IJCRT.ORG ISSN : 2320-2882
INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

Fpga-Based Signal Generation For Pcb Testing Simulator"

SHREYAS TR, Dr. JEERU DINESH REDDY

MTech Student, Assistant Professor Department of Electronics & Communication Engineering

BMSCE, Bengaluru, India

Abstract: The increasing complexity of modern electronic systems necessitates efficient and reliable testing mechanisms during printed circuit board (PCB) development and manufacturing. This project, titled *"FPGA-Based Signal Generation for PCB Testing Simulators,"* aims to design and implement a high-speed, versatile signal generator using FPGA technology. The system can generate various waveforms, including sine, square, triangular, and pulse signals, with precise control over parameters such as frequency, amplitude, and duty cycle.
The FPGA-based implementation employs techniques such as lookup tables (LUTs) for waveform synthesis and programmable logic for customizable signal parameters. The generated signals are output through a digital-to-analog converter (DAC) and verified using visualization tools like oscilloscopes. This project emphasizes simplicity and flexibility, reducing complexity compared to existing automated test systems while ensuring high accuracy and real-time signal generation capabilities.
Applications include PCB testing, educational tools for understanding signal processing and FPGA programming, and pre-processing in communication systems. The results demonstrate the feasibility of a cost-effective, configurable signal generator that serves as a foundation for future enhancements, such as feedback analysis and adaptive testing.

Index Terms— FPGA, Signal Generation, PCB Testing, UART, VHDL, DAC, Logic Analyzer, Waveform Synthesis, Logic Locking, LFSR.

I. Introduction

In the era of advanced integrated circuit (IC) and printed circuit board (PCB) technology, ensuring the security and reliability of electronic systems has become paramount. While testing mechanisms like scan chains are indispensable for verifying functionality and diagnosing faults during the development and manufacturing of ICs and PCBs, they also present significant vulnerabilities. Specifically, scan-based side-channel attacks exploit these mechanisms to extract sensitive information, posing a severe threat to intellectual property and system integrity. PCBs are the foundational platforms for modern electronics, housing ICs and providing interconnectivity for various components. Ensuring their functionality and security is critical, as any fault or vulnerability can compromise the entire system. This project addresses these challenges by proposing a novel secure scan design based on a logic locking scheme that integrates seamlessly into PCB testing workflows.

The approach involves incorporating parallel latches and a dynamic key mechanism, which includes a clock signal to generate a unique key for each design. This key enhances the security of the scan chain, ensuring that unauthorized access is effectively prevented. Additionally, the use of parallel latches introduces minimal hardware overhead, making it suitable for space-constrained PCB designs. To further strengthen protection, a linear feedback shift register (LFSR) is employed for dynamic obfuscation of scan outputs. This technique ensures unpredictability, making the system resistant to attack methods reliant on predictable scan patterns. The initial activation response of the parallel latches is securely stored in Read-Only Memory (ROM) as the golden key, forming the foundation of the secure design.

The proposed solution ensures that the scan chain remains functional for PCB testing purposes while significantly enhancing its security. By enabling reliable signal generation and secure test configurations, this project contributes to the development of secure and dependable PCBs. The dual focus on usability and protection makes the design a robust and efficient countermeasure against all known scan-based side-channel attacks, supporting advancements in the testing and security of ICs and PCBs alike.

II. LITERATURE SURVEY

Several studies have focused on enhancing PCB inspection techniques through automation and intelligent algorithms. Mukesh Kumar and colleagues [1] proposed an image enhancement algorithm designed to sharpen the edges of PCB tracks and effectively filter out noise using National Instrument Vision Assistant software, achieving an impressive inspection rate of approximately 99.43 parts per second. T. J. Mateo Sanguino et al. [2] developed a visual inspection system utilizing subtraction methods and statistical analysis to automatically detect and classify twelve types of defects, significantly improving the accuracy of error classification by optimizing light intensity conditions. Ziyin Li and colleagues [3] introduced an Automated Optical Inspection (AOI) system that combines image enhancement, denoising, and segmentation algorithms to identify a range of PCB defects, including wire gaps, voids, scratches, short circuits, and open circuits, achieving a remarkable detection success rate exceeding 95% with a resolution of 15 µm. In another study, Manasa H R and Anitha D B [4] implemented an automated inspection system based on template matching and a referential approach, capable of detecting missing components and verifying physical dimensions on specific PCBs, although the method requires customized code for each board type. Anitha D B and Mahesh Rao [5] proposed a multifaceted defect detection approach employing contour analysis, optical character recognition (OCR), and pixel subtraction techniques on the LabVIEW platform, enabling quick identification of misplaced or incorrect SMT components while reducing inspection time. Further advancing the field, Hendawan Soebhakti and Farkhad Ihsan Hariadi [8] applied neural network-based automated optical inspection methods to detect missing components on surface-mount PCBs by extracting histogram values from captured images, achieving an inspection accuracy rate of approximately 93%. Lastly, Tejas Khare, Vaibhav Bahel, and Anuradha C. Phadke [9] introduced the "PCB Fire" system, which leverages object detection, pixel manipulation, and the YOLO (You Only Look Once) convolutional neural network algorithm to classify and detect missing PCB components, achieving an accuracy rate of 75.48% while significantly reducing time and capital loss during fault identification. These studies collectively demonstrate the ongoing advancements in automated PCB inspection systems, emphasizing the integration of machine vision, AI techniques, and specialized algorithms to improve defect detection efficiency and reliability.

III. RESEARCH METHODOLOGY

This project utilizes CPLD/FPGA-based signal generation to produce configurable waveforms such as sine, square, and triangular signals. The design is implemented using Verilog/VHDL, with lookup tables (LUTs), The generated signals are output via a DAC, visualized on an oscilloscope, and integrated into a PCB testing simulator for validation.

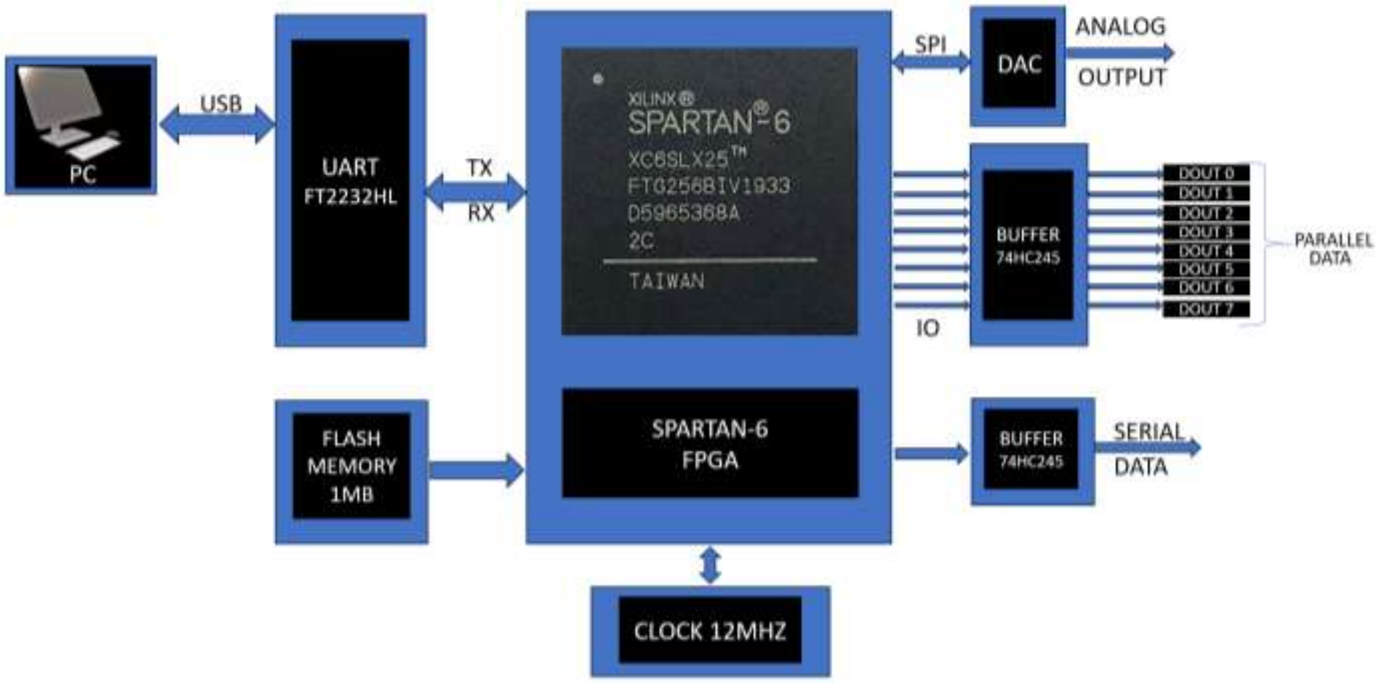


Figure1. Block Diagram of signal generator simulator

DIGITAL SIGNAL GENERATION

FPG-based signal generation has emerged as a powerful alternative to traditional microcontroller-based systems due to its superior speed, flexibility, and inherent parallelism. In this project, waveform generation plays a critical role in PCB testing, where signals such as sine, square, triangular, and pulse waves are used to validate circuit functionality and diagnose faults. The goal is to develop a versatile signal generation system that can replicate various signal types with high accuracy, adaptability, and real-time control, leveraging the reconfigurability of FPGA hardware.

The system supports multiple waveform types using digital design strategies tailored for FPGA platforms. For sine wave generation, lookup tables (LUTs) are employed to store precomputed sine values. These values are accessed sequentially based on clock cycles, enabling smooth and continuous waveform output. Square waves are produced using counters that alternate between high and low states. Their frequency and duty cycle can be precisely controlled by adjusting the thresholds within the counting logic. Triangular waveforms are generated through linear increment and decrement operations, creating consistently rising and falling edges. These waveforms are also regulated using counters to manage both amplitude and frequency. Pulse signals are generated using pulse width modulation (PWM), which allows dynamic control over pulse width and duty cycle. Additionally, the system is designed to support custom waveform profiles by directly programming specific values into the LUT, offering maximum flexibility for unique testing scenarios or specialized applications.The system uses a Universal Asynchronous Receiver Transmitter (UART) protocol for serial communication between devices. This protocol operates using just two signal lines: one for transmitting (TX) and the other for receiving (RX). Data is transmitted asynchronously in a frame-based format, which allows two devices to communicate without a shared clock signal. Each
UART frame consists of a start bit, followed by data bits, an optional parity bit, and a stop bit. The line remains high during idle states. A start bit—represented by a low signal—indicates the beginning of a transmission. Once the data has been sent, a stop bit—represented by a high signal—marks the end of the

frame. This structure enables consistent and efficient communication, even over simple hardware setups.hardware setups.

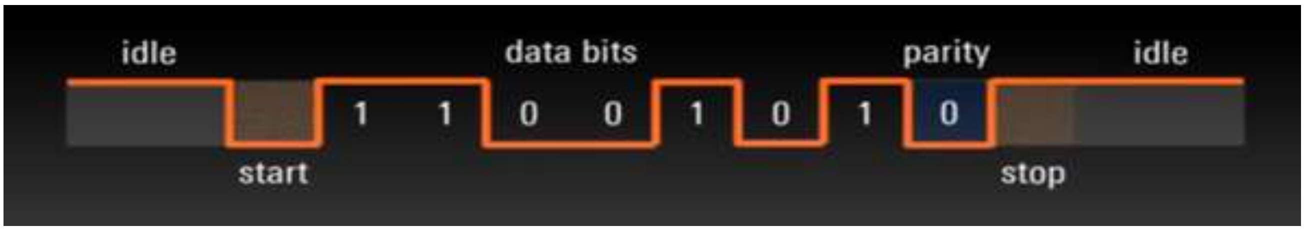
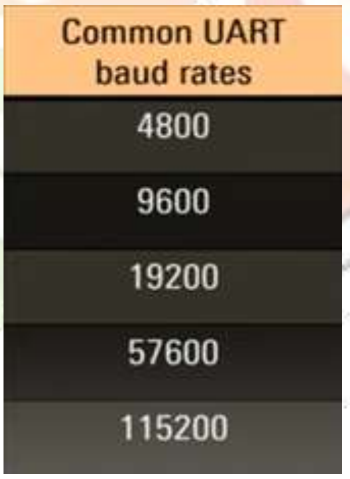


Figure-2: UART

Since UART operates asynchronously, the transmitter and receiver do not share a common clock signal. Instead, both devices must be configured to transmit and receive data at the same predefined speed to ensure proper communication. This characteristic eliminates the need for clock synchronization lines, simplifying hardware design while maintaining reliable data exchange. In the proposed FPGA-based signal measurement system, the UART interface has been configured to operate at a baud rate of 9600 bits per second, balancing ease of implementation with sufficient data throughput for real-time signal control and monitoring. The transmitter and receiver in the proposed FPGA-based signal generation system are modeled using finite state machines to ensure accurate UART communication. The transmitter begins in an idle state, waiting for the transmission mode to be enabled. Once activated, it progresses through a series of states including initialization, synchronization, message transmission, and conclusion. The flow involves starting the transmission, sending synchronization signals, transmitting the actual message, and finally either resetting the system or sending the message in hexadecimal format depending on the condition. On the receiver side, the process starts by waiting for a start bit to signal the beginning of data transmission.



Common UART baud rates
4800
9600
19200
57600
115200

,Figure-3:Baud rate

to obtain data and subsequently receives a complete byte, in a bit-by-bit manner. After successfully receiving all eight bits, it checks for the presence of a valid stop bit to confirm the integrity of the communication before resetting for the next incoming signal. Together, these state machines ensure reliable and synchronized serial data exchange within the FPGA system.

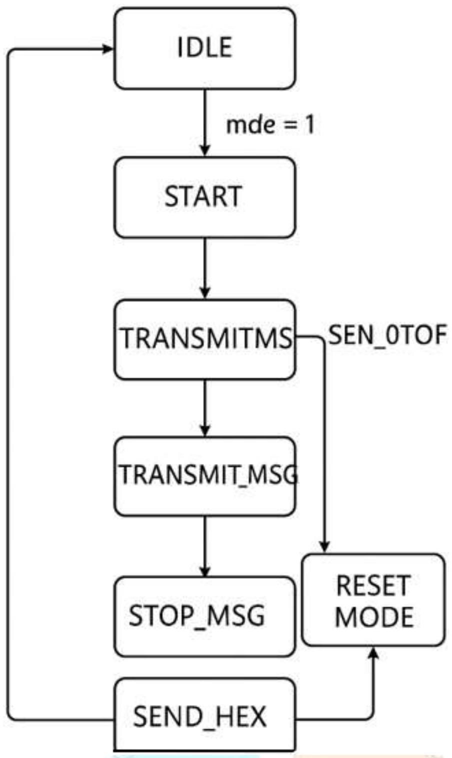


Figure4.State Diagram Of TRANSMITTER

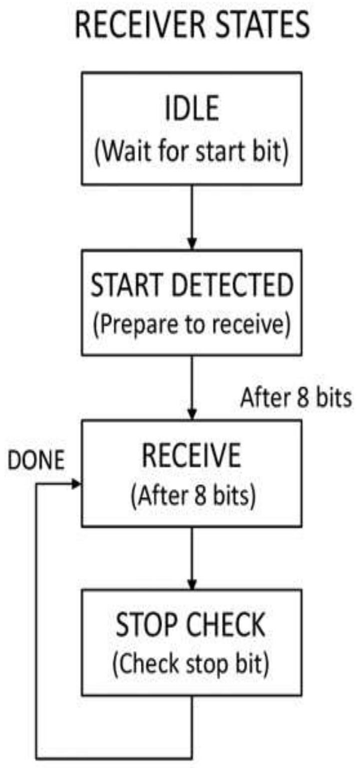
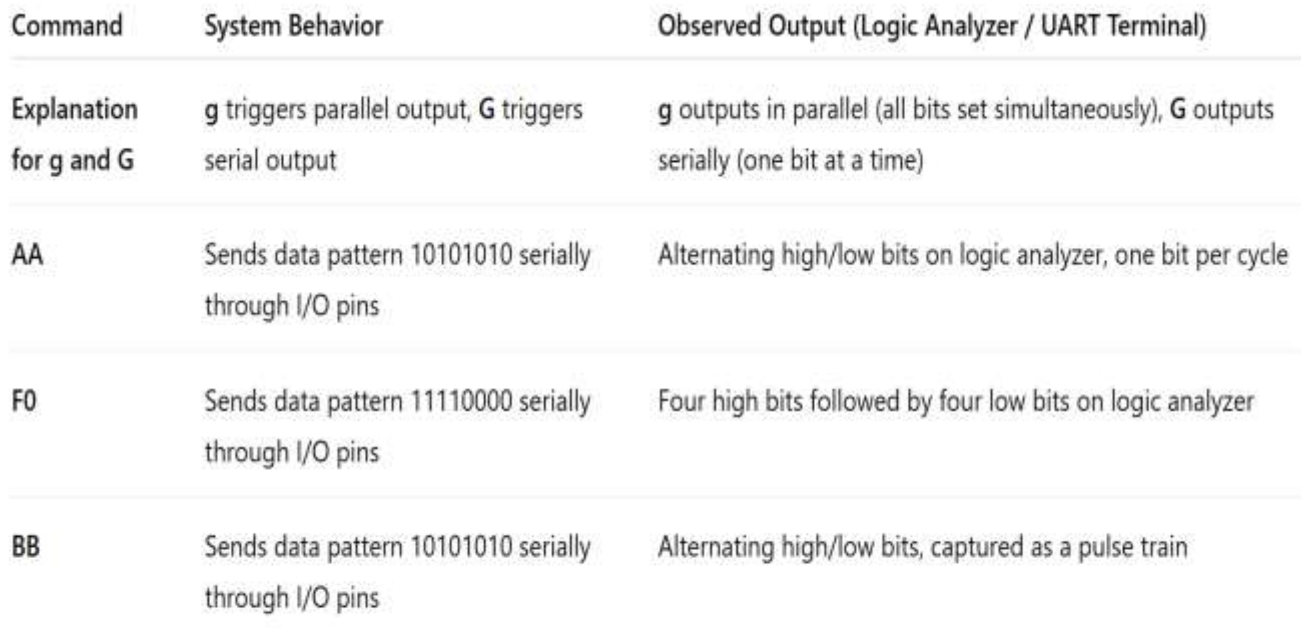


Figure5. State Diagram Of RECEIVER

The final digital implementation integrates UART-based user interaction for configuring signal outputs via FPGA. Upon receiving two-character commands through a serial terminal, the system interprets and responds accordingly. Commands beginning with 'G' trigger predefined 8-bit parallel patterns transmitted through the FPGA's general-purpose I/O pins. These outputs are then captured and visualized using a logic analyzer, allowing detailed observation of digital transitions. Similarly, commands starting with 'g' initiate serial output through a shift register, transmitting one bit at a time via a designated I/O pin. The resulting serial data stream is also monitored using the logic analyzer software, enabling verification of bit sequences and signal timing. Additionally, when the character 'I' is received, the system enters a communication mode and transmits structured messages such as “Signal Generator” followed by “MAINPROJECT 2024” through the UART transmitter. This configuration allows the user to dynamically control signal patterns and analyze the output in real time, providing a flexible and insightful approach to FPGA-based signal generation and validation.



Command	System Behavior	Observed Output (Logic Analyzer / UART Terminal)
Explanation for g and G	**g** triggers parallel output, **G** triggers serial output	**g** outputs in parallel (all bits set simultaneously), **G** outputs serially (one bit at a time)
AA	Sends data pattern 10101010 serially through I/O pins	Alternating high/low bits on logic analyzer, one bit per cycle
F0	Sends data pattern 11110000 serially through I/O pins	Four high bits followed by four low bits on logic analyzer
BB	Sends data pattern 10101010 serially through I/O pins	Alternating high/low bits, captured as a pulse train

Figure6. Working of Command Behavior and Output Observation on Logic Analyzer

Following the table, the logic analyzer outputs corresponding to the different commands are shown in the images below. These outputs visually confirm the system's ability to generate specific bit patterns as described in the previous section. The logic analyzer captures the transitions of high and low bits for each command, providing valuable insights into the timing and behavior of the signals generated by the FPGA.

ANALOG SIGNAL GENERATION

The analog section of signal generator uses the FPGA to communicate with a DAC (Digital-to-Analog Converter) using the SPI protocol. The DAC converts the 8-bit digital data sent from the FPGA into a corresponding analog voltage, which can then be measured on an oscilloscope or multimeter. The primary motivation behind this project is to provide a flexible and programmable analog signal generator using an FPGA and a DAC module (DAC121S101). The fpga receives user commands over a serial interface, processes them, and sends corresponding signals to the DAC via SPI communication. This allows for real-time voltage output control based on user input.

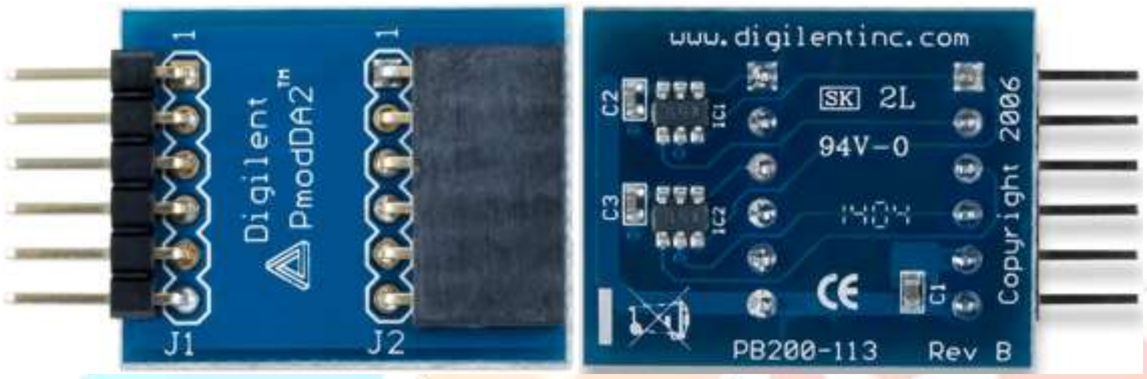


Figure7. Digilent PmodDA2 DAC front and back view

To communicate with the DAC121S101 digital-to-analog converter (DAC), the SPI protocol is implemented. SPI (Serial Peripheral Interface) is a synchronous serial communication protocol widely used in embedded systems. It involves four primary signals:

- **SCLK** (Serial Clock): Controls timing of data transmission.
- **MOSI** (Master Out Slave In): Used to send data from the master (FPGA) to the slave (DAC).
- **MISO** (Master In Slave Out): Not used here since the DAC only receives data.
- **nCS / nSYNC**: Active-low chip select line to enable communication.

For the DAC121S101, **16-bit data words** are sent via SPI to update the analog output. The format includes 12 bits of actual data and 4 control bits. The DAC samples the incoming bits on the **rising edge of the clock** and latches the data when nSYNC goes high.The signal nSYNC is kept low during the entire 16-bit transmission, then driven high immediately after to latch the data into the DAC. The process ensures reliable conversion of digital signals into analog voltages.

To generate analog voltages, the digital values are sent from the FPGA to the DAC121S101 using SPI. This involves creating. When the FPGA sends a digital command (e.g., 8-bit binary like 0xAA or 0xB9), the FSM triggers the DAC via SPI. Once the command is received and latched (via rising nSYNC), the DAC outputs a corresponding analog voltage.UART (Universal Asynchronous Receiver/Transmitter) is a common protocol for serial communication. It is used here for user input, allowing voltage levels to be typed in and interpreted by the fpga. The main idea behind my DAC-based working setup is simple yet powerful to translate user-entered digital commands into precise analog voltage signals using an FPGA and a DAC121S101 module. The system operates over UART and SPI, forming a seamless bridge between digital user input and real-world analog output. As soon as the fpga is programmed, it enters a waiting state and expects a user to type a command. If the user types 1, the system enters DAC mode and displays:

"DAC mode activated. Type 3-digit voltage (000–330):"

Once in DAC mode, if a user enters a 3-digit number like 100, the fpga interprets it as 100 millivolts or 1.0 volts. This digital input is processed using a simple voltage-scaling formula and then converted to a 12-bit digital value ranging from 0 to 4095. This value is transmitted to the DAC using SPI communication.

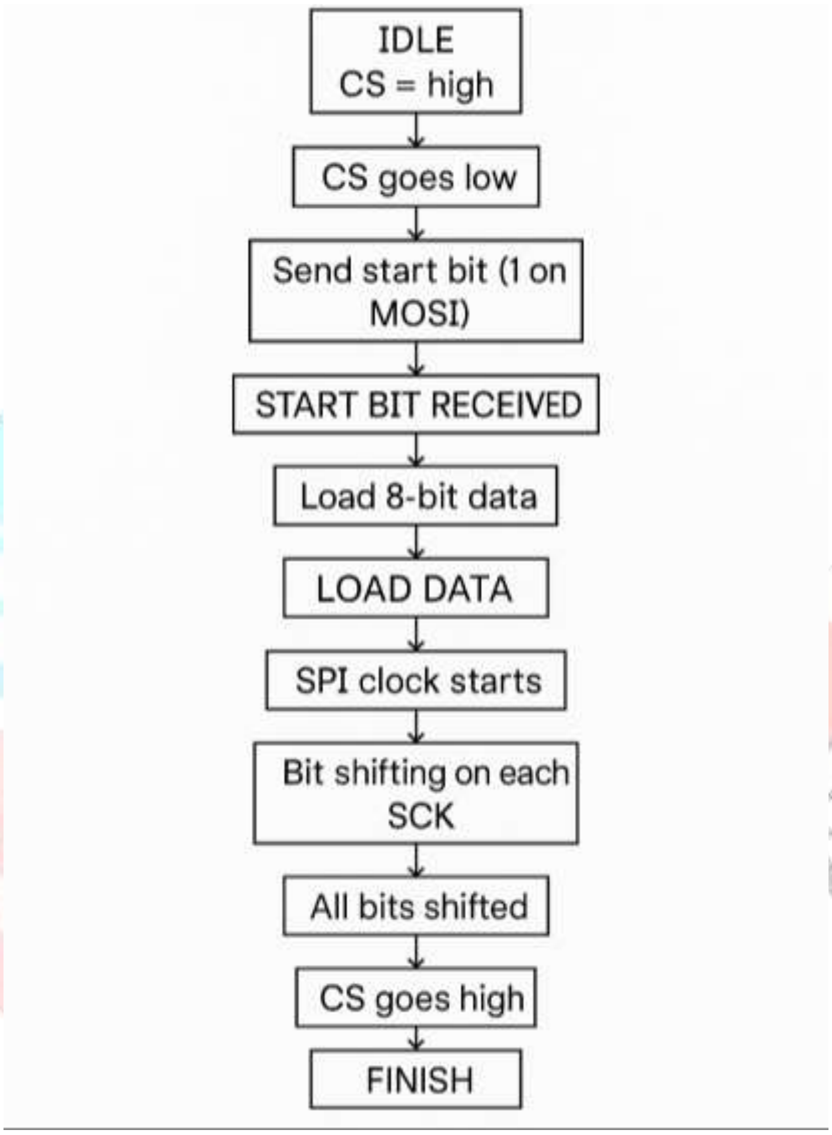


Figure8. A state diagram to manage SPI timing

Basic Formula:

The formula to convert a digital value (D) to its corresponding analog voltage (Vout) is:

Vout = (D / 4095) × Vref

Where:

- D is the digital value (0–4095)
- Vref is the reference voltage (3.3V)

The DAC121S101, a 12-bit DAC, receives this 12-bit data over SPI and converts it into a corresponding analog voltage. Since it operates with a reference voltage (Vref) of 3.3V, the maximum output it can generate is 2.5V, and all voltages between 0 and 3.3V are scaled proportionally based on the input.

To calculate the DAC digital value from a desired input voltage (given in millivolts), we use:

D = (Input_mV × 4095) / 2500

Example: If the user inputs 1000 mV:

D = (100 × 4095) / 2500 = 1638

Now using the first formula to verify:

Vout = (1638 / 4095) × 2.5 = 1.0V

To transmit a 12-bit DAC value over the SPI interface, we must divide it into two 8-bit chunks because SPI transmits data in 8-bit (1-byte) packets. This is done using simple bitwise operations. The **high byte** is obtained by shifting the 12-bit value 8 bits to the right, which essentially extracts the upper 4 bits (and pads with zeros). The **low byte** is obtained by performing a bitwise AND with 0xFF to retain only the lower 8 bits of the 12-bit value.

To send a 12-bit DAC value over SPI, we split it into two 8-bit values:

- High Byte = DAC_Value >> 8
- Low Byte = DAC_Value & 0xFF

Example for DAC_Value = 1638:

- High Byte = 1638 >> 8 = 6 = 0x06
- Low Byte = 1638 & 0xFF = 102 = 0x66

The user can type any value between 0000 and 330, and the DAC will output the exact corresponding voltage.

User Input (mV)	12bit DAC Cod (04095)	Analog Voltage Output (V)
0000	0	0.00
0500	819	0.50
1000	1638	1.00
1500	2457	1.50
2000	3276	2.00
2500	4095	2.50

Figure9. Analog signal generation visualization

The analog output is continuously held stable and can be observed with a multimeter connected to the DAC output pin.

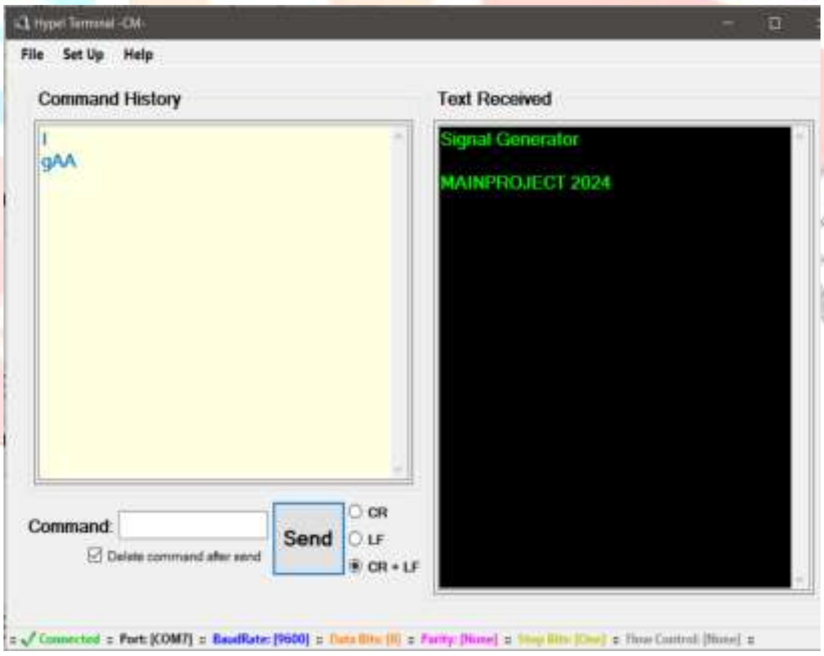
Let’s say a student wants to generate a 1.7V analog signal. They simply open the Serial Terminal, type 170, and press enter. Instantly, the DAC generates 1.7V and outputs it via the analog output pin. This process doesn’t require reprogramming or physical adjustment, everything is software-controlled. This kind of dynamic voltage control is highly beneficial in laboratory experiments, sensor simulations, and waveform generation. The simplicity of the interface makes it accessible to beginners, while the underlying accuracy and resolution offer utility for advanced users.

IV. RESULTS

DIGITAL SIGNAL GENRATION

- 8-Bit Parallel Digital Data :

The user enters command "gAA" via HyperTerminal. Since "gAA" corresponds to a parallel output, this input will prompt the FPGA to output an 8-bit pattern on the I/O pins, which can be observed on the logic analyzer. The command is entered in parallel mode, meaning all bits are set simultaneously. On the other hand, if the user enters command "g" via HyperTerminal, this triggers serial output mode. The FPGA will then transmit the 8-bit pattern parallel, one bit at a time, with each bit's transition captured sequentially by the logic analyzer, confirming the parallel output functionality as expected from the command "g".



```
I
gAA
```

```
Signal Generator

MAINPROJECT 2024
```

Delete command after send [✓]; [✓ CR + LF]

✓ Connected :: Port: [COM7] :: BaudRate: [9600] :: Data Bits: [8] :: Parity: [None] :: Stop Bits: [One] :: Flow Control: [None] ::

Figure10. HyperTerminal displaying user input of command AA

The logic analyzer captures the parallel output of the bit pattern triggered by the "gAA" command. The FPGA outputs the pattern **10101010** across the I/O pins, with all bits set simultaneously. Each high/low state is captured by the logic analyzer, confirming the parallel output behavior. The output corresponds to the behavior expected from the FPGA when the "gAA" parallel output command is received, with all bits transmitted simultaneously.

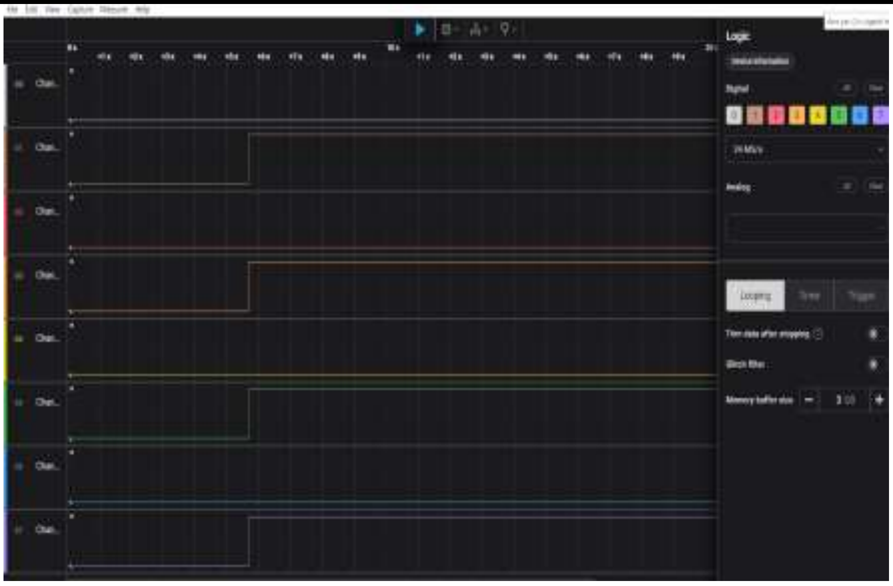
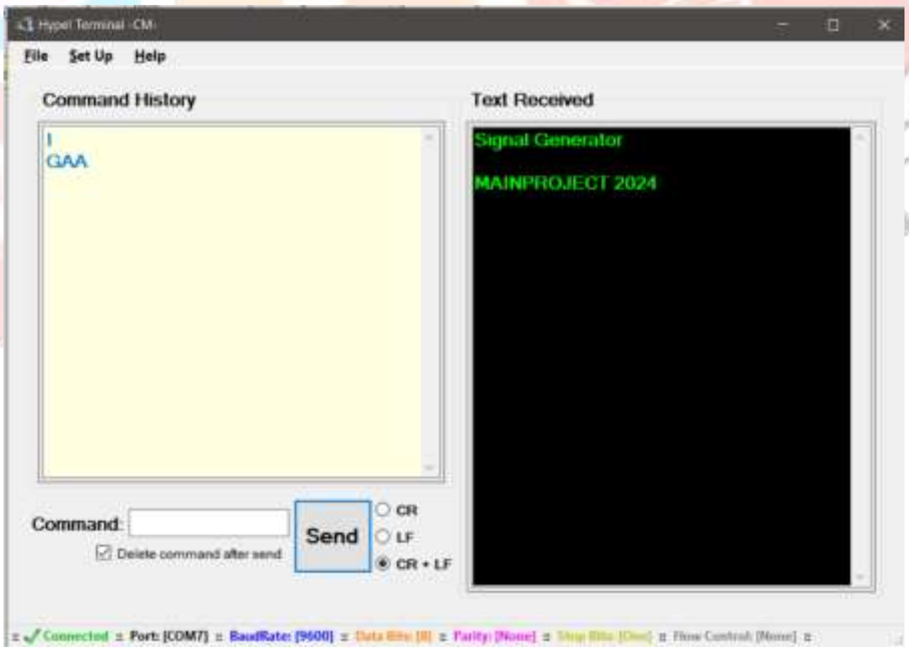


Figure11. Logic analyzer output showing the signal corresponding to command GAA.

- 8-Bit Serial Digital Data :

The user enters command "GAA" via HyperTerminal. Since "GAA" corresponds to a serial output, this input will prompt the FPGA to output an 8-bit pattern on the I/O pins, which can be observed on the logic analyzer. The command is entered in parallel mode, meaning all bits are set simultaneously. On the other hand, if the user enters command "G" via HyperTerminal, this triggers serial output mode. The FPGA will then transmit the 8-bit pattern **serially**, one bit at a time, with each bit's transition captured sequentially by the logic analyzer, confirming the **serial output** functionality as expected from the command "G".



Command History

I
GAA

Text Received

Signal Generator

MAINPROJECT 2024

Figure12. HyperTerminal displaying user input of command AA.

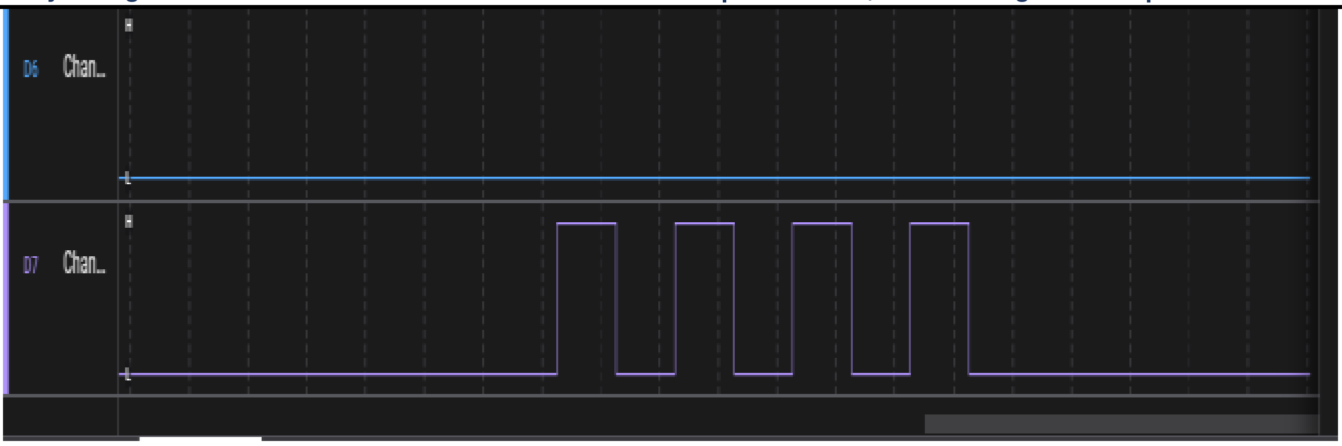


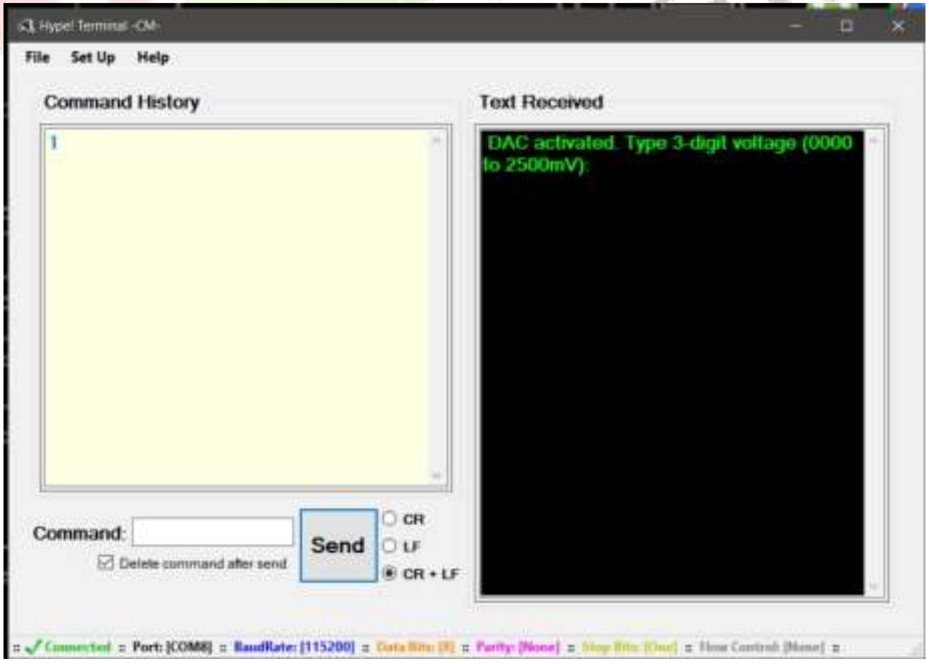
Figure13.Logic analyzer output showing the signal corresponding to command GAA.

The logic analyzer captures the serial transmission of the bit pattern triggered by the "G" command. The pattern is transmitted bit by bit, with each high/low state appearing sequentially. The output corresponds to the behavior expected from the FPGA when the "G" serial output command is received.

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data = 0xAA	1	0	1	0	1	0	1	0
Data generated: Bit 7 to 0								

ANALOG SIGNAL GENERATION

The functionality of the DAC system was validated by entering various 4-digit millivolt values through the serial terminal. The Arduino Nano interpreted these inputs, converted them to a 12-bit digital format using the scaling formula, and transmitted the values to the DAC121S101 module over SPI. The resulting analog voltages were measured using a digital multimeter connected to the DAC output pin.



Command History: 1

Text Received:

```
DAC activated. Type 3-digit voltage (0000
to 2500mV):
```

Command:

[✓] Delete command after send

CR | LF | [✓ CR + LF]

Connected :: Port: [COM8] :: BaudRate: [115200] :: Data Bits: [8] :: Parity: [None] :: Stop Bits: [One] :: Flow Control: [None]

Figure 14. Shows the system response when the user inputs 1. The system enters DAC mode and prompts the user

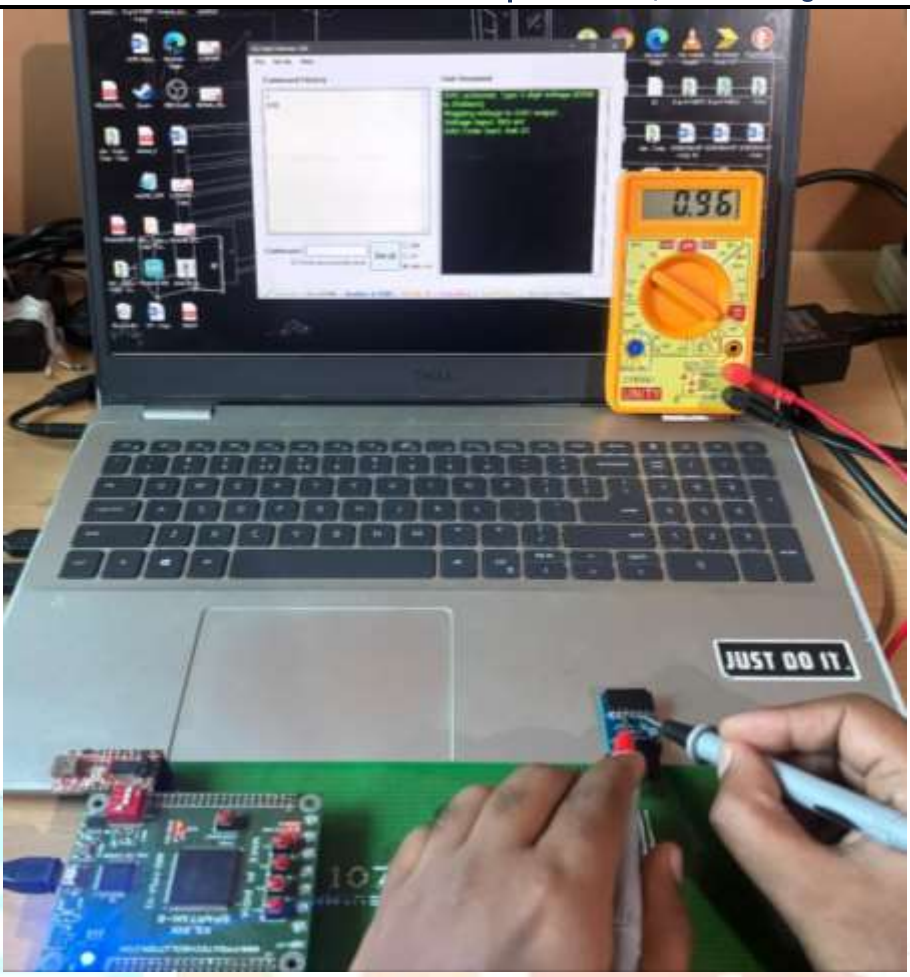


Figure 15. Demonstrates the input of 100 (representing 100 mV). The multimeter reading for this input was approximately 0.96 V, closely matching the expected theoretical output (minor deviation attributed to loading effects or absence of a buffer).

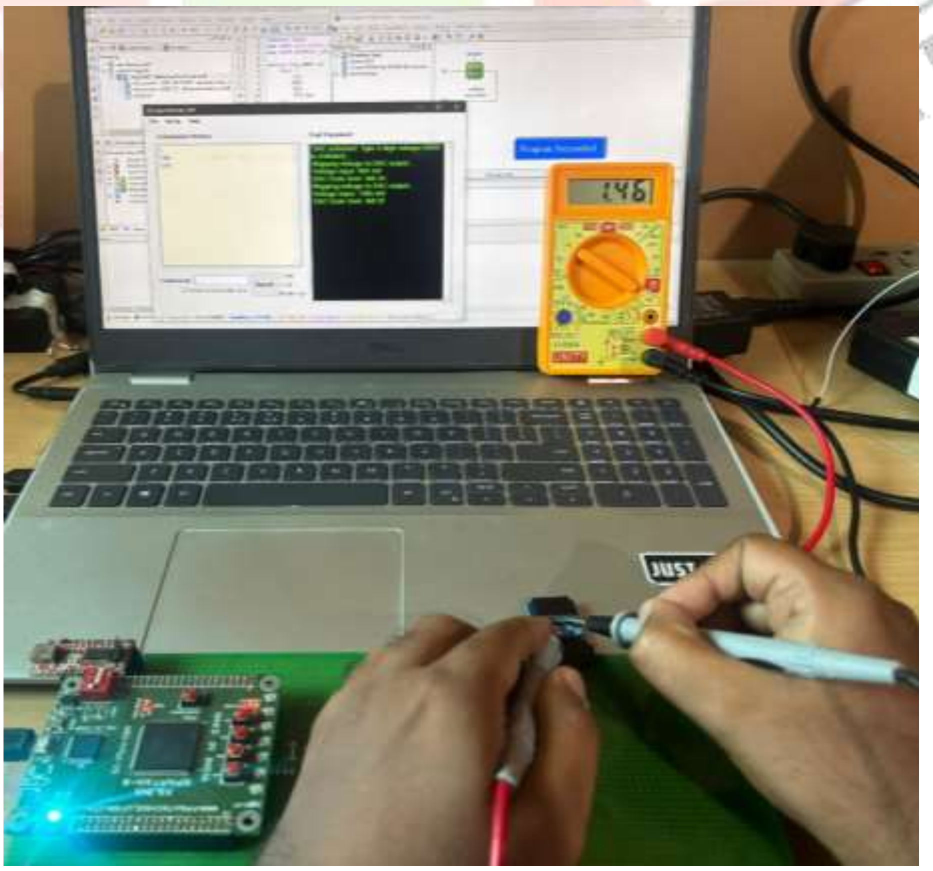


Figure 16. Shows the response for input 150 (150 mV), and the observed analog output was ~1.56 V, verifying correct DAC operation for mid-range values.

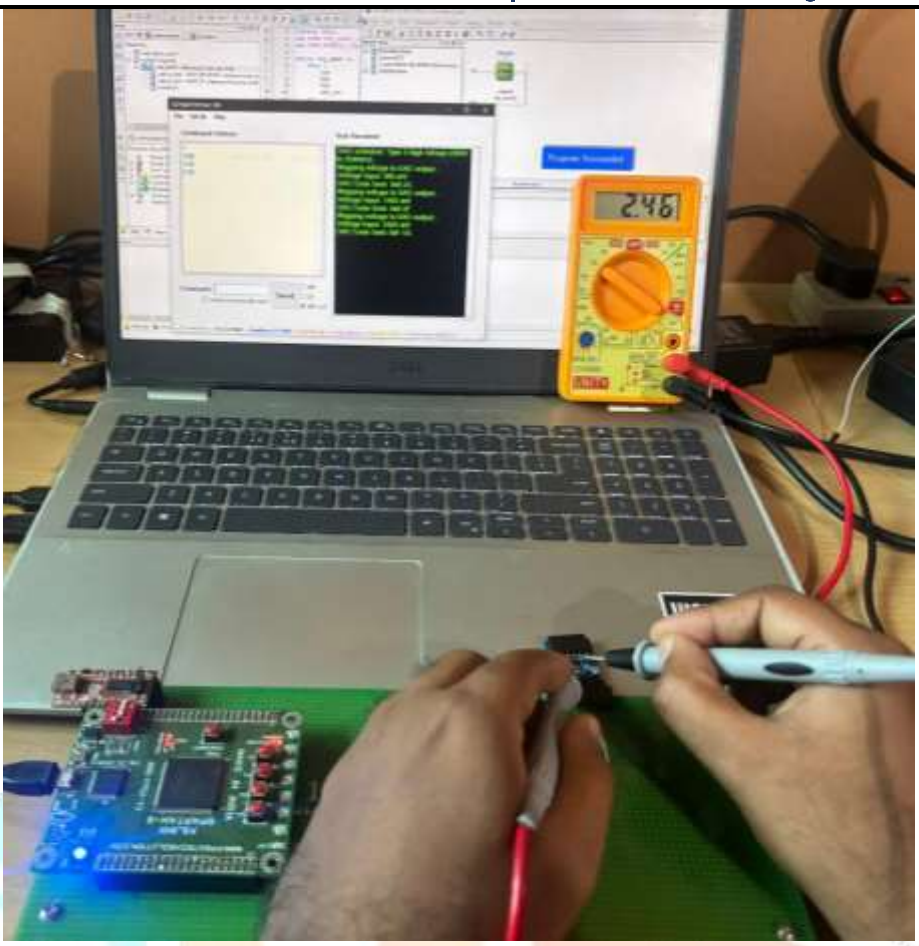


Figure 17. Corresponds to the input 250 (250 mV), and the DAC output was measured at ~2.46 V, further confirming linear scaling of input to output.

V. CONCLUSION

The digital signal generator system has been successfully designed and implemented on an FPGA platform using VHDL. The digital logic responsible for generating waveform control signals and handling user commands via a UART interface has been thoroughly tested and verified. The UART module reliably interprets user-defined instructions such as 'GXX' and 'gXX', which are used to configure the waveform selection logic through shift registers. These configurations effectively control both parallel and serial data outputs.

The system's analog functionality has also been successfully implemented. A DAC121S101 digital-to-analog converter (DAC) was interfaced using the SPI protocol, allowing the generation of analog voltage levels corresponding to user input. The DAC receives 12-bit digital values from the FPGA and accurately converts them into analog voltages ranging from 0 V to 2.5 V. User inputs such as "100" (representing 100 mV) result in corresponding analog outputs (e.g., 1.0 V), which were measured using a multimeter and verified to be within acceptable accuracy.

Real-time output testing was conducted using a serial terminal and a multimeter, confirming that the system provides dynamic and stable voltage outputs based on software-controlled input. This confirms the successful integration of the digital and analog domains within the project and validates the system's functionality as a programmable voltage generator.

VI. FUTURE SCOPE

Building on the successful implementation of the DAC-based signal generation system, future enhancements can transform this setup into a compact, multifunctional laboratory instrument. By integrating an analog measurement system, such as an ADC with input protection and signal conditioning, the same FPGA platform can be extended to perform both signal generation and real-time signal measurement. This dual capability would allow the system to function as a basic oscilloscope or signal analyzer, enabling voltage waveform display and inspection in addition to generation. Such a device, combining DAC-based output and ADC-based input, could mimic essential functionalities of both a signal generator and a multimeter-sized oscilloscope. With the addition of a small display or serial plotting interface, this compact, USB-powered tool could serve as a portable test instrument ideal for embedded systems labs, field testing, or educational demonstrations.

REFERENCES

[1].Automated Inspection System for Assembled Printed Circuit Board Using Machine Vision HarishO. Goupale ,NeetuN. Gyanchandani, PranayA. Chauhan, Sneha C. Kumbhare, Twinkle B. Bhaisare 2023. In Saroj Hiranwal & Garima Mathur (eds.)

[2]. Jouppi, N. P. Design implications of memristor-based RRAM cross-point structures. In 2021 Design, Automation & Test in Europe, 1–6 (IEEE, 2021).

[3]. ARM Ltd. AMBA AXI and ACE Protocol Specification (2021).

[4]. Merced-Grafals, E. et al. Repeatable, accurate, and high speed multi-level programming of memristor 1t1r arrays for power efficient analog computing applications. Nanotechnology 27, 365202 (2021).

[5]. Chang, W., See, K. & Hu, B. Characterization of component under dc biasing condition using an inductive coupling approach. IEEE Trans. Instrum. Meas. 59, 2109–2114 (2020).

[6]. Upadhye, M. & Sathe, A. P. An integrated pc-based system for measurement of various parameters for two-terminal devices used in the industry. IEEE Trans. Instrum. Meas. 41, 706–709 (2019).

[7]. Tektronix. Tektronix MDO34 Datasheet (2022).

[8]. Wang, Y. et al. High speed test system of current pulse for phase change memory devices.J. Phys. Conf. Ser. 1237, 042064 (2019).

[9]. An FPGA-based system for generalised electron devices testing (2022) 12:13912 | https://doi.org/10.1038/s41598-022-18100-3.

[10]. IEEE-SA Standards Board, "IEEE Standard Test Access Port and Boundary-Scan Architecture." Institute of Electrical and Electronics Engineers, Inc, 200 ,(2019).