Configurable FPGA-Based Watchdog Timer for Real-Time Fault Detection in SCADA Energy Systems

A. Soke[1*], L. C. Ngugi[2], J. Adebisi[3]

[1]Pan African University Institute for Basic Sciences, Technology and Innovation (PAUSTI) Nairobi, Kenya.
[2]Department of Telecommunication and Information Engineering, Jomo Kenyatta University of Agriculture and Technology (JKUAT), Nairobi, Kenya.
[3]Department of Electrical Engineering and Computer Engineering, University of Namibia, Windhoek, Namibia.

ABSTRACT: Supervisory Control and Data Acquisition (SCADA) systems play a critical role in modern energy infrastructures, where reliable real-time fault detection is essential for maintaining operational safety and system availability. However, software execution faults, timing violations, and communication disturbances can compromise the effectiveness of conventional monitoring mechanisms. This work presents a configurable Field-Programmable Gate Array (FPGA)-based dual-window watchdog timer for real-time fault detection in SCADA-based energy systems. The objective is to provide deterministic hardware-level supervision capable of detecting abnormal software execution patterns and initiating appropriate recovery actions. The proposed dual-window watchdog architecture employs configurable service and frame windows implemented in Verilog Hardware Description Language (HDL) and integrates dedicated fault detection, fault classification, and reset-control logic. The watchdog detects missed-service, out-of-window service, and double-service faults while providing a dual-stage response mechanism consisting of fault signaling followed by delayed system reset. The design was implemented on a Xilinx Artix-7 FPGA and validated through simulation, synthesis, timing analysis, and experimental hardware measurements. Results demonstrated correct detection and classification of all considered fault conditions. Out-of-window service and double-service faults were detected within a single System Clock (SYSCLK) cycle after occurrence, whereas missed-service faults were detected upon frame-window expiration. Timing analysis confirmed stable operation up to 149.992 MHz with a critical-path delay of 5.421 ns and positive timing slack. Resource utilization remained below 1% of the available FPGA resources, with no inferred BRAM or DSP blocks. These results demonstrate the suitability of the proposed dual-window watchdog timer as an efficient, lightweight, and deterministic hardware supervision mechanism for SCADA-based energy systems.

KEYWORDS: FPGA, watchdog timer, SCADA, fault detection, real-time systems, reliability.

[Received May 24, 2026; Revised Jun. 22, 2026; Accepted Jun. 04, 2026] Print ISSN: 0189-9546 | Online ISSN: 2437-2110

I. INTRODUCTION

In safety-critical systems, ensuring reliability and fault tolerance is essential. This requirement becomes even more critical in industrial environments such as Supervisory Control and Data Acquisition (SCADA) systems, where failures in monitoring or control processes may lead to serious operational disruptions, equipment damage, or safety risks. SCADA-based industrial control systems form a critical part of modern infrastructures, including power grids, transportation systems, and industrial automation. However, their increasing interconnection with external networks has significantly raised both security and reliability concerns, as highlighted by (Pliatsios et al., 2020; Yipeng Zhang, Ye Li and Zhoujun Li, 2023). In such contexts, human intervention is often limited or delayed, which makes autonomous fault detection and recovery mechanisms indispensable. In addition, FPGA-based SCADA implementations have shown improved performance, reduced latency, and enhanced reliability compared to conventional PLC-based solutions, as reported by (Ahmed, 2023).

Watchdog timers (WDTs) are widely used hardware components for supervising the correct operation of embedded and real-time systems. Unlike conventional timers, a watchdog timer continuously monitors system execution by expecting periodic service events from the supervised application, commonly referred to as heartbeat signals, as described by (Nikola Zlatanov, 2014). During normal operation, the system regularly refreshes the watchdog to prevent it from expiring. However, in the presence of software failures, hardware faults, or abnormal execution conditions, the absence or incorrect timing of

*Corresponding author: soke.azaria@students.jkuat.ac.ke

doi: http://doi.org/10.63746/njtd.v23i2.4824

this service leads to a timeout event, which triggers corrective actions such as a system reset or transition to a safe state. This mechanism plays a key role in improving system reliability, particularly in cyber-physical and safety-critical environments, as discussed by (Khairullah and Elks, 2020; Solouki, Angizi and Violante, 2024).

In real-time systems, strict timing constraints must be respected to ensure correct operation. These systems are widely used in industrial control, energy management, and other mission-critical applications. External watchdog timers are particularly advantageous in such environments because they operate independently from the processor. As a result, reliable supervision can still be maintained even when the processor or its clock fails, as emphasized by (B. B. Manjula et al., 2021). Despite their widespread use, conventional watchdog timer architectures present several limitations. Most standard watchdog timers are effective in detecting system hangs or slow failures, but they often fail to capture fast faults or abnormal execution patterns. In addition, many implementations rely heavily on processor interaction, which introduces a potential single point of failure. Software-based watchdog timers further increase this risk, as failures in the scheduler or driver may compromise the monitoring of all system processes. Furthermore, modern embedded systems are increasingly exposed to fault injection attacks and hardware vulnerabilities, which can significantly impact system reliability, as demonstrated by (Breier and Hou, 2022). As a result, traditional watchdog solutions generally provide limited fault classification and insufficient diagnostic information, making fault identification and debugging more difficult.

To overcome these limitations, advanced watchdog architectures such as windowed watchdog timers have been introduced. These designs enforce stricter timing constraints by defining valid service intervals, allowing the detection of both early and late service events. Several studies, including those by (Kumar Misra and Swetha, 2022; Bafna and Verma, 2025) have shown that windowed watchdog architectures improve fault detection coverage. However, these approaches still face challenges related to configurability, scalability, and integration within SCADA-oriented industrial monitoring environments. More recent research has focused on enhancing watchdog performance and flexibility. Multi-tier and redundant watchdog architectures have been proposed to increase system robustness, as shown by (Alkady et al., 2025). FPGA-based implementations, on the other hand, offer greater flexibility and adaptability for real-time applications, as highlighted by (Mária Pohronská and Tibor Krajčovič, 2011; Supraja, Kumar and Prasad, 2025). In addition, hardware-based fault detection techniques, including redundancy and error-checking mechanisms, have been explored to further improve system reliability, as presented by (Maatallah et al., 2025; Mestiri, Barraj and Machhout, 2025). Nevertheless, many of these solutions still lack precise timing analysis, fine-grained fault classification, deterministic fault-detection latency, and predictable hardware-level fault response timing.

Although previous studies have demonstrated the benefits of windowed, redundant, and FPGA-based watchdog architectures, important limitations remain. Most existing solutions primarily focus on timeout detection and system reset mechanisms while providing limited support for explicit fault classification and deterministic fault-response timing. Furthermore, detailed timing characterization and integration within SCADA-oriented monitoring architectures are rarely addressed. A comparative summary of representative watchdog solutions and the proposed approach is presented in Section IV to highlight these differences.

To address these limitations, this work proposes a configurable FPGA-based dual-window watchdog timer for real-time fault detection in SCADA-based energy systems. The proposed design integrates multiple fault detection mechanisms, including missed-service detection, out-of-window service detection, and double-service detection, enabling accurate classification of system anomalies. A latency observation mechanism is also incorporated to evaluate the timing behavior of fault detection events. The watchdog is designed as a modular and configurable hardware unit, making it suitable for integration into real-time industrial control systems.

The main contributions of this work can be summarized as follows:

i. Design of a configurable dual-window watchdog timer architecture tailored for real-time and SCADA-based applications.
ii. Implementation of missed-service, out-of-window service, and double-service fault detection with explicit hardware-level fault classification.
iii. Integration of latency monitoring for fault-detection timing characterization and response analysis.
iv. FPGA-based implementation and validation using Verilog HDL, simulation, and timing analysis.

The remainder of this paper is organized as follows. Section II presents the theoretical analysis of the proposed watchdog timer architecture and fault detection logic. Section III describes the implementation methodology and integration of the watchdog into SCADA-oriented systems. Section IV presents the simulation and implementation results together with discussion. Finally, Section V concludes the paper.

II. THEORETICAL ANALYSIS

An efficient watchdog timer must be capable of detecting abnormal software execution and bringing the system back to a safe and known state. In safety-critical environments, it is also essential that the watchdog operates independently from the processor, uses a dedicated timing mechanism, and is able to trigger a hardware reset when a fault is detected (B. B. Manjula et al., 2021). The dual-window watchdog timer proposed in this work is designed for real-time SCADA-based energy systems and implemented on an FPGA platform to improve flexibility, configurability, and reliability. Similar to conventional hardware watchdogs, the proposed design operates independently from the processor and relies on timing supervision implemented in dedicated hardware. However, unlike basic watchdog implementations, it adopts a configurable dual-window watchdog architecture, where both the service window and frame window durations can be adjusted during system initialization (Bafna and Verma, 2025; Supraja, Kumar and Prasad,

2025). In the proposed system, a fault condition results in the assertion of a failure signal (WDFAIL). After a predefined delay, a reset signal (RSTOUT) is generated to bring the system back to a safe state. The delay between fault detection and reset allows the system to store diagnostic data, which is useful for fault analysis and debugging.

Conventional watchdog timers are effective in detecting system hangs caused by abnormal software execution, such as infinite loops. However, they often fail to detect fast faults or situations where the watchdog continues to be serviced despite incorrect system behavior. As a result, certain abnormal execution patterns may remain undetected (Manjudevi, 2021). To overcome this limitation, a windowed watchdog approach is adopted. In this architecture, the watchdog must be serviced within a predefined time interval. Servicing the watchdog too early or too late is considered a fault. This mechanism enables the detection of both slow and fast faults, thereby improving fault coverage and timing reliability, which are essential for real-time SCADA applications.

A. I/O Interface and Configuration

Figure 1 illustrates the input–output interface of the proposed FPGA-based watchdog timer. The design provides two main output signals: the watchdog failure signal (WDFAIL) and the reset output (RSTOUT). When the system reset input is asserted, the watchdog starts in a fault state, ensuring that the system initializes in a controlled and safe condition.

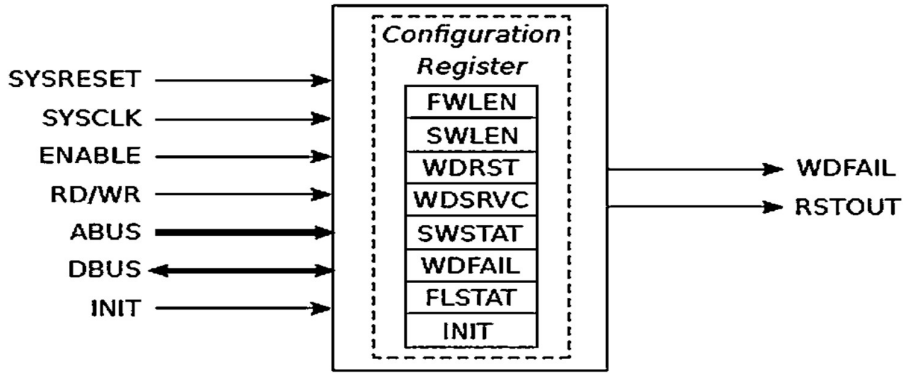

Figure 1 Input–output interface of the proposed FPGA-based watchdog timer.

The watchdog behavior is controlled through a configuration register composed of multiple fields. These fields allow both parameter configuration and runtime monitoring of the watchdog status. In particular, the WDRST and WDSRVC signals are used to reset and service the watchdog, respectively. The SWSTAT signal indicates whether the service window is active, while the FLSTAT field stores the detected fault condition. The watchdog is interfaced with the processor through standard control and data signals. Read and write operations are controlled using ENABLE and RD/WR signals, while address and data buses provide access to internal registers. This interface facilitates integration into FPGA-based embedded and industrial control systems.

The proposed architecture relies on two configurable timing intervals: the service window and the frame window. The service window defines the valid period during which the watchdog must be serviced, whereas the frame window represents the overall monitoring cycle. Typically, the service window is shorter than the frame window. Both parameters can be configured through the SWLEN and FWLEN fields during initialization (Kumar Misra and Swetha, 2022; Devi and Sreedhar, 2023).

To ensure system robustness, configuration registers may be protected against unintended modifications by restricting configuration access after initialization. Additional protection mechanisms may also be incorporated to further reduce the risk of accidental reconfiguration caused by software faults. The watchdog operation is initiated through an initialization signal. Once activated, the service window opens, and the system must generate a valid service signal within this interval. A correct service closes the service window and continues the monitoring cycle. The frame window can be configured according to the execution timing requirements of the monitored system, ensuring synchronization with real-time SCADA processes. This architecture reduces processor dependency and enables predictable hardware-level fault supervision.

B. Watchdog Timer Initialization

At power-up or after a system reset, the watchdog enters an initialization state where the WDFAIL signal is asserted, as illustrated in Figure 2. This behavior ensures that the system does not start operating without proper initialization, which is critical in safety-sensitive SCADA environments. During this initialization phase, the reset mechanism remains inactive, thereby preventing unintended reset generation before the watchdog has been correctly initialized.

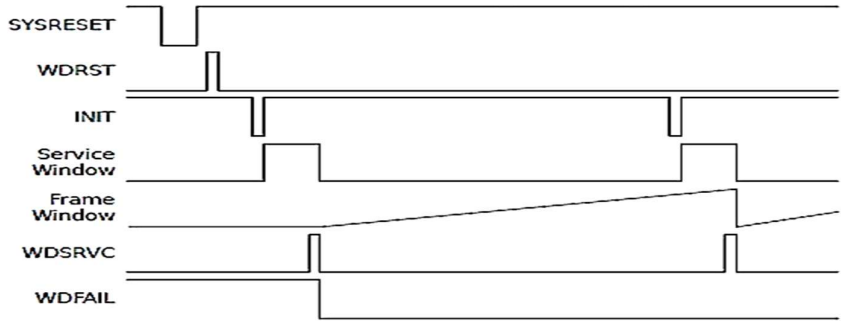

Figure 1 Timing diagram of watchdog initialization and service operation

To activate the watchdog, the reset control field must first be triggered. This initializes the internal logic and prepares the system for monitoring. The watchdog must then be serviced within the valid service window. Once a correct service is detected, the fault flag is cleared, and the system transitions to normal operation. During normal operation, the watchdog alternates between the service window and the frame window. Each valid service reinitializes the monitoring cycle, ensuring continuous supervision of system behavior. As long as the watchdog is serviced correctly within the defined timing constraints, no fault is triggered. The initial assertion of the WDFAIL signal serves as an indication that the system is not yet operational. Once the watchdog is properly initialized, the system can safely enter normal operation. During runtime, any abnormal behavior results in the assertion of the WDFAIL signal. This signal can be used by external supervisory logic to isolate faulty components and maintain overall system stability.

Following fault detection, a configurable reset-delay mechanism is activated. The delay interval can be adjusted according to application requirements and allows diagnostic information to be captured before the reset action is executed. After expiration of the configured delay period, the watchdog asserts the RSTOUT signal, forcing the supervised system to return to a known safe state.

The watchdog control logic is implemented using a finite state machine (FSM) comprising initialization, monitoring, alarm, and reset-hold states. A binary state-encoding scheme is adopted to minimize register utilization while maintaining deterministic operation and efficient FPGA implementation.

C. Fault Detection Mechanisms

To improve system reliability, several fault detection mechanisms are integrated into the proposed FPGA-based watchdog timer. These mechanisms are specifically designed to detect abnormal software execution patterns and ensure that both slow and fast faults are effectively captured in real time. The watchdog continuously monitors system behavior based on the configured service window and frame window. Any violation of the expected service timing constraints is interpreted as a fault condition. Unlike conventional watchdog implementations, the proposed design not only detects faults but also classifies them, enabling more precise identification of abnormal system behavior. This capability is particularly important in SCADA-based energy systems, where reliable timing supervision is required. Figure 3 illustrates the valid and faulty watchdog service conditions considered in the proposed design, including normal service operation, out-of-window service, missed-service, and double-service scenarios.

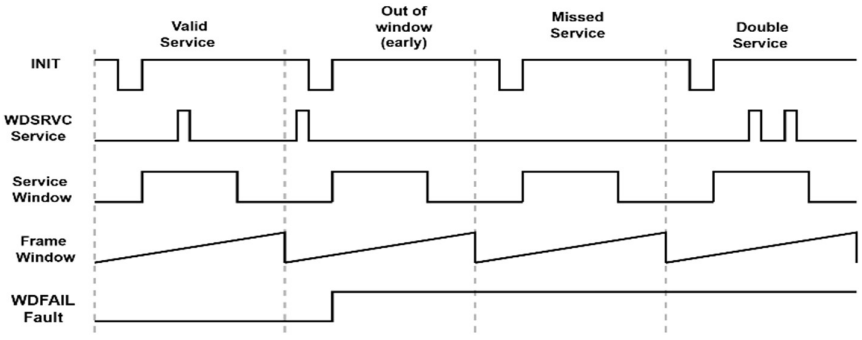

Figure 2 Timing diagram of valid and faulty watchdog service conditions.

Fault detection in the proposed watchdog timer is achieved through multiple complementary mechanisms that ensure robust monitoring of software execution. Missed-service fault: This occurs when the system fails to service the watchdog within the valid monitoring cycle. As a result, the frame window is not reinitialized and eventually expires, leading to the assertion of the WDFAIL signal. This mechanism also enables the detection of processor failures or processor-clock failures that prevent the generation of valid watchdog service events.

Out-of-window service fault: This fault occurs when the watchdog is serviced either before the opening or after the closing of the valid service window. Any service operation performed outside the valid service window is immediately considered invalid and triggers a fault condition.

Double-service fault: After a valid watchdog service, the service window is considered satisfied for the current monitoring cycle. Any additional service request within the same monitoring cycle is treated as an invalid operation and results in a fault. Watchdog service detection is edge-sensitive and is evaluated using the rising edge of the service signal. Consequently, a service pulse is considered valid only if its rising edge occurs while the service window is active. A service event occurring after the service window has expired is classified as an out-of-window service fault rather than a double-service fault, thereby ensuring deterministic fault classification at timing boundaries.

Fault detection in the proposed design is achieved within a single SYSCLK cycle after the occurrence of out-of-window service and double-service faults. This ensures a deterministic and fast response, which is essential for real-time monitoring applications. In contrast, missed-service faults are detected when the frame window expires without a valid service event. Since the fault detection latency depends directly on the system clock period and configured timing windows, the watchdog provides predictable timing behavior suitable for safety-critical SCADA systems.

D. Fault Response and Control Logic

Once a fault condition is detected, the watchdog asserts the WDFAIL signal. This signal can be used to trigger safe-state operations, notify external supervisory logic, or initiate higher-level fault-management procedures. In SCADA environments, the WDFAIL signal supports partial recovery strategies, such as switching control functions to a backup Remote Terminal Unit (RTU) or activating redundant supervisory components before a complete system reset is performed.

After fault detection, a programmable delay is introduced before issuing a system reset. This delay allows the system to store diagnostic information, including the detected fault type recorded in the fault status register. The delay interval also provides sufficient time for external supervisory systems to process fault notifications and perform any required mitigation actions. Once the delay expires, the watchdog asserts the RSTOUT signal to reset the system. This two-stage response mechanism, based on fault detection followed by delayed reset, improves fault traceability and provides sufficient time for debugging, logging, and supervisory intervention. As a result, the proposed watchdog enhances both system safety and reliability in real-time SCADA-based applications while supporting integration with higher-level fault-tolerant control architectures.

III. MATERIALS AND METHODS

A. Watchdog Timer Implementation in FPGA

This section presents the implementation of the proposed configurable dual-window watchdog timer on an FPGA platform for real-time fault detection in SCADA-based energy systems. The overall architecture of the design is shown in Figure 4. The watchdog operates using a dedicated system clock (SYSCLK), independent of the processor clock, ensuring reliable monitoring even in the presence of processor stalls or software failures. The timing parameters of the watchdog are defined according to application requirements, particularly real-time industrial supervision constraints. The service window length (SWLEN) and frame window length (FWLEN) are configured through internal registers after system initialization. Once configured, these parameters are protected against unintended modification during runtime, thereby enhancing system robustness. Additional protection mechanisms, such as controlled reconfiguration procedures and configuration verification, may be incorporated to further strengthen configuration security in future implementations.

The watchdog timing windows are implemented using SYSCLK-driven counters, thereby avoiding additional clock domains and minimizing clock-domain crossing hazards. In practical deployments, watchdog service signals generated by the processor are sampled within the SYSCLK domain and processed using edge-detection logic. When asynchronous processor-generated service signals are present, a two-stage flip-flop synchronizer is employed before edge detection to reduce metastability risks. The service window is initiated by a transition on the INIT signal. Its duration is determined by the configurable parameter SWLEN, while the frame window duration is determined by FWLEN. These parameters can be selected according to the timing requirements of the supervised application. For example, assuming a SYSCLK frequency of 100 MHz, a service-window length of N_{SW} cycles corresponds to a service interval given by

$$T_{SW} = \frac{N_{SW}}{100 \text{ x } 10^6} \quad (1)$$

while the frame-window duration is given by

$$T_{FW} = \frac{N_{FW}}{100 \text{ x } 10^6} \quad (2)$$

where T_{SW} and T_{FW} denote the service-window and frame-window durations (s), respectively, while N_{SW} and N_{FW} represent the corresponding service-window and frame-window lengths expressed in SYSCLK cycles.

This allows the watchdog timing parameters to be mapped directly to real-time scan cycles commonly encountered in SCADA applications. The service window is implemented using timing counters that supervise the valid servicing interval. The timing mechanism accounts for alignment between the initialization event and the watchdog monitoring cycle. Conceptually, the effective service-window duration can be expressed as the sum of an initial alignment interval, the programmed service-window duration, and a final alignment interval. The effective service-window duration may therefore be approximated as

$$T_{eff} = T_{up} + T_{SW} + T_{down} \quad (3)$$

where T_{up} and T_{down} represent the initial and final alignment intervals, respectively, and T_{SW} denotes the programmed service-window duration.

This approach provides precise timing supervision while maintaining implementation simplicity. The status of the service window is continuously updated and made available through the configuration registers. Once a valid service occurs, the service window is considered satisfied

and the monitoring cycle continues under frame-window supervision. If a valid service occurs within the next monitoring cycle, the watchdog operation proceeds normally. Otherwise, the frame window expires and a fault condition is detected, leading to the assertion of the WDFAIL signal. This dual-window mechanism allows the detection of both early and late faults, improving fault coverage compared to conventional watchdog architectures. Figure 4 illustrates the finite state machine (FSM) controlling the initialization, monitoring, fault detection, and reset behavior of the proposed watchdog timer.

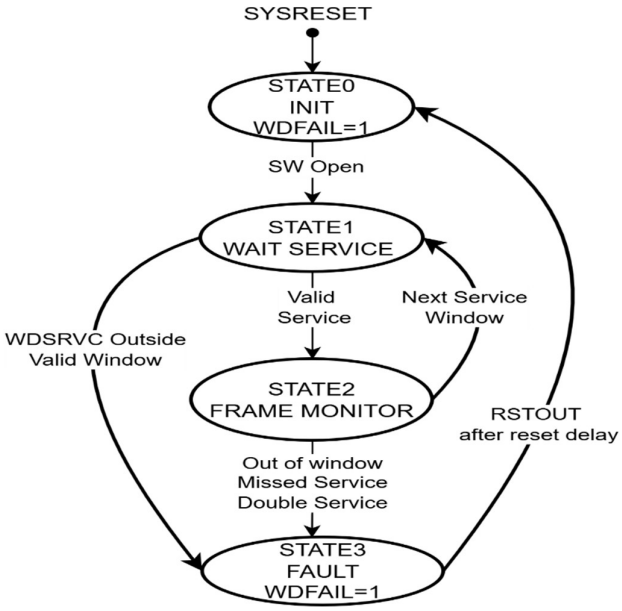

Figure 3 Finite state machine of the proposed watchdog fault detection logic.

B. Reset Initialization and Fault Detection

The initialization and fault detection behavior of the watchdog is controlled by a finite state machine (FSM), as illustrated in Figure 4. At power-up, the watchdog starts in an initialization state with the WDFAIL signal asserted. This ensures that the system begins operation in a safe and controlled condition, which is particularly important in SCADA-based energy systems. During this initialization phase, the reset delay mechanism remains inactive, preventing unintended reset generation before the watchdog has been properly initialized.

To initialize the watchdog, the reset control field (WDRST) must be activated. Once the service window opens, a valid service event clears the WDFAIL signal and transitions the system to normal operation. If the service is invalid, the initialization process is aborted and must be repeated. During normal operation, any violation of the expected timing behavior, such as missed-service, out-of-window service, or double-service faults, results in the assertion of the WDFAIL signal. The detected fault type is stored in the configuration register, enabling explicit fault classification and facilitating system diagnosis.

Following fault detection, a configurable reset-delay counter is triggered. The reset-delay interval is configured through the RSTLEN register. The supported delay range depends on the selected register value and system clock frequency, allowing the reset delay to be adapted to application-specific timing requirements. After the configured delay period, the watchdog asserts the RSTOUT signal, forcing a system reset. The delay interval can be adjusted according to application requirements and allows the system to capture diagnostic data or notify external supervisory components before recovery is enforced.

The reset counter remains inactive during the initial power-up phase, ensuring that no unintended reset is generated. Once the watchdog is properly initialized, the reset mechanism becomes fully operational, providing reliable fault recovery throughout system execution. The watchdog control logic is implemented using a finite state machine comprising initialization, monitoring, alarm, and reset-hold states. A binary state-encoding scheme is adopted to minimize register utilization while maintaining deterministic operation and efficient FPGA implementation. Figure 5 presents the functional block diagram of the proposed FPGA-based watchdog timer, including the main control, timing, and fault detection modules.

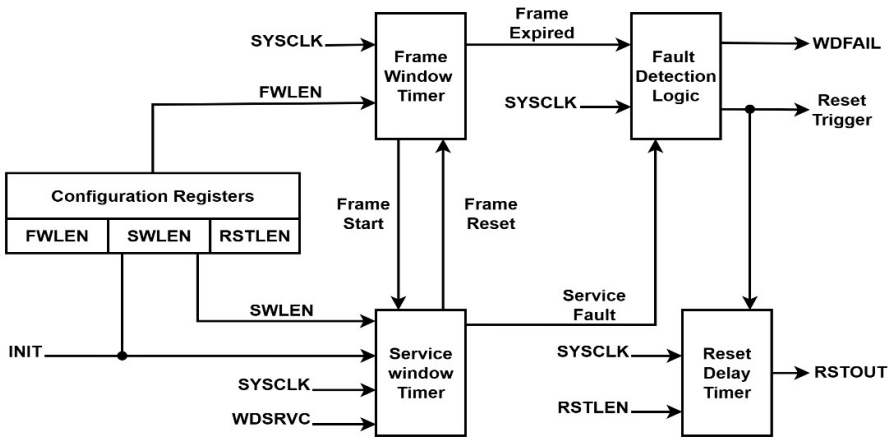

Figure 4 Functional block diagram of the proposed FPGA-based watchdog timer.

C. Integration of the Watchdog Timer into SCADA-Oriented Systems

To illustrate a potential industrial deployment scenario of the proposed design, the FPGA-based watchdog timer is integrated into a SCADA-oriented energy system architecture, as illustrated in Figure 6. The architecture shown in Figure 6 is intended as a conceptual integration framework demonstrating how the proposed watchdog can be deployed within a SCADA-based energy system.

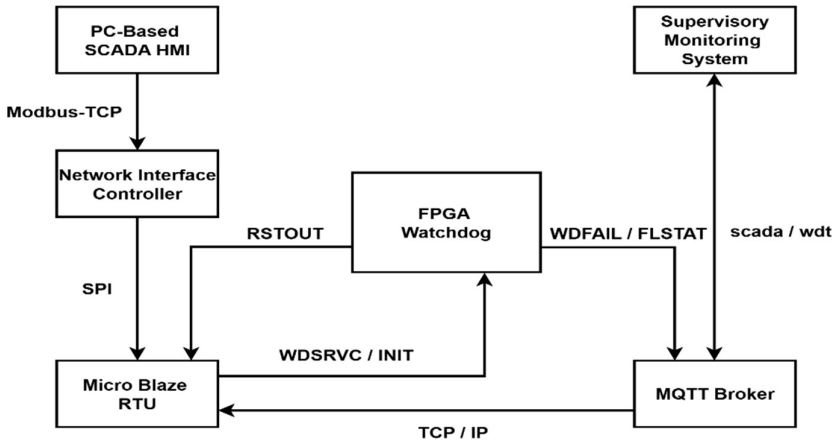

Figure 5 Integration of the proposed FPGA-based watchdog timer into a SCADA-oriented architecture.

In the proposed architecture, the watchdog timer is implemented as an independent supervisory hardware module within the FPGA platform. It interfaces directly with both the control layer and the communication infrastructure. A soft-core processor, such as MicroBlaze, can be used as a Remote Terminal Unit (RTU) controller, responsible for generating periodic service signals (WDSRVC and INIT) that reflect the execution of real-time control tasks. The watchdog supervises the timing of these service events independently of the processor software. Consequently, if a processor malfunction or processor-clock failure prevents the generation of valid service signals, the watchdog eventually detects a missed-service fault and initiates the corresponding fault response.

The SCADA Human–Machine Interface (HMI), running on a PC-based system, communicates with the FPGA through a Modbus-TCP protocol via a network interface. This communication allows operators to configure watchdog parameters, monitor system status, and interact with the embedded control logic. The use of Modbus-TCP ensures compatibility with existing industrial communication standards and legacy SCADA infrastructures (Memon and Kauhaniemi, 2021). Although Modbus-TCP is widely adopted in industrial environments, it is not a strictly deterministic real-time communication protocol. Nevertheless, watchdog timing supervision remains entirely hardware-based and independent of network latency or communication jitter.

In addition, fault information generated by the watchdog can be transmitted through an MQTT-based communication layer. Signals such as WDFAIL and fault classification data (FLSTAT) are transmitted through an MQTT broker acting as a centralized communication hub. Monitoring applications can subscribe to specific topics such as scada/wdt, enabling real-time visualization of alarms and system health information. This publish–subscribe mechanism supports scalable and distributed monitoring architectures. Since MQTT latency may vary

depending on network conditions, MQTT is used exclusively for alarm reporting, monitoring, and visualization purposes. The actual fault detection process and WDFAIL assertion remain fully deterministic and are performed directly in hardware.

The watchdog supervises RTU behavior by analyzing the timing of service signals. In the presence of abnormal conditions such as missed-service, early-service, or late-service events, the watchdog asserts the WDFAIL signal. This information can be propagated through the supervisory communication layer, allowing rapid detection and operator awareness. The proposed architecture is also compatible with RTOS-based industrial controllers, including environments such as FreeRTOS and VxWorks. By appropriately configuring the service-window and frame-window durations, expected task-scheduling jitter can be accommodated while preserving reliable fault detection.

In addition to fault reporting, the watchdog generates a reset signal (RSTOUT), which can be directly connected to the RTU controller. This enables automatic system recovery by forcing a reset when critical faults persist beyond a predefined delay. This two-level response, consisting of fault signaling followed by controlled reset, significantly improves system resilience and operational reliability.

IV. RESULTS AND DISCUSSION

Figures 7–10 present the simulation waveforms obtained from the proposed watchdog timer under normal and faulty operating conditions. The results illustrate the detection of missed-service, out-of-window service, and double-service faults together with the corresponding watchdog response signals.

A. Simulation and Fault Injection Results

1) Testbench Methodology

The proposed watchdog timer was verified using a dedicated Verilog HDL testbench developed in the Xilinx Vivado simulation environment. Normal operation was evaluated by generating watchdog service pulses within the valid service window. To validate the fault detection mechanisms, fault injection was performed by generating missed-service, out-of-window service, and double-service conditions. Additional boundary-condition tests were conducted to verify correct fault classification near service-window limits and during watchdog recovery. The resulting waveforms were analyzed to confirm the correct operation of the WDFAIL, RSTOUT, and fault classification signals. Table 1 summarizes the fault-injection scenarios considered during verification.

Table 1 Fault Injection Scenarios

S/N	Scenario	Expected Result
1	Normal operation	No fault detected
2	Missed-service fault	Fault code = 01
3	Out-of-window service fault	Fault code = 02
4	Double-service fault	Fault code = 03
5	Boundary-condition test	Deterministic fault classification
6	Recovery verification	Return to normal operation

2) Normal Operation

Under normal watchdog supervision, service events are generated within the valid service window and no fault condition is detected. Consequently, both watchdog response outputs remain inactive (WDFAIL = 0 and RSTOUT = 0), indicating correct watchdog operation. The waveform shown in Figure 7 confirms that valid service requests are accepted and that the monitoring cycle is continuously renewed without generating alarms. This behavior demonstrates proper operation of the service-window and frame-window supervision mechanisms under nominal conditions.

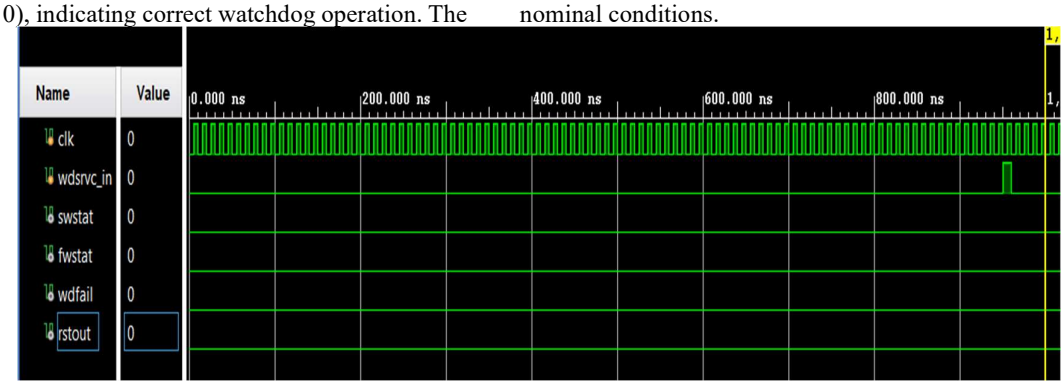

Figure 6 Simulation waveform during normal watchdog operation.

3) Missed-Service Fault

A missed-service fault occurs when the watchdog is not serviced before expiration of the frame window. Under this condition, the watchdog detects the absence of a valid service event and classifies the fault as a missed-service condition. As shown in Figure 8, the watchdog asserts the WDFAIL signal and records fault code 01. The waveform confirms correct detection of the fault and activation of the corresponding fault response mechanism. This result demonstrates the watchdog's ability to detect processor stalls, software hangs, and other conditions that prevent timely servicing of the watchdog.

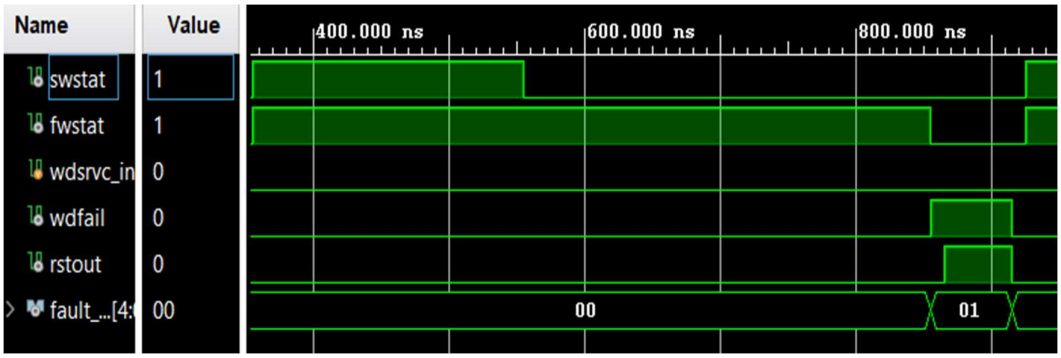

Figure 7 Simulation waveform showing missed-service fault detection.

4) Out-of-Window Service Fault

An out-of-window service fault occurs when the watchdog is serviced outside the valid service interval. In this scenario, the service request is rejected and immediately classified as an invalid operation. Figure 9 shows that the watchdog correctly identifies the fault and generates fault code 02 while asserting the WDFAIL signal. The result confirms that the proposed dual-window watchdog can detect timing violations associated with premature or delayed service events, thereby improving fault coverage compared with conventional watchdog implementations.

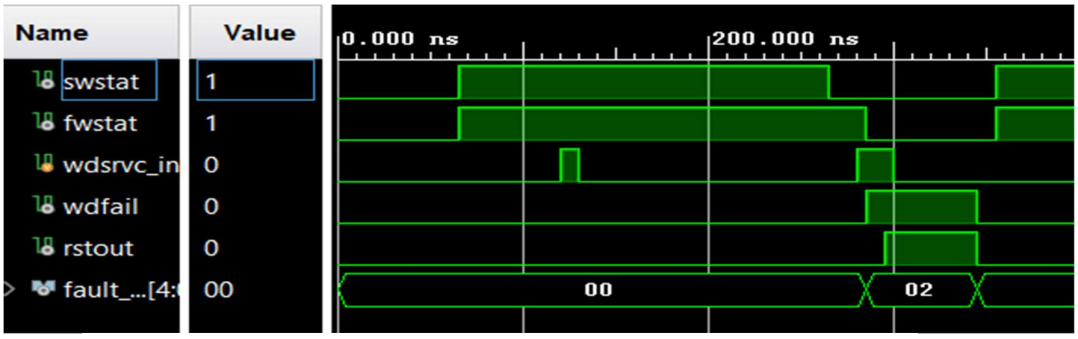

Figure 8 Simulation waveform showing out-of-window service fault detection.

5) Double-Service Fault

A double-service fault occurs when more than one watchdog service pulse is issued within the same monitoring cycle after a valid service has already been accepted. Under this condition, the watchdog interprets the additional service event as abnormal behavior and generates a fault indication. As illustrated in Figure 10, the fault is classified using fault code 03 and the WDFAIL signal is asserted. The waveform confirms that the proposed watchdog successfully detects repeated service attempts and distinguishes them from other fault conditions through explicit fault classification.

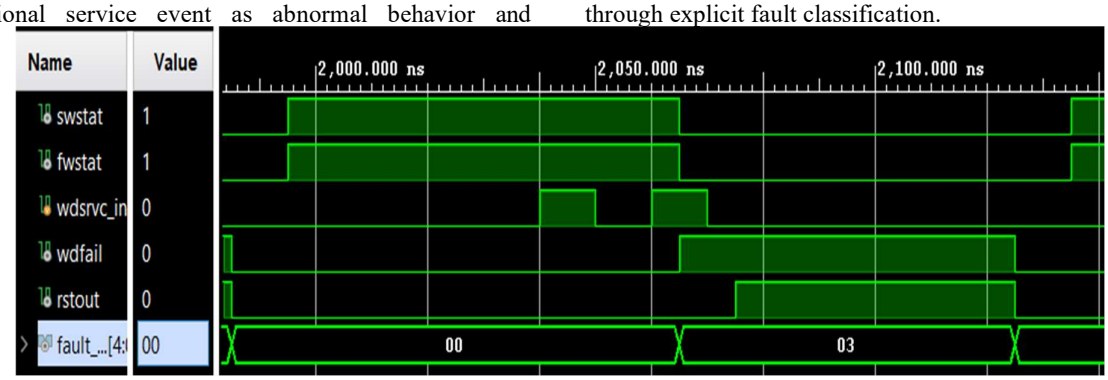

Figure 9 Simulation waveform showing double-service fault detection.

The simulation results confirm that the proposed watchdog correctly detects and classifies all considered fault conditions. Distinct fault codes are assigned to missed-service, out-of-window service, and double-service faults, enabling explicit fault identification. Out-of-window service and double-service faults are detected within one SYSCLK cycle after the occurrence of the invalid event, whereas missed-service faults are detected upon frame-window expiration. This behavior ensures deterministic fault supervision while maintaining compatibility with configurable monitoring intervals. Table 2 summarizes the implemented fault classification scheme and watchdog response.

Table 2 Fault Code Classification and Watchdog Response

S/N	Fault Code	Fault Type	WDFAIL	RSTOUT
1	00	No fault	0	0
2	01	Missed-service fault	1	Asserted after reset delay
3	02	Out-of-window service fault	1	Asserted after reset delay
4	03	Double-service fault	1	Asserted after reset delay

6) Experimental FPGA Validation

To complement the simulation-based verification, the proposed dual-window watchdog timer was implemented and validated on a Xilinx Artix-7 Basys3 FPGA platform. Figure 11 presents the oscilloscope capture obtained during hardware testing. The synthesized design was successfully programmed onto the FPGA using the Vivado Hardware Manager and evaluated under real operating conditions. The oscilloscope measurements confirmed the correct generation and sequencing of the WDSRVC, WDFAIL, and RSTOUT signals during fault conditions. The observed hardware behavior was consistent with the simulation results, demonstrating correct watchdog servicing, fault detection, and recovery operation. In particular, the watchdog successfully detected fault events, asserted the WDFAIL signal, and generated the RSTOUT signal after the configured recovery interval. These experimental results validate the practical implementation of the proposed architecture and confirm its suitability for real-time fault detection and recovery in FPGA-based SCADA energy systems.

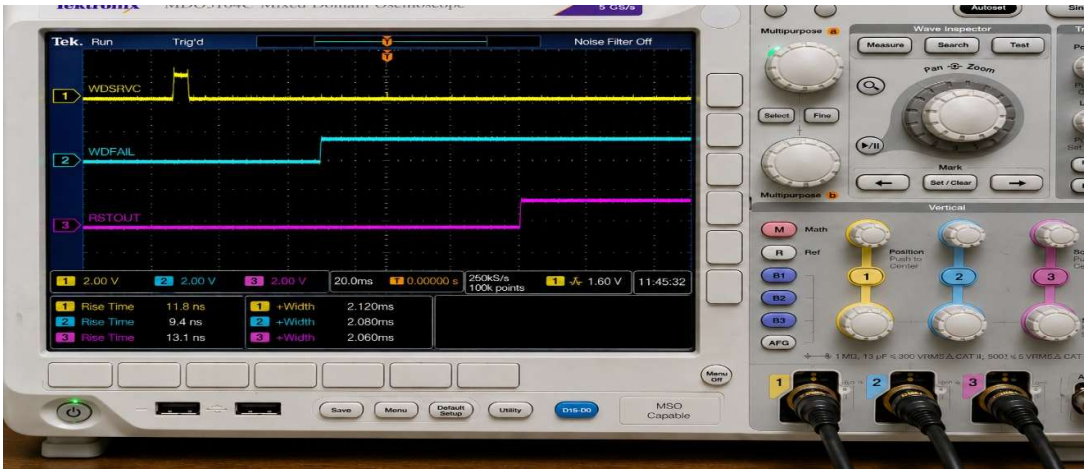

Figure 11 Experimental FPGA validation showing WDSRVC, WDFAIL, and RSTOUT signals captured using a digital oscilloscope.

B. FPGA Resource Utilization

Table 3 summarizes the FPGA resource utilization of the proposed watchdog implementation on the Xilinx Artix-7 device. The results show that the design requires only 198 LUTs and 107 registers, corresponding to less than 1% of the available device resources. The utilized resources are mainly associated with timing counters, control logic, and fault detection circuitry. Most of the utilized LUTs and registers are associated with the service-window and frame-window counters, finite-state-machine control logic, configuration registers, and fault-classification circuitry. No memory-intensive or arithmetic-intensive modules were required. In addition, synthesis reports confirmed that no Block RAM (BRAM) or DSP resources were inferred. The low resource utilization demonstrates the lightweight nature of the

proposed watchdog and facilitates its integration into larger FPGA-based SCADA systems.

Table 3 FPGA Resource Utilization Summary

Device Utilization Summary				
S/N	**Logic Utilization**	**Used**	**Available**	**Utilization**
1	Slice LUTs	198	20800	0.95%
2	Slice Registers	107	41600	0.26%
3	Block RAM	0	50	0%
4	DSP Blocks	0	90	0%

C. Timing Analysis

Table 4 summarizes the timing performance of the proposed watchdog implementation. The design achieved a maximum operating frequency of 149.992 MHz with a worst negative slack (WNS) of 0.962 ns and no failing endpoints. Timing constraints were specified using a Vivado XDC constraint file, including create_clock definitions and the required timing constraints for implementation on the target FPGA device. The positive timing margin confirms that the design satisfies all timing requirements. Given the very low resource utilization, the proposed watchdog can be integrated into larger FPGA-based SCADA systems while maintaining sufficient timing headroom, although timing verification should be repeated after full system integration.

Table 4 Timing Analysis Summary

S/N	**Parameter**	**Value**
1	Frequency	149.992 MHz
2	Clock period	6.667 ns
3	WNS	0.962 ns
4	Worst path delay	5.421 ns
5	TNS	0.000 ns
6	Failing endpoints	0

D. Reliability and Scalability Analysis

To further assess the suitability of the proposed watchdog for safety-critical SCADA applications, a simplified Failure Modes and Effects Analysis (FMEA) and a scalability assessment were conducted. The FMEA identifies representative failure scenarios and the corresponding watchdog response mechanisms, while the scalability analysis evaluates the feasibility of deploying multiple watchdog instances within the same FPGA device. Table 5 summarizes the simplified FMEA of the proposed watchdog, including representative failure modes and their associated detection mechanisms.

Table 5 Simplified FMEA of the Proposed Watchdog

S/N	**Failure Mode**	**Effect**	**Detection Mechanism**
1	Missed watchdog service	Loss of task supervision	Missed-service fault detection
2	Out-of-window service	Timing violation	Out-of-window fault detection
3	Double service	Abnormal execution behavior	Double-service fault detection
4	Processor clock failure	No watchdog service generated	Missed-service fault detection
5	Software hang	Loss of periodic servicing	Missed-service fault detection
6	Invalid service sequence	Faulty watchdog operation	Fault classification logic

The low FPGA resource utilization enables replication of the watchdog module to supervise multiple RTUs or independent software tasks. Assuming approximately linear resource growth, multiple watchdog

instances can be implemented while maintaining acceptable overall FPGA utilization. Table 6 presents the estimated resource requirements for multiple watchdog instances based on the measured utilization of a single implementation.

Table 6 Estimated Resource Scaling for Multiple Watchdog Instances

S/N	Number of Instances	Estimated LUTs	Estimated Registers
1	1	198	107
2	10	1,980	1,070
3	50	9,900	5,350
4	100	19,800	10,700

The scalability results indicate that dozens of watchdog instances can be implemented on the target Artix-7 FPGA while remaining within available device resources. This capability is particularly attractive for SCADA **architectures** containing multiple RTUs or independently supervised control processes. The 100-instance configuration represents a theoretical upper-bound estimate based on linear resource extrapolation and would require additional timing and implementation verification following full system integration. Nevertheless, the results demonstrate that the proposed architecture can support large-scale deployment scenarios while preserving the lightweight characteristics of the watchdog design.

E. Comparative Discussion

Table 7 compares the proposed watchdog timer with representative fault-detection and monitoring approaches reported in the literature. The comparison highlights the capability of the proposed design to provide deterministic hardware supervision, configurable watchdog windows, autonomous reset generation, and explicit fault classification. For clarity, ✓ indicates full support of a feature, ✗ indicates that the feature is not supported, and Partial indicates limited or indirect support.

Table 7 Comparison of Existing and Proposed SCADA-Based Fault Detection and Monitoring Systems

S/N	Features	(Mufti et al., 2017)	(Islam et al., 2025)	Proposed Work
1	SCADA integration	✓	✓	✓
2	Real-time monitoring	✓	✓	✓
3	FPGA-based implementation	✗	✗	✓
4	Autonomous reset generation	Partial	Partial	✓
5	Explicit fault classification	✗	Partial	✓
6	Deterministic hardware supervision	✗	✗	✓
7	Configurable watchdog windows	✗	✗	✓

Source: Compiled from (Mufti et al., 2017), (Islam et al., 2025), and the proposed work.

In addition to research-oriented watchdog solutions, the proposed design was qualitatively compared with representative industrial watchdog devices. The comparison focuses on configurability, fault classification capability, and integration flexibility. Table 8 compares the proposed watchdog with representative commercial watchdog devices in terms of configurability, fault classification, and integration flexibility.

Table 8 Comparison with Representative Commercial Watchdog Devices

S/N	Feature	MAX6369	TPS3850	Proposed Work
1	Hardware watchdog	✓	✓	✓
2	FPGA implementation	✗	✗	✓
3	Configurable service window	Limited	Limited	✓
4	Fault classification	✗	✗	✓
5	Autonomous reset generation	✓	✓	✓
6	Multi-instance deployment	✗	✗	✓
7	SCADA-oriented integration	✗	✗	✓

Source: Compiled from MAX6369 and TPS3850 datasheets and the proposed work.

The results demonstrate that the proposed FPGA-based watchdog combines the flexibility of programmable hardware with deterministic fault supervision and explicit fault classification. Compared with existing research and commercial watchdog solutions, the proposed design provides enhanced configurability, support for multiple fault types, and seamless integration into FPGA-based SCADA environments. The main limitation of the current study is that validation was performed on a laboratory-scale FPGA platform rather than within a live SCADA installation. Future work will focus on deployment in operational SCADA environments, integration with industrial controllers, and evaluation under realistic field conditions. Nevertheless, the obtained results indicate that the proposed architecture constitutes a lightweight, efficient, and effective supervision mechanism for real-time industrial control systems.

V. CONCLUSION

This work presented a configurable FPGA-based dual-window watchdog timer for real-time fault detection in SCADA-based energy systems. The proposed architecture combines configurable service and frame windows with dedicated fault detection logic to provide deterministic supervision of system timing behavior. The watchdog successfully detected and classified missed-service, out-of-window service, and double-service faults while providing a two-stage fault response based on WDFAIL assertion followed by delayed system reset. Simulation, FPGA implementation, and experimental oscilloscope measurements confirmed the correct operation of the proposed design under both normal and faulty operating conditions. The implementation achieved low FPGA resource utilization, no BRAM or DSP usage, and stable timing performance, demonstrating its suitability for real-time industrial applications. In addition to automatic system recovery, the generated fault indications can support higher-level supervisory actions such as alarm generation, system isolation, or switchover to backup control units. Overall, the results demonstrate that the proposed watchdog provides an efficient, lightweight, and reliable hardware-based supervision mechanism for FPGA-based SCADA systems. Future work will focus on deployment in live SCADA environments, integration with industrial controllers, and the development of enhanced security and fault-recovery mechanisms.

AUTHOR CONTRIBUTIONS

A. Soke: Conceptualization, methodology, FPGA design and implementation, simulation, validation, data analysis, writing-original draft preparation. **L. C. Ngugi**: Supervision, methodology review, technical guidance, writing-review and editing. **J. Adebisi**: Co-supervision, technical review, validation of results, writing-review and editing.

ACKNOWLEDGMENTS

The authors would like to acknowledge the Pan African University Institute for Basic Sciences, Technology and Innovation (PAUSTI) for providing the academic environment and facilities that supported this research.

FUNDING

The work was unfunded.

CONFLICTS OF INTEREST

The authors declare that there is no conflict of interest regarding this article.

REFERENCES

Ahmed and Shoaib (2023) 'FPGA-Based Implementation of SCADA System for Fuel Management', Quaid-e-Awam University Research Journal of Engineering, Science & Technology, 21(2), pp. 21–28. https://doi.org/10.52584/qrj.2102.03

Alkady G.I., Ramez M. D., Hassanein H. A., Yves S. and Hani F.R. (2025) 'Dual-Level Fault-Tolerant FPGA-Based Flexible Manufacturing System', Designs, 9(3). https://doi.org/10.3390/designs9030056

B. B. Manjula, N. Santhosh, K. S. Ravikiran, K. Pooja and H. V. Sahana (2021) 'Fault Detection Mechanism using improved watchdog', IJRESM, 4(7), pp. 204–207. https://journal.ijresm.com/index.php/ijresm/article/view/1034

Bafna, A. and Verma, S. (2025) 'International Journal of Research Publication and Reviews FPGA Based Fault Recovery Architecture of Watchdog Timer', International Journal of Research Publication and Reviews, 6(9), pp. 1595–1604. https://ijrpr.com/uploads/V6ISSUE9/IJRPR52676.pdf

Breier, J. and Hou, X. (2022) 'How Practical Are Fault Injection Attacks, Really?', IEEE Access, 10, pp.

113122–113130. https://doi.org/10.1109/ACCESS.2022.3217212.

Devi, V.R. and Sreedhar, J. (2023) 'Design and Implementation of an Improved Watchdog Timer for Memory Applications', 2023 Global Conference on Information Technologies and Communications, GCITC 2023. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/GCITC60406.2023.10426468.

Islam, N. Ahmad S., Mahbub-E-Elahi P., Shoaib M, Chowdhury K., Mizan S.,Ahmed A., Jahan E., Hazari M. and Hossain C. (2025) 'An Intelligent SCADA System for Power Distribution Network Cable Fault Detection with Real-Time Monitoring and Autonomous Maintenance'. 2025 4th International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST): https://doi.org/10.1109/ICREST63960.2025.10914481.

Khairullah, S.S. and Elks, C.R. (2020) 'Self-repairing hardware architecture for safety-critical cyber-physical-systems', IET Cyber-Physical Systems: Theory and Applications, 5(1), pp. 92–99: https://doi.org/10.1049/iet-cps.2019.0022.

Kumar Misra, N. and Swetha, T. (2022) 'Design and Synthesis of Windowed Watchdog Timer for High Speed Memory Applications Literature Review', Acta Scientific Computer Sciences, 4(6), pp. 54–58: https://www.actascientific.com/ASCS/pdf/ASCS-04-0282.

Maatallah, N., Mestiri H., Mohammed A. and Machhout M. (2025) 'Enhancing IoT Security for Sustainable Development: A Parity Checking Approach for Fault Detection in PRESENT Block Cipher', Engineering, Technology and Applied Science Research, 15(2), pp. 21982–21988: https://doi.org/10.48084/etasr.10109.

Manjudevi, and Prakruthi. P (2021) 'FPGA Implementation of an Improved Watchdog Timer for Safety Critical Applications', International Journal of Advances in Engineering and Management (IJAEM), 3, p. 190: https://doi.org/10.35629/5252-0311190194.

Mária Pohronská and Tibor Krajčovič (2011) 'FPGA Implementation of Multiple Hardware Watchdog Timers for Enhancing Real-Time Systems Security', Proceedings of EUROCON 2011 - International Conference on Computer as a Tool. Lisbon, Portugal: IEEE: https://doi.org/10.1109/EUROCON.2011.5929215.

Memon, A.A. and Kauhaniemi, K. (2021) 'Real-Time Hardware-in-the-Loop Testing of IEC 61850 GOOSE-Based Logically Selective Adaptive Protection of AC Microgrid', IEEE Access. Institute of Electrical and Electronics Engineers Inc., pp. 154612–154639: https://doi.org/10.1109/ACCESS.2021.3128370.

Mestiri, H., Barraj, I. and Machhout, M. (2025) 'Innovative Fault Detection for AES in Embedded Systems: Advancing Resilient and Sustainable Digital Security', Engineering, Technology and Applied Science Research, 15(2), pp. 20660–20667: https://doi.org/10.48084/etasr.9852.

Mufti H., Tanoli A., Waqar M., Rashid A., Durrani F., and Durrani S. (2017) 'Design and Implementation of Smart Fault Detection System for Industrial Power House using PLC and SCADA', IJSRST, 3(2), pp. 105–111: https://www.academia.edu/111105167/Design_and_Implementation_of_Smart_Fault_Detection_System_for_Industrial_Power_House_using_PLC_and_SCADA.

Nikola Zlatanov (2014) 'Architecture and Operation of a Watchdog Timer', Embedded Systems Conference. San Jose, CA, USA. https://doi.org/10.13140/RG.2.1.1149.1605.

Pliatsios D., Sarigiannidis P., Lagkas T., and Sarigiannidis A. (2020) 'A Survey on SCADA Systems: Secure Protocols, Incidents, Threats and Tactics', IEEE Communications Surveys and Tutorials, 22(3), pp. 1942–1976. Available at: https://doi.org/10.1109/COMST.2020.2987688.

Mohammadreza S. A., Angizi, S. and Violante M. (2024) 'Dependability in Embedded Systems: A Survey of Fault Tolerance Methods and Software-Based Mitigation Techniques', IEEE Access, 12, pp. 180939–180967: https://doi.org/10.1109/ACCESS.2024.3509633.

Supraja D., Kumar M., and Prasad S.V.S. (2025) 'FPGA Implementation Of An Improved Watchdog Timer For Safety-Critical Applications', Proceedings of International Conference on Computer Science and Communication Engineering (ICCSCE 2025). Atlantis Press, pp. 2421–2434. https://doi.org/10.2991/978-94-6463-858-5_202.

Yipeng Zhang, Ye Li and Zhoujun Li (2023) 'Aye: A Trusted Forensic Method for Firmware Tampering Attacks', Symmetry, 15(1). https://doi.org/10.3390/sym15010145.