ensure that the design meets the required specifications.

4.4 Generation of Look-Up Tables Using MATLAB

In MATLAB, the waveforms were constructed for 1V Ampitude and 1Hz Frequency, and shifted above the x-axis by a value of 1, then the signal was sampled at a frequency of 32Hz and normalizing the amplitude to 255.

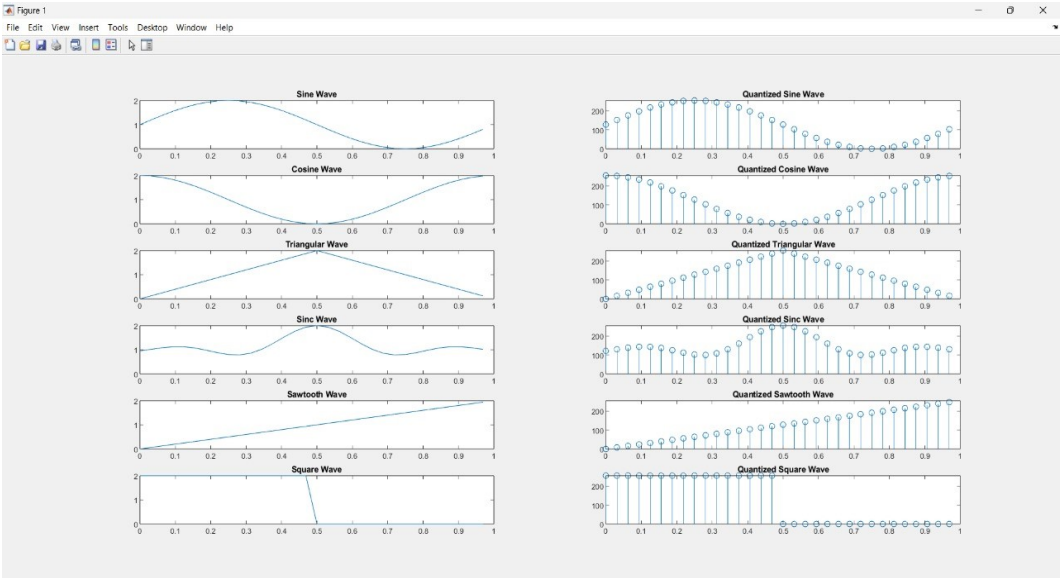

Figure 4.1: Sampling of waveforms

Chapter 5
Results and Conclusions

5.1 Setup

Figure 5.1 shows the final setup of waveform generator using FPGA This setup includes an FPGA evaluation board connected to a waveform generator board using jumper cables. The FPGA board(MAX-10), handles waveform generation and is interfaced with the peripheral for user inputs and signal outputs. The waveform generator board features a 4x4 keypad for input selection, a 16x2 LCD for displaying feedback, and several ICs for signal processing. The system allows real-time control of waveform parameters via the keypad, while the FPGA processes inputs and generates analog waveforms, likely sent through a DAC and amplifier for output.

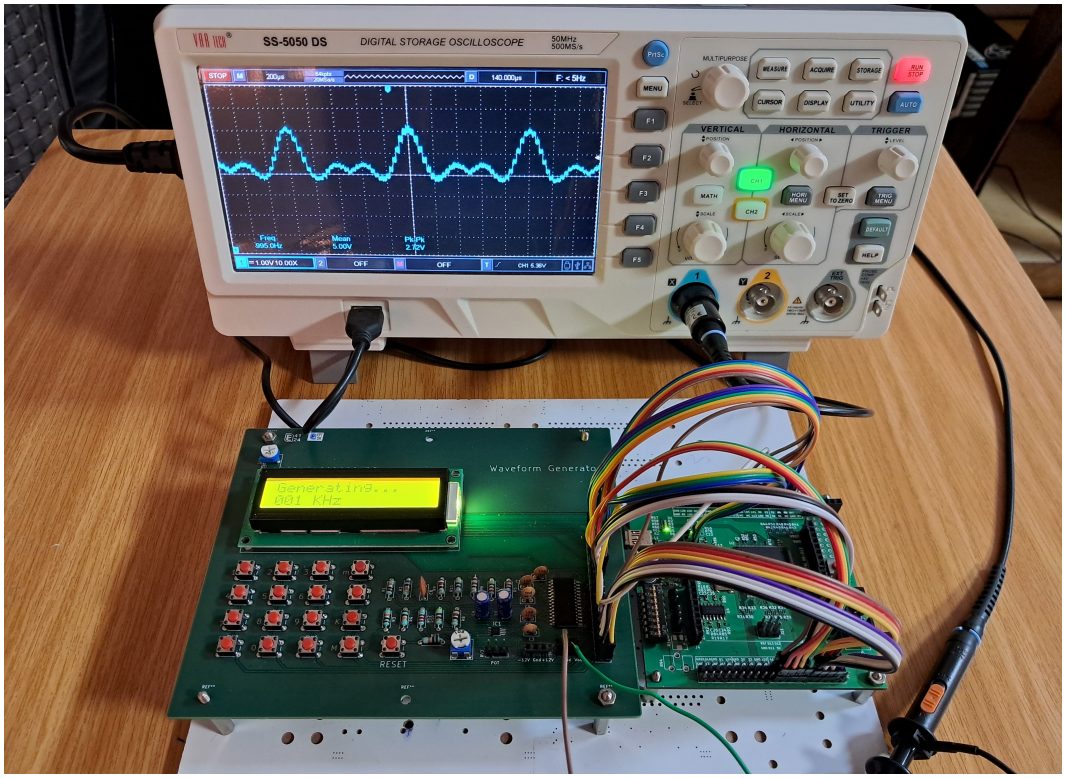

Figure 5.1: Setup

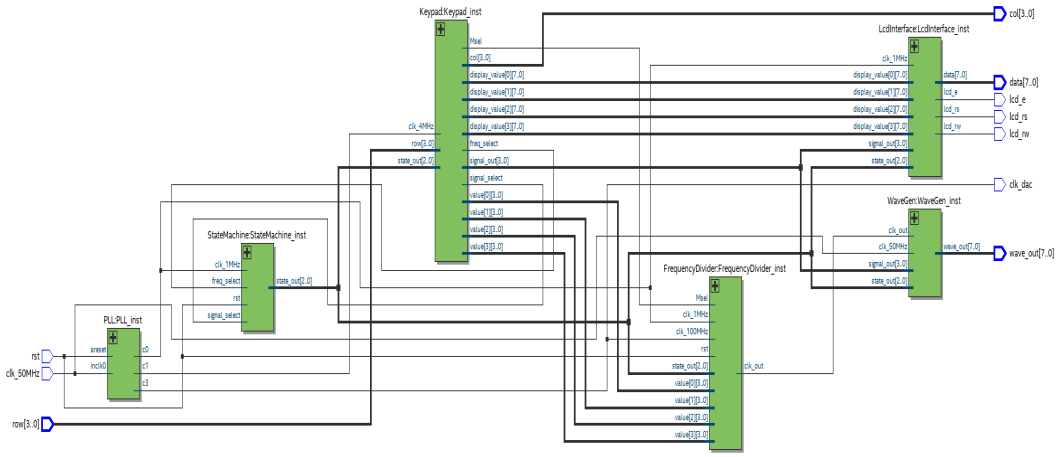

Figure 5.2: Top Level Module

5.2 Top Level Module

The above figure 5.2 shows the Top Level Module of waveform Generator.It consists of following blocks:

PLL_inst (Phase-Locked Loop Instance):

This module generates clock signals of different frequencies. The input clock clk_500 MHz is used to derive multiple clock outputs (c0, c1, and c3) at various frequencies. The PLL ensures that the system components (waveform generator, state machine, and peripherals) are synchronized.

StateMachine_inst:

This is the central Finite State Machine (FSM) that manages the system's operation. It processes user inputs (such as the waveform and frequency selection) and generates control signals (state_out[2:0]) to drive the other components. Inputs like clk_1MHz, freq_select, signal_select, and rst (reset) influence the FSM's behavior, determining how it controls the waveform generation and display.

Keypad_inst:

This module interfaces with the 4x4 keypad matrix, which allows the user to input selections. The row and column signals (row[3:0], col[3:0]) indicate which button is pressed. The outputs (display_value, freq_select, signal_select, value[3:0]) convey the user's selection to the rest of the system, controlling the type of waveform and the frequency to be generated. The keypad runs at (clk_4MHz to scan the key matrix efficiently.

FrequencyDivider_inst: This module divides the clock frequencies to generate a range of lower-frequency clocks clk_out . These clocks can be used for different

parts of the system, particularly for timing the waveform generation at different frequencies.
Inputs like clk_1MHz , clk_100MHz , Msel, and the state signals (state_out[2:0]) determine how the frequencies are divided and distributed.

LCDInterface_inst:
This module interfaces with the 16x2 LCD display. It drives the control signals (lcd_e, lcd_rs, lcd_rw) and data lines (data[7:0]) required to display information such as the selected waveform, frequency, and system state. Inputs include clk_1MHz for timing and display values from the Keypad_inst and StateMachine_inst, which it uses to update the screen.

WaveGen_inst:
Based on the user's selections signal_out[3:0] and state_out[2:0], it generates the appropriate waveform (e.g., sine, cosine, triangular, or square). It takes the base clock clk_50MHz , control signals from the state machine, and outputs the digital waveform signal (wave_out[7:0]), which will be sent to the DAC for conversion to analog.

5.3 Results obtained on Oscilloscope

The WaveGen module is designed to generate different types of waveforms shown in figure(5.3-5.9) based on user input by utilizing lookup tables (LUTs). It takes in several inputs: clk_50MHz, which is the main clock signal operating at 50 MHz, a slower clock signal clk_out used to control the timing of waveform output, and two control signals: state_out and signal_out. The signal_out input is a 4-bit signal that specifies the type of waveform to be generated, such as sine, cosine, triangular, sinc, sawtooth, or square waves. Based on this selection, different values are loaded into the wave_lut (a lookup table with 32 entries), which contains the pre-calculated waveform values. The module contains a process that initializes the waveform LUTs for each waveform type when signal_out is selected. For instance, if signal_out is 1, the sine wave values are stored in the LUT, and for other values, corresponding waveform values are stored. The module also has a 5-bit address addr that is incremented with each clock cycle to traverse through the LUT and output the corresponding waveform values in wave_out. The waveform output occurs only when state_out equals 5, ensuring the output is controlled. This structure allows the FPGA to generate precise waveforms that can be fed to a DAC for analog signal generation.

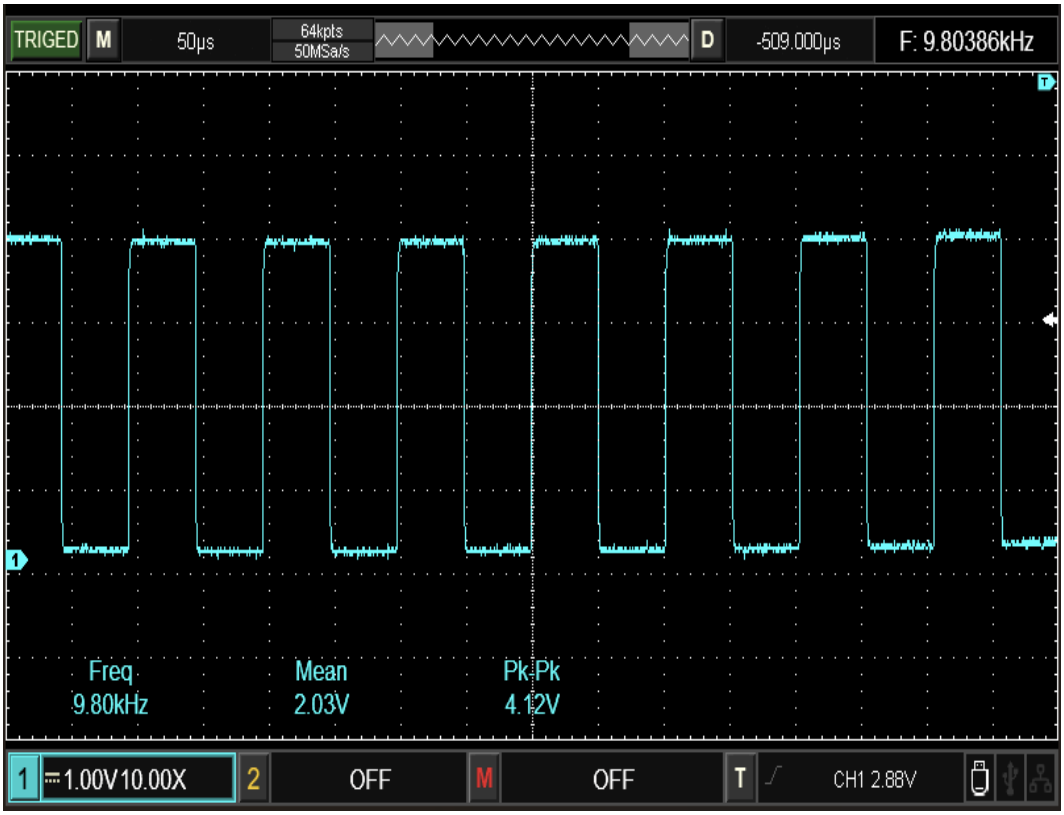

Figure 5.3: Square wave of 10 KHz

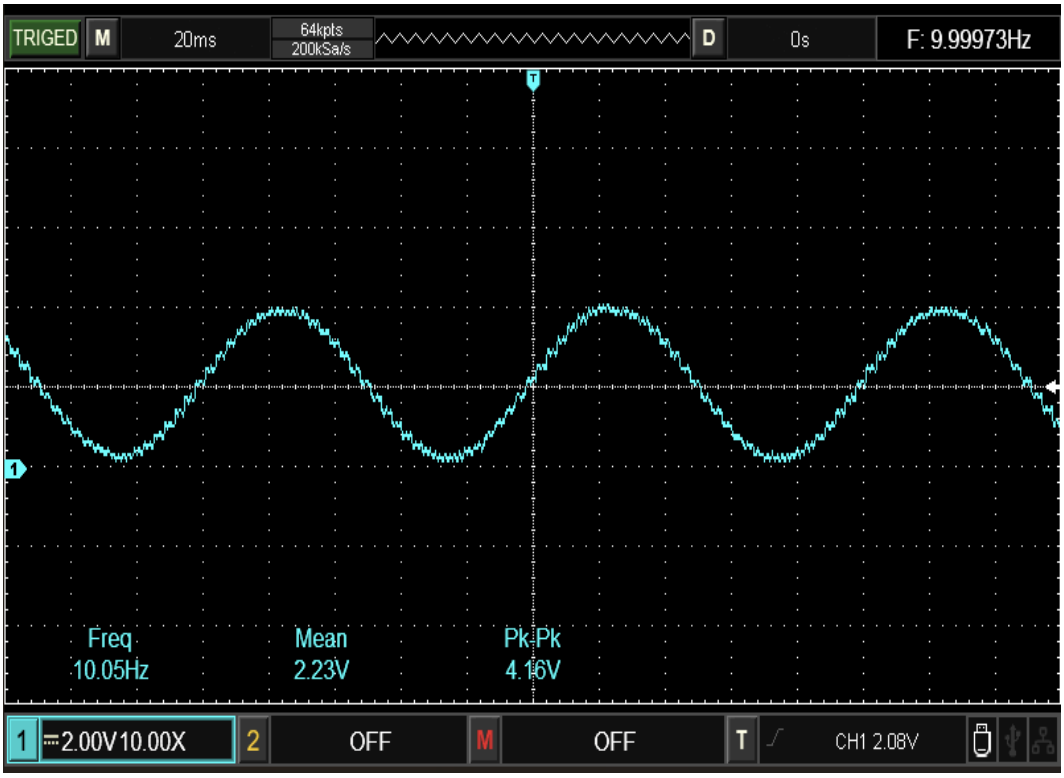

Figure 5.4: Sine wave of 10 Hz

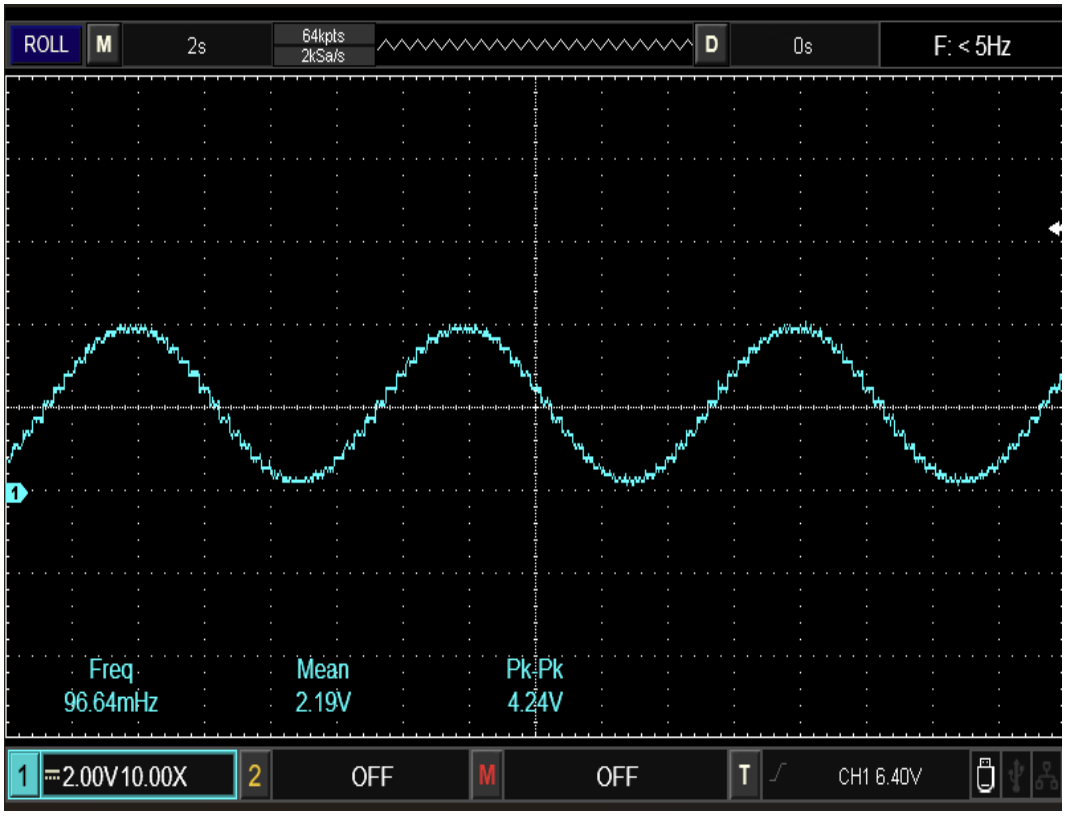

Figure 5.5: Sine wave of 96 mHz

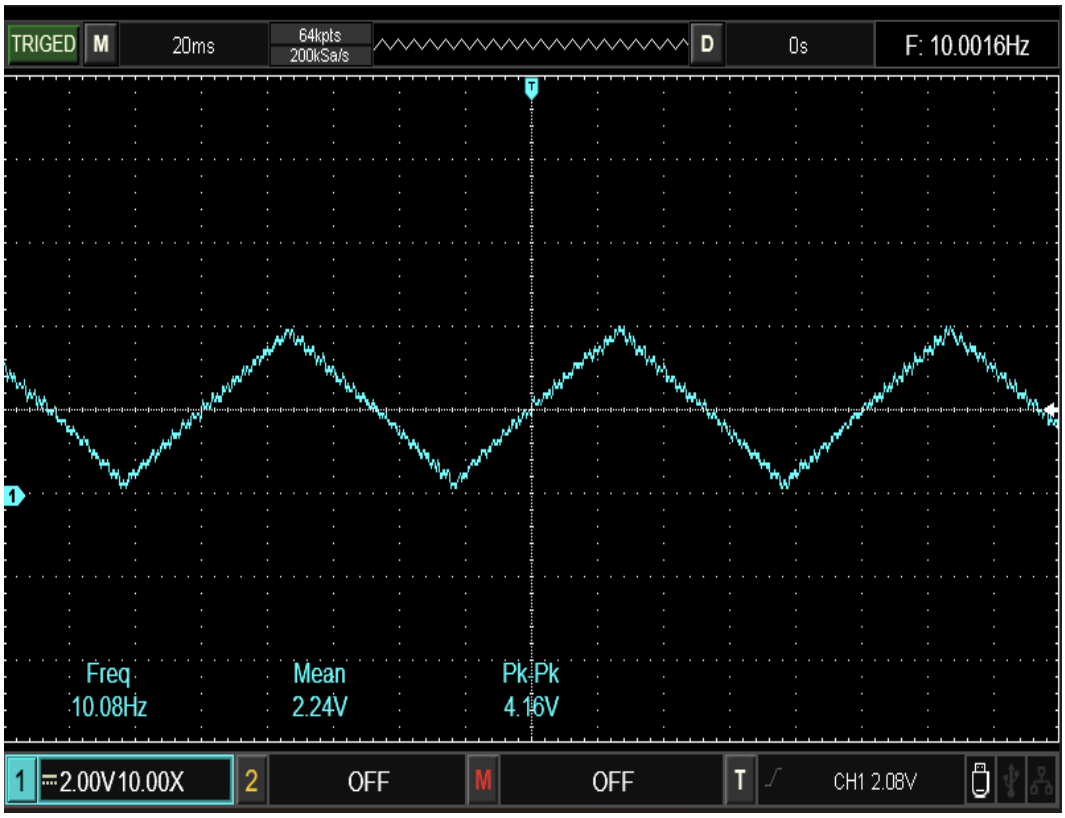

Figure 5.6: Triangular wave of 10 Hz

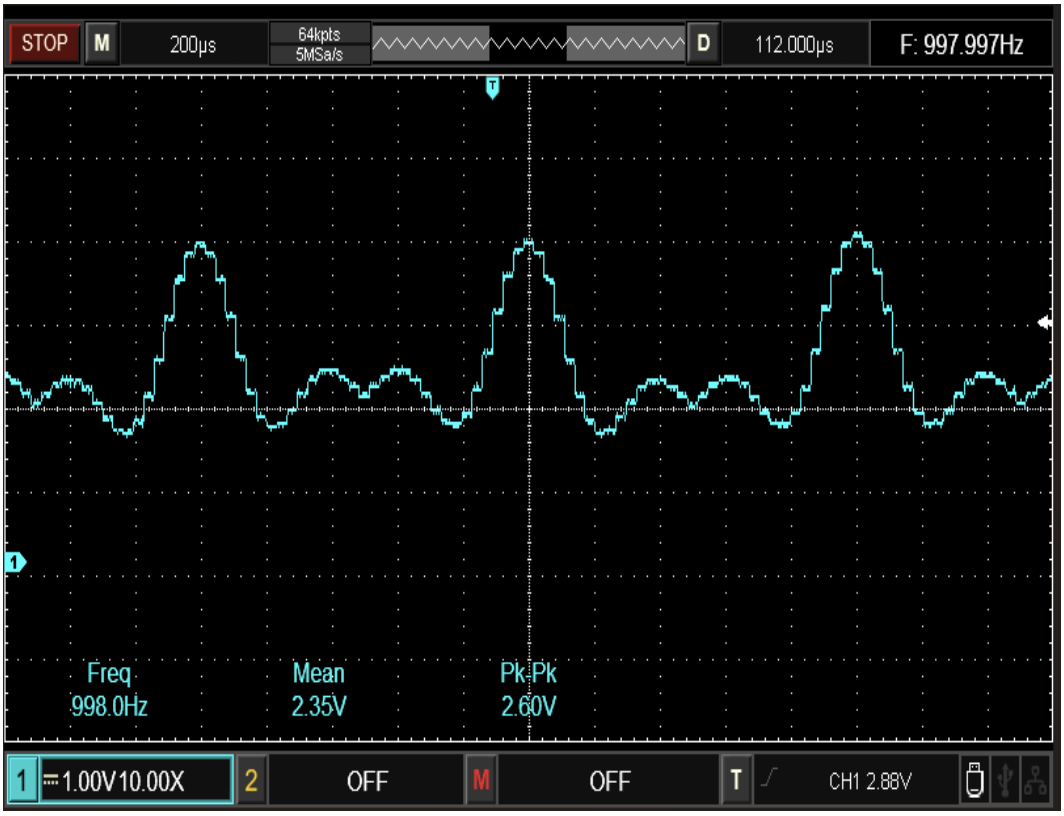

Figure 5.7: Sinc wave of 1 KHz

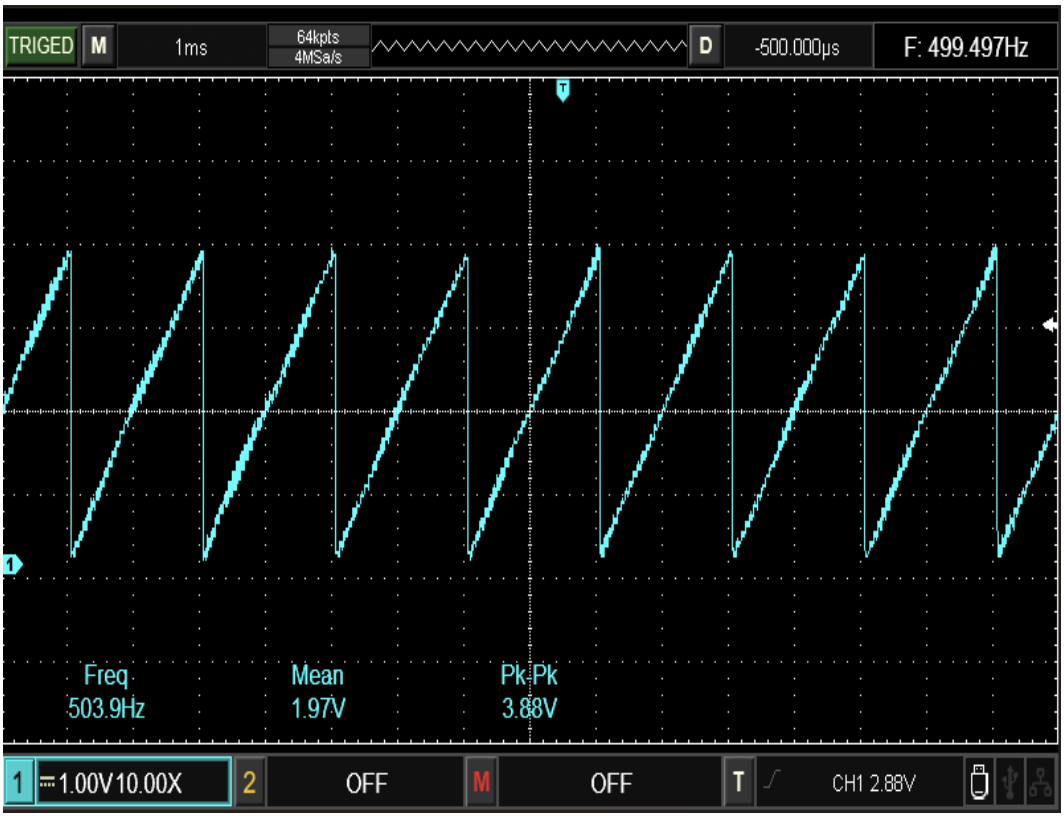

Figure 5.8: Sawtooth wave of 500 Hz

5.4 Conclusion

The FPGA-based waveform generator was successfully designed, implemented, and tested. The system, built on an MAX-10 FPGA, utilized a 4x4 matrix keypad for input, a 16x2 LCD for user guidance, and the AD9708 DAC for digital-to-analog conversion. Various waveforms, including sine, cosine, triangular, sinc, sawtooth, and square waves, were generated using precomputed lookup tables varying from range 1 mHz to 10 MHz. The waveforms were quantized into 8-bit values, stored as hexadecimal codes, and validated through ModelSim simulation. The FPGA accurately reproduced the intended waveforms with minimal distortion, confirming the design's reliability and precision. The system demonstrated efficient frequency control, rapid waveform switching, and real-time user interaction through the keypad and LCD. Furthermore, the signal generated was amplified using the AD8066, ensuring smooth, high-quality analog output suitable for use in signal processing and communication applications.

Bibliography

[1] S.-C. Ding, A. An, and X. Gou, "Digital waveform generator basedon fpga," *Research Journal of Applied Sciences, Engineering and Technology*, vol. 4, no. 14, pp. 2160–2166, 2012.

[2] R. Yue, Y. Wen-Ji, and W. Jinming, "An fpga based multi-functional signal generator using sopc design methodology," in *2016 3rd International Conference on Information Science and Control Engineering (ICISCE)*. IEEE, 2016, pp. 1257–1261.

[3] H. S. Nie, H. S. Liu, Y. M. Zhang, Q. Y. Xie, Q. Liu, J. T. Diao, and H. Xu, "Realization of multi-channel, high-speed waveform generator based on fpga," *Applied Mechanics and Materials*, vol. 182, pp. 506–510, 2012.

[4] Z. Liu, "Design of typical waveform generator based on dds/sopc," in *Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC)*. IEEE, 2013, pp. 642–645.

[5] Q. Chen, "Design and simulation of signal generator based on fpga," *Applied Mechanics and Materials*, vol. 608, pp. 949–953, 2014.

[6] A. Roy, L. Sharma, I. Chakraborty, S. Panja, V. Ojha, and S. De, "An fpga based all-in-one function generator, lock-in amplifier and auto-relockable pid system," *Journal of Instrumentation*, vol. 14, no. 05, p. P05012, 2019.

[7] Y. Liu, K. Sun, Y. Hang, S. Zhu, and X. Cheng, "A design and implementation for multi-function signal generator based on fpga," in *2015 3rd International Conference on Machinery, Materials and Information Technology Applications*. Atlantis Press, 2015, pp. 1253–1256.

[8] S. Lukin, O. Zemlyaniy, and K. Lukin, "Fpga based random waveform generators for noise radars," in *2015 16th International Radar Symposium (IRS)*. IEEE, 2015, pp. 777–782.

Appendix A
FPGA Board

A.1 Board Components

Table A.1: Board Components

	Board Reference	Type	Description
Featured Device	U2	FPGA	10M08SAE144C8G, EQFP, 144 pins, 22 mm x 22 mm.
Configuration, Status, and Setup Elements	J6	Jumper for analog input channel 8	Default connection is to GND. Change jumper to pins 1 and 2 to switch analog source to Arduino header.
	J7	Jumper for analog input channel 7	Default connection is to potentiometer (customer option to purchase and install). Change jumper to pins 1 and 2 to switch analog source to Arduino header.
	SW3	User-defined DIP switch	6-position switch. SW3.1 through SW3.5 are user-defined. SW3.6 is predefined for dual-image configuration.
	D1, D2, D3, D4, D5	LED, Red	These LEDs cycle off and on when the kit is powered on.
	D6	Power LED, green	Illuminates when USB power is present.
	SW2	FPGA reconfiguration push-button	Toggling this button causes the FPGA to reconfigure from on-die Configuration Flash Memory (CFM).
Clock Circuitry	X1	50 MHz oscillator	50 MHz crystal oscillator for general purpose logic.

General User Input and Output	D1, D2, D3, D4, D5	User-defined LEDs, red	User-defined LEDs
	SW1	FPGA register RESET-button	Toggling this button resets all registers in the FPGA.
	R94	Potentiometer	You must purchase and install this device to provide analog inputs signals to the Intel MAX 10 ADC I/P block (analog input channel 8).
Connectors	J2, J3, J4, J5	Arduino UNO R3 connectors	You can mount Arduino UNO R3 compatible Shields (i.e. daughter cards) to connect to the Arduino headers installed on the board.
	J10	JTAG header	Connects an Altera Intel FPGA Download Cable to program or configure the FPGA.
	-	Prototype Area	This through-hole area is not connected to the FPGA. You can use this area to connect or solder additional components.
Power Supply	J1	USB connector	Connects a USB cable to a power source.

A.2 Arduino Connectors

With reference to the layout, arduino connectors J3, J4, and J5 connect to the Intel MAX 10 FPGA. Any analog inputs signals sourced through the Arduino header J4 are first filtered by the evaluation boards op-amp based circuit. This circuit scales the maximum allowable voltage per the Arduino specification (5.0 V) to the maximum allowable voltage per the Intel MAX 10 FPGA ADC IP block (2.5 V). The Arduino pin assignments are given in Table A.2.

Table A.2: Arduino Connectors

Board Reference	Schematic Signal Name	FPGA pin Number	Description
J3.1	ANALOG VREF	5	Arduino analog Vref input

J3.2	GND	-	Arduino GND input
J3.3	ARDUINO_IO13	70	Arduino digital I/O input to FPGA
J3.4	ARDUINO_IO12	69	Arduino digital I/O input to FPGA
J3.5	ARDUINO_IO11	66	Arduino digital I/O input to FPGA
J3.6	ARDUINO_IO10	65	Arduino digital I/O input to FPGA
J3.7	ARDUINO_IO09	64	Arduino digital I/O input to FPGA
J3.8	ARDUINO_IO08	62	Arduino digital I/O input to FPGA
J4.1	ARDUINO_A0	6	Arduino analog channel input through the op-amp filter circuit to the FPGA ADC IP input channel ADCIN1
J4.2	ARDUINO_A1	7	Arduino analog channel input through the op-amp filter circuit to the FPGA ADC IP input channel ADCIN2
J4.3	ARDUINO_A2	8	Arduino analog channel input through the op-amp filter circuit to the FPGA ADC IP input channel ADCIN3
J4.4	ARDUINO_A3	10	Arduino analog channel input through the op-amp filter circuit to the FPGA ADC IP input channel ADCIN4
J4.5	ARDUINO_A4	11	Arduino analog channel input through the op-amp filter circuit to the FPGA ADC IP input channel ADCIN5
J4.6	ARDUINO_A5	12	rduino analog channel input through the op-amp filter circuit to the FPGA ADC IP input channel ADCIN6

J4.7	ARDUINO_A6	13	Arduino analog channel input through the op-amp filter circuit to the FPGA ADC IP input channel ADCIN7
J4.8	ARDUINO_A7	14	Arduino analog channel input through the op-amp filter circuit to the FPGA ADC IP input channel ADCIN8
J5.1	ARDUINO_A7	86	Arduino digital I/O input to FPGA
J5.2	ARDUINO_A6	84	Arduino digital I/O input to FPGA
J5.3	ARDUINO_A5	81	Arduino digital I/O input to FPGA
J5.4	ARDUINO_A4	79	Arduino digital I/O input to FPGA
J5.5	ARDUINO_A3	77	Arduino digital I/O input to FPGA
J5.6	ARDUINO_A2	76	Arduino digital I/O input to FPGA
J5.7	ARDUINO_A1	75	Arduino digital I/O input to FPGA
J5.8	ARDUINO_IO0	74	Arduino digital I/O input to FPGA

A.3 User-Defined DIP Switch

Board reference SW3 is a 6-pin DIP switch. Switches 1 through 5 are user-defined, and provide additional FPGA input control. When the switch is in the OPEN or OFF position, a logic 1 is selected. When the switch is in the CLOSED or ON position, a logic 0 is selected. There is no board-specific function for these switches. The following Table 3 lists the user-defined DIP switch schematic signal names and their corresponding FPGA pin numbers.

Table A.3: User-Defined DIP Switch

Board Reference SW3	**Schematic Signal Name**	**I/O Standard (V)**	**FPGA Device Pin Number**
1	Switch1	3.3	120
2	Switch2	3.3	124
3	Switch3	3.3	127
4	Switch4	3.3	130
5	Switch5	3.3	131

A.4 User-Defined LEDs

The development board includes five user-defined LEDs. Board references D1 through D5 are user LEDs that allow status and debugging signals to be driven to the LEDs from the designs loaded into the Intel MAX 10 FPGA device. The LEDs illuminate when a logic 0 is driven, and turn off when a logic 1 is driven. There is no board- specific function for these LEDs. The following Table 4 lists the user-defined LED schematic signal names and their corresponding FPGA pin numbers.

Table A.4: User-Defined DIP Switch

Board Reference SW3	**Schematic Signal Name**	**I/O Standard (V)**	**FPGA Device Pin Number**
D1	LED1	2.0	132
D2	LED2	2.0	134
D3	LED3	2.0	135
D4	LED4	2.0	140
D5	LED5	2.0	141

A.5 Power Management

To measure the actual power of the FPGA, there are test pads on the board to be used as probe points for multi-meter probes. You can measure the current and using the equation Power = Resistance x Current Squared, calculate the power dissipation. Test pads TP2 and TP3 measure the current consumed by the FPGA core. Test pads TP4 and TP5 measure the current consumed by all of the FPGA's I/O banks. All the other test pads are used to verify the voltage levels of various nodes on the board. Table 2.3 gives a brief idea of power management.

Table A.5: Power Management

Test Pads	Measuring	Description	Expected Value
TP1	Board input voltage	Verify the USB input voltage	5.0 V
TP2-TP3	FPGA core current	Power calculation for FPGA Vcc_core power consumption. Resistor R1 = 0.1. Current measured by user's multi-meter.	3.3V

TP4-TP5	FPGA I/O current	Power calculation for FPGA Vcc_io power consumption. Resistor R4 = 0.12. Current measured by user's multi-meter.	3.3V
TP6	Analog voltage	Verify the proper voltage required by the FPGA Vcca inputs.	3.3 V
TP7	Analog GND	Verify the proper voltage required by the FPGA ADC IP block GND inputs.	0 V
TP8	Digital GND	Verify the proper voltage required by the FPGA digital GND inputs.	0 V
TP9	Digital GND	Verify the proper voltage required by the FPGA digital GND inputs.	0 V