Journal of Information Systems Engineering and Management
2025, 10(15s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Enhancing Sustainable Farming Practices through FPGA Technology

K. Suganthi[1], Manikandan AVM[2], B. Vishwanath Reddy[3], S. Karun Kumar[4]

[1,2,3,4]*Department of ECE, SRM Institute of Science and Technology, Kattankulathur, India.*
[1]*suganthk@srmist.edu.in,* [2]*manikanm@srmist.edu.in,* [3]*br5640@srmist.edu.in,* [4]*ks5945@srmist.edu.in*

ARTICLE INFO

Received: 02 Dec 2024

Revised: 25 Jan 2025

Accepted: 02 Feb 2025

ABSTRACT

Sustainable farming is essential to address the increasing demand for food while minimizing environmental impact. Field-Programmable Gate Arrays (FPGAs) provide a powerful and energy-efficient platform for real-time processing, which can be leveraged to optimize various aspects of agricultural management. This paper focuses on the use of FPGA technology to enhance real-time monitoring and control systems for irrigation, ensuring that crops receive optimal water levels based on current soil moisture conditions. By interfacing with soil moisture sensors, FPGA systems can collect data and immediately adjust irrigation schedules, preventing over-watering and conserving water resources. The low-power consumption and high-speed processing capabilities of FPGAs make them ideal for continuously monitoring moisture levels and activating irrigation systems as needed. Additionally, FPGA-based systems provide the flexibility to easily adjust to varying environmental conditions and crop types, making them highly adaptable for different farming scenarios. The implementation of FPGA in these areas promotes more precise and efficient water usage, contributing to the overall sustainability of farming operations by reducing water waste and maintaining soil health.

Keywords: sustainable farming, FPGA, real-time monitoring, irrigation systems, soil moisture sensors, water conservation, adaptive algorithms, energy efficiency, scalability, dynamic agricultural environments.

1. INTRODUCTION

Sustainable farming practices are essential in today's world due to the increasing demands for food and the escalating environmental impact of traditional agriculture. Water scarcity and high energy consumption in farming are critical issues that need to be addressed to ensure long-term agricultural productivity while minimizing environmental damage. Traditional irrigation systems are often inefficient, leading to water overuse and increased energy demands. Inefficient resource management not only increases operational costs but also exacerbates environmental concerns such as groundwater depletion and carbon emissions from energy-intensive operations platforms. In light of these challenges, there is a pressing need for smarter, more efficient irrigation management systems [1], [2]. While modern irrigation systems equipped with sensors and controllers have shown promise, they often rely on microcontrollers or general-purpose processors that are limited in terms of real-time data processing, scalability, and power efficiency. These systems typically operate with fixed schedules, regardless of real-time environmental conditions, leading to suboptimal water usage [3], [4].

Field Programmable Gate Arrays (FPGAs) are a powerful alternative to traditional processing systems, offering advantages such as parallel processing, low power consumption, and real-time reconfigurability. These features make FPGAs an ideal platform for developing an intelligent irrigation system that can process environmental data in real time and adapt irrigation patterns accordingly. Unlike fixed-function processors, FPGAs can be reprogrammed on-site to accommodate new algorithms or respond to changing requirements, making them particularly suitable for dynamic agricultural environments [3], [5], [6], [7].

2. OBJECTIVES

The objective of this research is to design and implement an FPGA-based irrigation system that optimizes water usage while minimizing energy consumption. The proposed system will collect real-time environmental data, process it using adaptive algorithms, and control irrigation mechanisms based on crop needs and external weather conditions. By doing so, the system aims to reduce waste and promote sustainable agricultural practices.

A key element in regulating the movement of water and heat energy through evaporation and plant transpiration between the earth's surface and the atmosphere is soil moisture. Therefore, the creation of precipitation and the evolution of weather patterns are significantly influenced by soil moisture. Therefore, controlling the beneficial amount of water for the soil will be made easier by using a specific soil sensor and integrating it with FPGA and Verilog code [8], [9], [10]. Small-scale integrated (SSI) circuits originated from early integrated circuits created in the 1960s, which had under 100 transistors per chip. Over the decades, the number of transistors on chips increased dramatically, reaching up to 1 billion today. This exponential growth in digital logic capacity posed challenges for designers, prompting the development of field programmable gate arrays (FPGAs) in the mid-1980s. FPGAs use RAM-based lookup tables instead of traditional AND-OR gates and consist of customizable logic blocks (CLBs) and I/O blocks. As digital circuit design evolved, particularly in the late 1980s and early 1990s, traditional design techniques became less viable. Today, computer-aided design and Hardware Description Languages (HDLs), such as VHDL, are essential for developing digital circuits, allowing for early error correction and easier debugging, especially for large systems [7], [11], [12].

3. DESIGN STRATEGIES

Figure 1 illustrates the basic model design strategy in a flow chart format. The process begins at the algorithmic level, which is comparable to conditional statements such as "if," "case," and loop statements commonly found in C code. Next, it moves to the Register Transfer Level (RTL), where registers are connected through Boolean equations. From there, the design progresses to the gate level, where components are linked using gates like AND and NOR. Finally, the process reaches the switch level, where MOS transistors are integrated into the gates and function as switches [5].

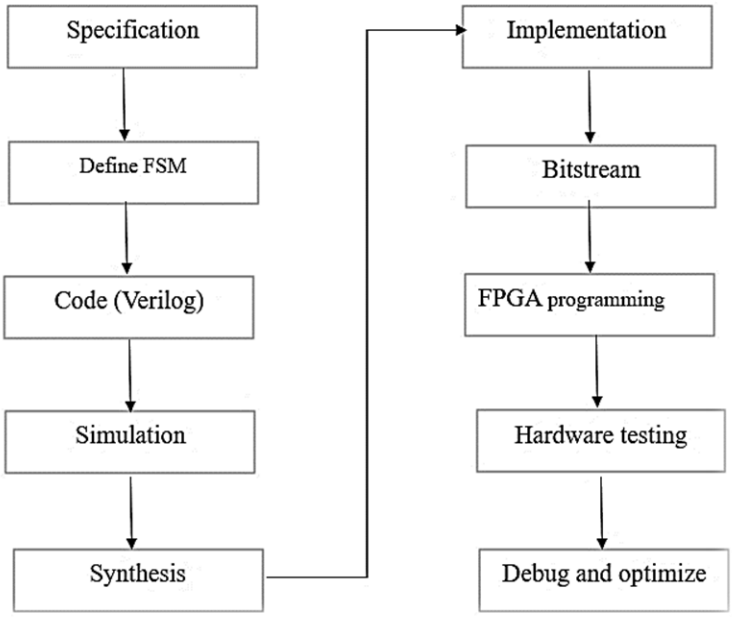

Fig.1 - Flow chart for FPGA design

The design flow of a Field-Programmable Gate Array (FPGA) involves several key steps from the initial project conception to final testing. It starts with the specification phase, where system requirements are gathered and documented. Stakeholders define the objectives and functionalities of the FPGA, ensuring that the final product meets user needs and expectations. This foundational phase is critical for guiding the subsequent design efforts [13], [14].

The next step involves defining the Finite State Machine (FSM), which models the system's operational flow as a state machine. This abstraction helps designers understand how the FPGA transitions between states based on different inputs, simplifying decision-making and aiding in debugging. Once the FSM is established, the design moves to the coding phase, where the specifications and FSM are translated into Verilog, detailing the FPGA's functionality. Attention to detail is crucial during coding, as errors can carry through the development process. [13].

After the Verilog code is written, it undergoes simulation to verify its correctness. Simulation tools assess the code's behavior under various input conditions, allowing designers to identify and fix issues before moving to the synthesis stage. During synthesis, the high-level Verilog code is transformed into a gate-level representation, resulting in a netlist that details the logic gates and interconnections needed for FPGA implementation. This step is essential for bridging high-level design and low-level hardware. Following synthesis, the netlist is placed and routed onto the FPGA fabric. This implementation process maps the design to the FPGA's physical resources, directly affecting the performance and efficiency of the final hardware implementation [14].

After successfully placing and routing the design, a bitstream file is generated, containing the configuration data needed to program the FPGA. This moment is pivotal as it marks the completion of the logical design and prepares for hardware programming. The bitstream is then loaded onto the FPGA, which configures it to perform tasks in real-time. Following programming, the design undergoes hardware testing to ensure it functions correctly under real-world conditions. If issues arise, the design enters the debug and optimize phase, where engineers diagnose problems and implement corrections, which may involve revisiting earlier stages. Once functioning properly, optimizations can enhance performance and efficiency, highlighting the need for flexibility in the development process [7].

4. METHODS

4.1. Irrigation control

In the context of our project, "Enhancing Sustainable Practices through FPGA Technology," effective irrigation control is paramount for optimizing water usage. This section delineates the methodologies implemented within the irrigation control system, detailing input-output mechanisms, interval time calculations, motor control logic, and LED dimming through Pulse Width Modulation. Figure 2 illustrates these processes in a flowchart format, providing a comprehensive overview of the system's architecture and functionality. By integrating these techniques, we aim to foster sustainable agricultural practices through precise and efficient irrigation management.

4.1.1. Inputs and Outputs Overview

This system takes several inputs and produces outputs that control hardware components like a motor and LEDs. The inputs include an array of switches (`SW[17:0]`) and a clock signal (`CLOCK_50`). The switches are used to configure the irrigation system. Specifically, `SW[5:0]` selects the time interval (in hours) between motor activations, which is essential for controlling irrigation cycles. Other switches, such as `SW[17:14]` and `SW[13:11]`, adjust the brightness of red and blue LEDs using Pulse Width Modulation (PWM). The output ports include `GPIO[20:0]` to control LEDs and the irrigation motor, and `HEX0` and `HEX1`, which display the selected time interval on seven-segment displays.

4.1.2. Interval Time Calculation (The `lim` Register)

The system uses the `lim` register to store the interval time, which determines when the irrigation motor should turn on. The value of `lim` is selected based on the hour chosen by `SW[5:0]`. This design uses a formula to calculate the interval time for each hour: `d99999999 + ((d149999999 / 24) * hour)'. The formula accounts for the base interval and adds time depending on the selected hour. For example, when `SW[5:0]` represents 1 hour, the interval is `106249999`; for 24 hours, it is `249999998`. This calculation ensures that the irrigation motor turns on after a specific delay.

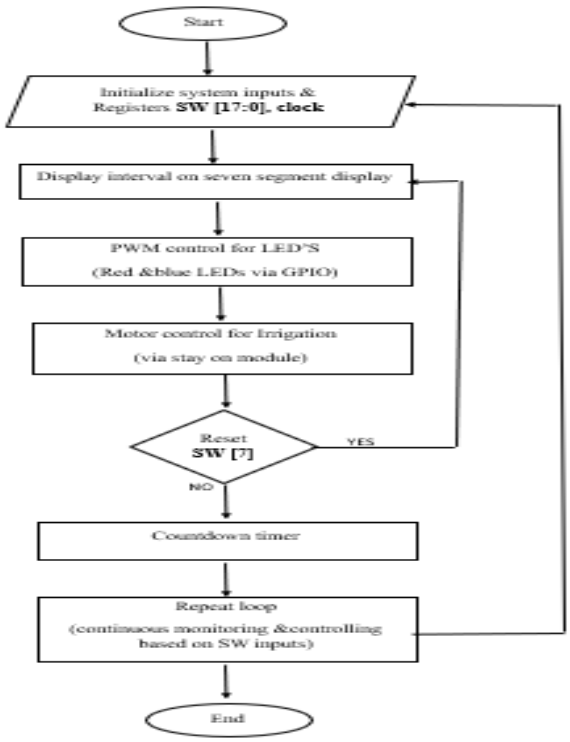

Fig.2 - Irrigation control mechanisms

4.1.3. Seven-Segment Display for Time Intervals

Two seven-segment displays (`HEX0` and `HEX1`) are used to show the current irrigation interval. The `SevenSegmentDecoder` modules convert the switch input (`SW[5:0]`) into a format that the displays can represent. `HEX0` shows the lower 4 bits (`SW[3:0]`), while `HEX1` shows the upper 2 bits (`SW[5:4]`). This allows the user to visually verify the selected irrigation hour on the displays.

4.1.4. Motor Control Logic

The motor is controlled using the `stay_on` module, which keeps the motor on for a defined period based on the calculated interval stored in `lim`. This module uses the system clock (`CLOCK_50`) to count down from the value stored in `lim`. When the countdown reaches zero, the motor turns on for a short duration (2 seconds, as indicated by the `stay_on` input being set to `28'd99999999`). After the motor has been on for the required time, it turns off until the next interval. If the reset signal (`SW[7]`) is activated, the countdown is reset immediately.

4.1.5. LED Dimming Using PWM

The system controls the brightness of red and blue LEDs using Pulse Width Modulation (PWM). Two `LED_PWM` modules handle this: one for the red LED (`GPIO[1]`) and another for the blue LED (`GPIO[0]`). Each PWM module takes a portion of the switch inputs to determine the duty cycle. For example, the red LED uses inputs from `SW[17]` and `SW[13:11]`, while the blue LED uses inputs from `SW[17:14]`. The PWM duty cycle controls how much time the LED remains on during each cycle, effectively dimming or brightening the LEDs depending on the switch settings.

4.2. Soil Moisture Monitoring

The soil_moisture_monitor module is a crucial component of an automated irrigation system that intelligently controls a water pump based on the soil moisture levels detected by a sensor. The design focuses on ensuring that the soil remains within a desired moisture range, promoting healthy plant growth while conserving water [8], [9], [11], [15].

4.2.1. Inputs and Outputs

The module has several key inputs and outputs that facilitate its operation. The clock input (`clk`) serves as the timing mechanism, enabling the module to execute state transitions and logic operations synchronously. The reset input

(`reset`) allows the system to return to a known starting condition, ensuring reliability and predictability in operation. The moisture level input (`moisture_level`) is a 10-bit digital value representing the current moisture level in the soil, as measured by an Analog-to-Digital Converter (ADC) connected to the soil moisture sensor. This ADC value provides a quantifiable measure of how wet or dry the soil is, allowing the system to make informed decisions about irrigation. The output signal (`water_pump`) is a digital control signal that directly activates or deactivates the water pump. When the signal is set to `1`, the pump is turned on to irrigate the soil; when it is set to `0`, the pump is turned off, ceasing irrigation [16].

4.2.2. Moisture Thresholds

To effectively manage irrigation, the module employs two predefined moisture thresholds: `DRY_THRESHOLD` and `WET_THRESHOLD`. The `DRY_THRESHOLD` indicates the moisture level below which the soil is considered dry and in need of watering. Conversely, the `WET_THRESHOLD` signifies the moisture level above which the soil is deemed sufficiently wet, at which point the irrigation should stop. These thresholds are crucial for preventing overwatering, which can lead to root rot and other plant health issues, while also ensuring that the soil does not become too dry, which can stress plants and hinder growth. The thresholds can be adjusted according to specific soil and plant requirements, making the module versatile for different agricultural contexts.

4.2.3. Finite State Machine (FSM)

The `soil_moisture_monitor` module's core features a finite state machine (FSM), as shown in Figure 3, comprising two states: DRY (noted as `1'b0`) and WET (noted as `1'b1`). This FSM governs the behavior of the water pump based on the moisture levels detected. In the DRY state, the system recognizes that the soil is too dry; therefore, it activates the water pump by setting the `water_pump` output to `1`. This action initiates the irrigation process to bring moisture back into the soil. The system continually monitors the moisture level, and if it detects that the level exceeds the `WET_THRESHOLD`, it transitions to the WET state, turning off the pump by setting `water_pump` to `0`. This prevents excess watering and promotes water conservation.

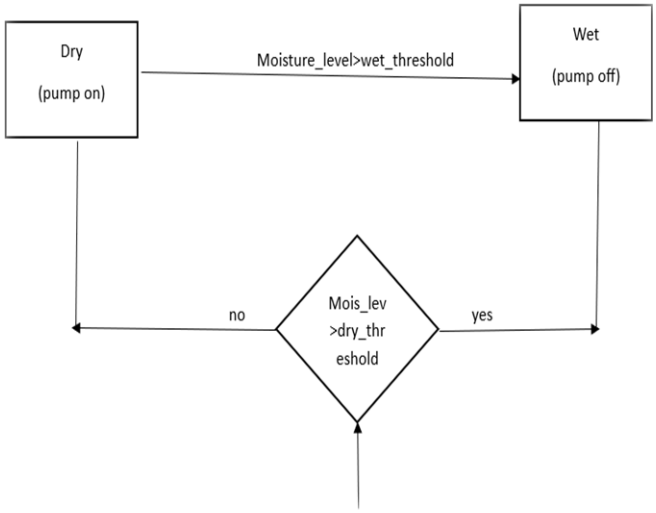

Fig.3 - FSM for soil moisture monitoring

Conversely, if the system finds itself in the WET state and the moisture level subsequently drops below the `DRY_THRESHOLD`, it transitions back to the DRY state, reactivating the pump to ensure the soil receives the necessary water. This continuous feedback loop allows the system to respond dynamically to changing soil moisture conditions. The transition logic is implemented using a combination of sequential and combinational logic. The FSM updates its state based on the current moisture readings on the rising edge of the clock signal, ensuring timely and efficient responses to environmental conditions [17].

4.2.4. State Transition Logic

The state transition logic is structured to guarantee that the system behaves predictably under various scenarios. On receiving a reset signal, the FSM initializes to the DRY state, which is critical for ensuring that the irrigation process

begins immediately after a reset, regardless of the previous state. This feature is particularly important in automated systems where unexpected resets may occur due to power outages or system reboots.

The combination of the moisture level input, thresholds, and FSM allows the `soil_moisture_monitor` module to function effectively as an irrigation controller. By integrating this module into an overall irrigation system, farmers and gardeners can ensure that their plants receive the right amount of water when they need it, optimizing growth and resource usage. This intelligent automation reduces the labor required for manual irrigation and helps maintain optimal soil conditions, contributing to sustainable agricultural practices [10], [18], [19].

5. RESULTS AND DISCUSSION

5.1. Irrigation Control System

In modern agricultural practices, efficient water management is crucial for sustainable farming. The irrigation control system presented herein continuously monitors soil moisture levels to optimize water usage through precise actuation of a water pump based on predefined threshold values, contributing to sustainable farming practices. Utilizing a 10-bit digital signal from a soil moisture sensor, the system incorporates comparator blocks to assess moisture levels and make informed state transitions. The underlying RTL schematic, depicted in Figure 4, illustrates the control logic governing these processes, including the use of multiplexers for decision-making. Furthermore, the simulation results in Figure 5 showcase the system's response to varying soil moisture conditions, validating its effectiveness in activating and deactivating the water pump as needed. This innovative approach not only conserves water but also enhances crop yield by ensuring adequate soil moisture at all times.

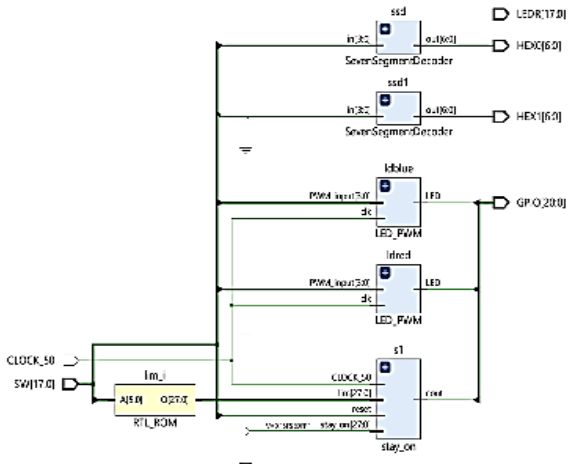

Fig. 4 - RTL for irrigation control

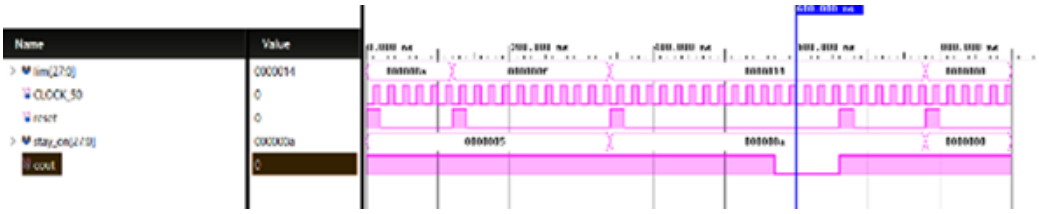

Fig. 5 - System's response to varying soil moisture conditions

The input moisture_level[9:0] represents a 10-bit digital signal corresponding to the soil moisture sensor's reading. This value is compared against preset threshold values using comparator blocks such as RTL_LT (less than) and RTL_GT (greater than), which determine whether the moisture level is below or above the thresholds. The comparators drive the state transitions within the system, with the next_state signals determining the control flow based on moisture conditions. The design uses multiplexer (RTL_MUX) blocks to select the next state of the control logic. These MUXes decide the operation of the water pump based on the current state of the soil moisture. The state_reg block stores the current state of the system, ensuring that the system remembers the previous state and can transition appropriately. This enables the system to effectively decide when to turn the water pump on or off.

The water_pump_i signal, which controls the water pump, is ultimately determined by the moisture level inputs and the system state. The output from the MUXes drives the final output water_pump, activating the irrigation system when soil moisture falls below the predefined threshold, and deactivating it once sufficient moisture is detected.

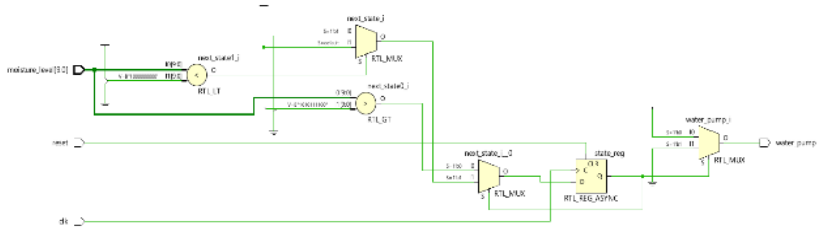

Fig. 6 - Schematic diagram for soil moisture monitoring

The schematic outlined in Figure 6 presents a modular approach to control logic in precision farming, focusing on real-time monitoring of moisture levels to optimize automated irrigation and minimize water waste. This RTL-level design is efficient, scalable, and compatible with multiple sensors, enhancing complex irrigation management systems. Using FPGAs allows for real-time processing, ideal for time-sensitive agricultural operations. The asynchronous register block (RTL_REG_ASYNC) ensures reliable state transitions, improving robustness in changing environmental conditions. Overall, this design integrates technology with agriculture, providing an efficient and adaptable solution for better water management in farming.

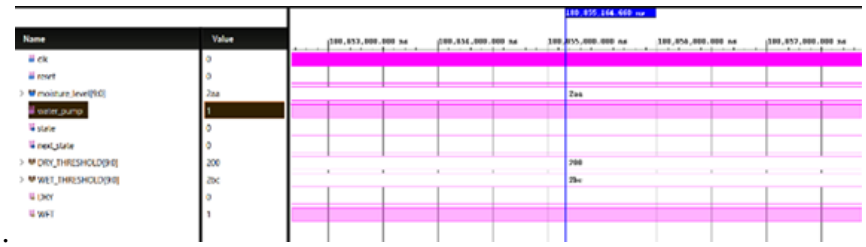

Fig. 7 - Simulation result for soil moisture monitoring

The soil moisture monitoring and irrigation control system was successfully implemented using the PYNQ-Z2 FPGA board, as shown in Figure 8. This Xilinx-based platform features the Zynq-7000 SoC, which integrates a dual-core ARM Cortex-A9 processor with FPGA fabric. The design was described using Verilog, synthesized, and programmed onto the PYNQ-Z2 board. The hardware implementation includes a finite state machine (FSM) to control the states of the system—whether to activate or deactivate the water pump—based on real-time soil conditions [8], [20], [21], [22]. After synthesis, the system showed low resource utilization on the FPGA fabric, with minimal usage of LUTs, flip-flops, and I/O blocks. The implementation is highly resource-efficient, leaving ample space for future system expansion, such as integrating additional sensors or more complex control logic. Furthermore, the use of the PYNQ-Z2 board allows for a Python-based overlay in future work, enabling quick prototyping and higher-level software integration.

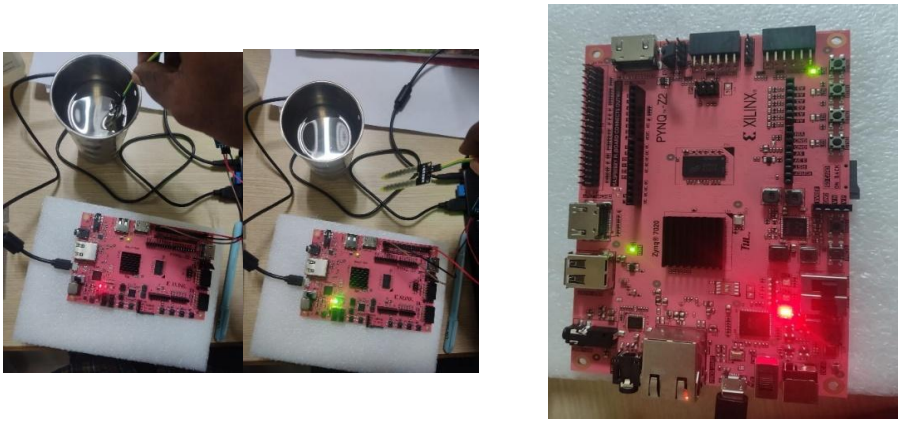

Fig. 8 – The PYNQ-Z2 FPGA board showing the soil moisture sensor status and the motor signal

5.2. FPGA Resource Utilization for Sustainable Farming Application

In the design flow of our FPGA-based sustainable farming system, efficient resource utilization is critical for achieving the desired performance while minimizing power consumption and hardware complexity. The post-synthesis FPGA utilization reports highlight key aspects of this efficiency, particularly in terms of logic resource usage, input/output (I/O) interface, and clock distribution.

In the post-synthesis report (Figure 9), the Look-Up Table (LUT) and flip-flop (FF) utilization is just 1%, indicating highly efficient core logic for sensor data processing, irrigation control, and decision-making. The system relies on simple, optimized operations suited for FPGA architecture. The I/O utilization, as shown in Table 1, is relatively high at 23%, reflecting significant interaction with external devices like moisture and temperature sensors, which is typical in precision agriculture. BUFG (Global Buffers) utilization is moderate at 3%, supporting synchronized timing without excessive resource demands. In the subsequent report (Figure 10), I/O utilization drops to 4%, while LUT, FF, and BUFG usage remain constant. This decline suggests design optimizations that reduce external connections, enhancing resource efficiency while keeping core functionality intact. Overall, low LUT and FF utilization, along with moderate BUFG use, demonstrates that the system is well-optimized for sustainable farming needs. It efficiently manages real-time data from sensors, controls irrigation, and minimizes power and resource demands, showcasing the flexibility of FPGA-based systems in precision agriculture.

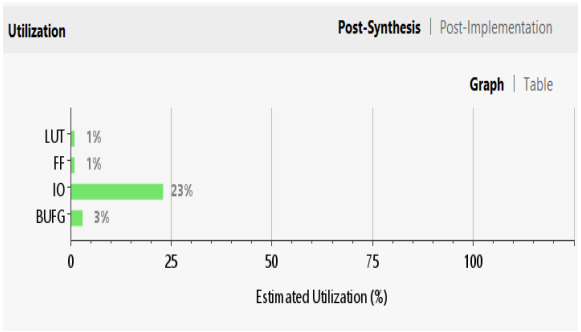

Fig. 9 - Utilization graph of FPGA (PYNQ-Z2): irrigation control

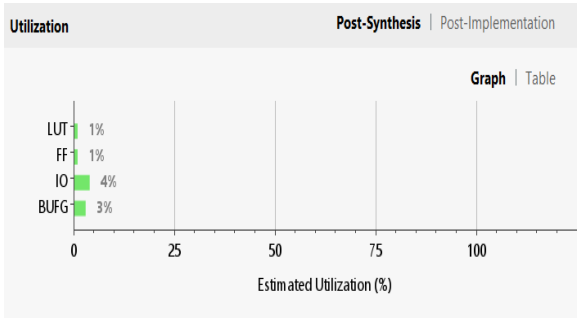

Fig. 10 - Utilization graph of FPGA (PYNQ-Z2): soil moisture monitoring

Table 1 – Post-synthesis report

Parameter	(PYNQ-Z2)	Spartan-6	Cyclone V
FPGA Platform	PYNQ-Z2 (Zynq-7000)	Spartan-6	Cyclone V
Clock Frequency	100 MHz	75 MHz	85 MHz
Power Consumption	1.2 W	1.3 W	1.5 W
Response Delay	<1 µs	2 µs	1.5 µs
LUT Utilization	1%	10%	12%
Flip-Flop Utilization	1%	8%	9%
I/O Utilization	23%	30%	28%
Max Frequency (Fmax)	95 MHz	65 MHz	80 MHz

This resource-efficient FPGA design presents a promising approach for sustainable farming, enabling real-time environmental monitoring and control with minimal hardware complexity. The findings from these synthesis reports underline the effectiveness of the design and its potential for scaling up to larger, more complex farming operations without significant increases in hardware resource consumption.

6. CONCLUSION

The use of FPGA-based systems in sustainable farming presents a powerful solution for addressing the increasing demands of precision agriculture while conserving critical resources like water and energy. This paper demonstrates the design and implementation of a soil moisture monitoring and automated irrigation system using an FPGA, emphasizing efficient resource utilization, real-time processing, and scalability. The low logic utilization observed in post-synthesis reports, combined with the flexibility of the FPGA, makes this approach ideal for applications that require responsive and adaptive control over environmental factors such as soil moisture. By leveraging the inherent parallelism and reconfigurability of FPGAs, the system effectively manages sensor data and actuates irrigation systems with minimal delay, ensuring optimal crop hydration while reducing water waste. Additionally, the modular design can be easily expanded to incorporate multiple sensors or extended control mechanisms, making it a versatile solution for various agricultural settings.

The results show that the PYNQ-Z2 FPGA board provides a robust and efficient platform for implementing real-time soil moisture monitoring and irrigation control. This FPGA technology offers significant benefits in sustainable farming applications, including energy efficiency, high-performance computing, adaptability to different crops or conditions, and the ability to optimize resource usage. As the need for more efficient and scalable farming systems grows, FPGA-based designs hold great potential for enhancing productivity and sustainability in agriculture, contributing to more resilient and environmentally friendly farming practices.

ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to the management of SRM Institute of Science and Technology, Kattankulathur campus, for providing the necessary research facilities that made this study possible. We also extend our heartfelt thanks to the Department of ECE for their constant moral support and motivation throughout this research. Their guidance and encouragement have been invaluable in the completion of this work.

REFERENCES

[1] A. Mitra *et al.*, “Smart Agriculture: A Comprehensive Overview,” *SN Comput. Sci.*, vol. 5, no. 8, p. 969, Oct. 2024, doi: 10.1007/s42979-024-03319-w.

[2] M. Dhanaraju, P. Chenniappan, K. Ramalingam, S. Pazhanivelan, and R. Kaliaperumal, “Smart Farming: Internet of Things (IoT)-Based Sustainable Agriculture,” *Agriculture*, vol. 12, no. 10, p. 1745, Oct. 2022, doi: 10.3390/agriculture12101745.

[3] M. A. Cusi Huarancca, L. C. León Huarache, R. Cristobal Holguino, and R. R. Sulla Torres, “Design and Development of an Integrated Monitoring and Automated Irrigation System with IoT Technologies for Sustainable Agriculture in Arequipa, Peru,” *Int. J. Electr. Electron. Eng.*, vol. 11, no. 6, pp. 31–40, Jun. 2024, doi: 10.14445/23488379/IJEEE-V11I6P104.

[4] S. D. Alwis, Z. Hou, Y. Zhang, M. H. Na, B. Ofoghi, and A. Sajjanhar, “A survey on smart farming data, applications and techniques,” *Comput. Ind.*, vol. 138, p. 103624, Jun. 2022, doi: 10.1016/j.compind.2022.103624.

[5] Z. Lai and Y. Dai, “An Irrigation Control System Based on an FPGA,” in *2012 Second International Conference on Instrumentation, Measurement, Computer, Communication and Control*, Harbin City, Heilongjiang, China: IEEE, Dec. 2012, pp. 159–163. doi: 10.1109/IMCCC.2012.44.

[6] S. R. Patil, N. S. Joshi, R. K. Kamat, and P. K. Gaikwad, “Design of Field Programmable Gate Array-based Soil Moisture Monitoring System for Precision-Agriculture,” vol. 12, no. 0898, 2021.

[7] M. Vivekanandan and S. Kanaga Suba Raja, “Virtex-II Pro FPGA Based Smart Agricultural System,” *Wirel. Pers. Commun.*, vol. 125, no. 1, pp. 119–141, Jul. 2022, doi: 10.1007/s11277-022-09544-x.

[8] M. I. Husni, M. K. Hussein, M. S. B. Zainal, A. B. Hamzah, D. B. M. Nor, and H. B. M. Poad, “Soil Moisture Monitoring Using Field Programmable Gate Array,” *Indones. J. Electr. Eng. Comput. Sci.*, vol. 11, no. 1, p. 169, Jul. 2018, doi: 10.11591/ijeecs.v11.i1.pp169-174.

[9] A. Oukaira, A. Z. Benelhaouare, E. Kengne, and A. Lakhssassi, "FPGA-Embedded Smart Monitoring System for Irrigation Decisions Based on Soil Moisture and Temperature Sensors," *Agronomy*, vol. 11, no. 9, p. 1881, Sep. 2021, doi: 10.3390/agronomy11091881.

[10] S. S. Shinde, J. M. Yewale, V. Ali, and S. M. Mahamuni, "FPGA based wireless communication system plant monitoring using FPGA," in *2017 International Conference on Communication and Signal Processing (ICCSP)*, Chennai: IEEE, Apr. 2017, pp. 2188–2191. doi: 10.1109/ICCSP.2017.8286798.

[11] T. Kavya, A. B, and R. K. Megalingam, "A Verilog-based Design for Real-Time Data Processing and Control in Agricultural Fields using FPGAs," in *2024 12th International Conference on Internet of Everything, Microwave, Embedded, Communication and Networks (IEMECON)*, Jaipur, India: IEEE, Oct. 2024, pp. 1–6. doi: 10.1109/IEMECON62401.2024.10846579.

[12] A. Muharemović, D. Jokić, M. Simeunović, and H. Hanjalić, "FPGA Technologies for Smart and Sustainable Agriculture: A Comprehensive Overview," in *2023 12th Mediterranean Conference on Embedded Computing (MECO)*, Budva, Montenegro: IEEE, Jun. 2023, pp. 1–5. doi: 10.1109/MECO58584.2023.10155051.

[13] P. A. Simpson, *FPGA Design: Best Practices for Team-based Reuse*. Cham: Springer International Publishing, 2015. doi: 10.1007/978-3-319-17924-7.

[14] U. Farooq, Z. Marrakchi, and H. Mehrez, "FPGA Architectures: An Overview," in *Tree-based Heterogeneous FPGA Architectures*, New York, NY: Springer New York, 2012, pp. 7–48. doi: 10.1007/978-1-4614-3594-5_2.

[15] K. Lakshmisudha, S. Hegde, N. Kale, and S. Iyer, "Smart Precision based Agriculture using Sensors," *Int. J. Comput. Appl.*, vol. 146, no. 11, pp. 36–38, Jul. 2016, doi: 10.5120/ijca2016910916.

[16] Tarun. Vedulla, Y. M. Reddy, A. Kalyan, and R. Jenila, "VLSI Architecture for Smart and Precision Agriculture Using Sensors," *Int. J. Adv. Sci. Comput. Eng.*, vol. 3, no. 1, pp. 18–27, Jun. 2021, doi: 10.62527/ijasce.3.1.32.

[17] S. S. Mathurkar, N. R. Patel, R. B. Lanjewar, and R. S. Somkuwar, "Smart sensors based monitoring system for agriculture using field programmable gate array," in *2014 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2014]*, Nagercoil, Tamil Nadu, India: IEEE, Mar. 2014, pp. 339–344. doi: 10.1109/ICCPCT.2014.7054914.

[18] Y. Gharde, P. K. Singh, R. P. Dubey, and P. K. Gupta, "Assessment of yield and economic losses in agriculture due to weeds in India," *Crop Prot.*, vol. 107, pp. 12–18, May 2018, doi: 10.1016/j.cropro.2018.01.007.

[19] A. Saddik, R. Latif, and A. El Ouardi, "Low-Power FPGA Architecture Based Monitoring Applications in Precision Agriculture," *J. Low Power Electron. Appl.*, vol. 11, no. 4, p. 39, Sep. 2021, doi: 10.3390/jlpea11040039.

[20] S. R. Prathibha, A. Hongal, and M. P. Jyothi, "IOT Based Monitoring System in Smart Agriculture," in *2017 International Conference on Recent Advances in Electronics and Communication Technology (ICRAECT)*, Bangalore, India: IEEE, Mar. 2017, pp. 81–84. doi: 10.1109/ICRAECT.2017.52.

[21] A. K. Singh, "Smart Farming: Applications of IoT in Agriculture," in *Handbook of Smart Materials, Technologies, and Devices*, C. M. Hussain and P. Di Sia, Eds., Cham: Springer International Publishing, 2022, pp. 1655–1687. doi: 10.1007/978-3-030-84205-5_114.

[22] V. K. Quy *et al.*, "IoT-Enabled Smart Agriculture: Architecture, Applications, and Challenges," *Appl. Sci.*, vol. 12, no. 7, p. 3396, Mar. 2022, doi: 10.3390/app12073396.