FPGA IMPLEMENTATION OF AN IMPROVED WATCHDOG TIMER FOR SAFETY – CRITICAL APPLICATIONS

[1]Imran Shaik, [2]Dr. V. Manohar

[1] M.Tech -VLSI, Department of ECE, Vaagdevi engineering College, Bollikunta, Khilaa Warangal (M), Warangal urban, Telangana.-506005

imrans4b0@gmail.com

[2] Associate Professor, Department of ECE, Vaagdevi engineering College, Bollikunta, Khilaa Warangal (M), Warangal urban, Telangana.-506005

ABSTRACT

Embedded systems used in safety-critical applications demand the highest level of reliability. External monitor timers are used for the automated handling and recovery of errors related to running time in these systems. Many of the external watchdog timers available use additional circuits to change their timeouts and provide only minimal functionality features. This paper describes the architecture and design of an improved configurable watchdog timer which can be used in critical security applications. Many mechanisms for detecting faults are integrated into the firewall, contributing to its robustness. The features and operations are very common and can be used to track the operations of any real-time processor-based device. This paper also addresses the proposed watchdog timer in a Field Programmable Gate Array (FPGA) implementation. This makes fast adaptation of the design to various applications, while reducing overall device costs. First, by analyzing the simulation model, the implementation of the proposed watchdog timer to detect and respond to faults is studied. The design is validated in real- time hardware by injecting faults through the software during the execution of the processor and drawing conclusions.

Keywords: Watchdog Timer, FPGA Implementation, Safety-Critical Systems, Embedded Systems, Fault Detection, Real-Time Monitoring, Configurable Architecture, Hardware Validation.

I.INTRODUCTION

Applications where failures can lead to severe damage or human injury require the highest degree of system reliability. Such safety-critical systems must be capable of detecting faults and recovering from unexpected failures without human intervention to maintain continuous and safe operation. Fault-tolerant mechanisms play a vital role in identifying errors, initiating corrective actions, and minimizing downtime to ensure overall system resiliency [1].

One of the commonly adopted approaches to fault tolerance is hardware redundancy, where multiple replicas of critical system components are used to improve system reliability [2]. However, this approach often increases hardware/software complexity and overall system costs. A more cost-effective technique for detecting runtime failures is the Watchdog Timer (WDT) [3].

A WDT is a hardware monitoring unit responsible for continuously supervising processor functionality and taking predefined actions when anomalies occur [4]. It mainly consists of a timer that the processor must periodically reset. Failure to do so indicates that the system has become unresponsive or stuck in an erroneous state [5], prompting the WDT to reset the system or shift it to a safe operational mode to prevent further damage.

WDTs can be implemented internally (on-chip) or externally. However, internal watchdogs contribute to increased hardware overhead, can be disabled by faulty software during runtime, and depend on the processor clock, making them unreliable if the system clock fails [3][6].

External watchdogs operate independently of the processor, making them a more robust solution for safety-critical environments [7]. Yet, commercially available external WDTs typically have fixed timeout periods or require additional external components for configuration, increasing both design complexity and cost.

These challenges can be effectively addressed by integrating the watchdog functionality within a Field Programmable Gate Array (FPGA). Modern embedded systems frequently utilize FPGAs due to their flexibility, reconfigurability, and capability to host multiple system modules on a single chip [8]. Implementing the WDT within FPGA architecture not only enhances reliability but also reduces cost and simplifies integration.

Previous research explored FPGA-based watchdog solutions, such as custom real-time monitoring systems [9] and sequenced watchdog timers [10]. However, these designs provided limited configurability and minimal fault detection features. A basic multi-hardware WDT structure for FPGA was presented in [11], but without advanced architectural enhancements.

To overcome these limitations, this paper proposes an improved Window Watchdog Timer (WWDT) implemented on FPGA with enhanced configurability, robust fault-detection mechanisms, and reusable IP-core design. This approach makes the system highly suitable for safety-critical applications with redundancy requirements, such as avionics, automotive safety, and defense systems.

Furthermore, integrating the WDT as a reusable IP core mitigates component obsolescence issues often encountered in long-life embedded systems like aerospace and military hardware [12].

II.LITERATURE SURVEY

2.1. The Design of a Fault-Tolerant COTS-Based Bus Architecture

This work presents the development of a fault-tolerant avionics bus system using Commercial Off-The-Shelf (COTS) technologies for NASA's Jet Propulsion Laboratory X2000 project. The system integrates two COTS buses — IEEE 1394 and I²C — while strictly maintaining commercial standards to preserve cost advantages. Hardware and software prototypes of the fault-tolerant bus have been validated and deployed in deep-space missions, proving that highly dependable networks can be efficiently built using low-cost COTS components.

2.2. Fault-Tolerant Platforms for Automotive Safety-Critical Systems

In the automotive sector, fault-tolerant subsystems are now essential due to increasing electronic integration in vehicles. This paper analyzes chip-level fault-tolerant architectures inspired by solutions in aerospace engineering. The authors propose a highly cost-effective dual-unison platform that operates as either a fail-operational or fail-silent system, optimizing performance, flexibility, and safety for evolving vehicular applications.

2.3. A Review of Watchdog Architectures and Their Application to CubeSats

To support low-cost and rapid-development spacecraft missions, CubeSats extensively utilize COTS hardware. While economically beneficial, such designs require reliable protection against harsh radiation and space-borne disturbances. This review emphasizes the importance of watchdog-based fault-tolerance in CubeSat onboard computers to maintain operational integrity in extreme conditions.

2.4. Concurrent Error Detection Using Watchdog Processors — A Survey

This survey investigates parallel error-detection mechanisms using watchdog processors — compact coprocessors that observe system behavior to detect anomalies. Compared to full hardware redundancy, watchdog-based detection requires significantly less hardware while still identifying data-flow and memory operation-level errors. Several flow-checking techniques and capability-based memory validation schemes are evaluated to improve processor reliability.

2.5. FPGA Adoption for High-Performance Embedded Applications

This study highlights the power of FPGAs in high-bandwidth digital communication applications such as Software-Defined Radios (SDR) for space missions. FPGAs efficiently execute computationally intensive tasks like filtering, decoding, and modulation using massive parallelism, leading to high throughput and low power. The integration of embedded processing elements further enhances SDR flexibility for future aerospace and defense communication systems.

2.6. FPGA Implementation of an Improved Watchdog Timer for Safety-Critical Applications

This referenced work proposes a configurable and externally operated watchdog timer implemented on FPGA for safety-critical embedded systems. The design includes multiple fault-detection mechanisms for system monitoring and recovery. Simulation-based validation confirms the timer's ability to detect and respond to faults, followed by successful hardware tests with intentional fault injection to ensure robustness.

2.7. Comparison of Internal and External Watchdog Timers

This application note compares microcontroller-integrated internal watchdogs and standalone external watchdog hardware. Internal watchdogs are easy to deploy but vulnerable to software failures and processor clock issues. In contrast, external watchdogs offer higher reliability and precision at the cost of additional board space. A comparative analysis highlights the strengths and limitations of each approach for safety-critical designs.

III.EXISTING SYSTEM

In existing safety-critical embedded systems, both internal and external watchdog timers are used to monitor the processor and prevent system crashes due to software malfunctions or execution failures. Internal watchdog timers are integrated within the microcontroller and rely on the same system clock and software execution, making them vulnerable to runaway code, clock failures, and accidental deactivation during runtime. Although they are low-cost and easy to implement, their reliability is insufficient for high-risk environments. External watchdog timers overcome these limitations by operating independently from the processor and providing more precise and robust fault coverage. However, most commercially available external watchdog solutions have fixed timeout values and limited configurability, requiring additional circuitry for customization. This increases hardware complexity, cost, and design effort. Furthermore, many existing watchdog systems only support basic timeout-based reset functionality and lack advanced fault detection mechanisms to identify different categories of system failures. As the demand for intelligent, fault-tolerant, and reconfigurable safety systems grows, current watchdog implementations are unable to fully satisfy the reliability requirements of modern aerospace, automotive, and industrial applications.

IV. PROPOSED SYSTEM

A successful controller should be able to identify all suspicious modes of software and return the device to a known condition. It will have its own clock and ought to be able to provide all peripherals with a hardware reset on timeout [3]. The watch-dog timer presented in this study acts independently of the operating system and uses for its operations a dedicated clock. The architecture follows a fenced-in Watchdog development where software preprocessing can customize window periods. Because when watchdog timer expires, a fail-flag is lifted, and a reset is activated after a specified period of time from waving the flag. The period between them may be used by app to store useful testing knowledge on a private internet. A typical control signal can identify errors in programs such as hanging due to infinite loops in code execution. However, the key downside to this watchdog is that if the program reaches a failure condition in which it constantly resets the timer, it could never notice the error condition. In other terms, a typical monitor timer can diagnose slow defects but it cannot identify fast flaws. [13]. This can therefore be done adequately by a windowed design. Here, the watchdog determines a limited time period during which to update the safeguard to prevent a timeout. This offers machine security from operating too quickly and too slow [14], thus growing the scope of error recognition. A. I / O Gui and Figure Setup. 1 Depicts the pro-posited watch input-output (I / O) interface. The watchdog has two outputs; the output of the watchdog fails (WDFAIL) and the output of the reset (RSTOUT). The WDFAIL output remains asserted when the SYSRESET input is low and the RSTOUT output stays de-asserted. The specification also consists of a configuration register described as in the figure, with bit fields. The list enables modification of the conditions of the monitor and also includes status details. The fields WDRST and WDSRVC are used to initialize and control the inspector, respectively. Throughout the configuration file, the status of the INIT input and the WDFAIL output are immediately changed. The SWSTAT field holds the service window status, and if any, the FLSTAT field logs the watchdog mode of failure.

V.SYSTEM ARCHITECTURE

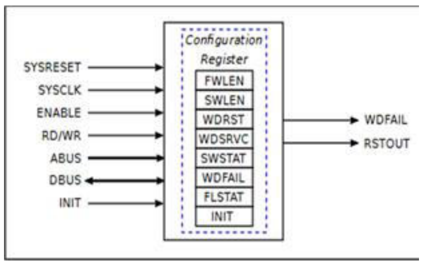


Fig 5.1 System Architecture

The architecture of the proposed FPGA-based watchdog timer is designed with a configurable register block that enables flexible control and monitoring of the system's operational status. Various input signals such as SYSRESET, SYSCLK, ENABLE, RD/WR, ABUS, DBUS, and INIT interface with the configuration registers to manage watchdog settings and communication. The system clock (SYSCLK) drives the timing operations, while SYSRESET and INIT are responsible for initializing or resetting the watchdog operation when required. The RD/WR, ABUS, and DBUS collectively support read/write operations between the processor and watchdog configuration registers, allowing real-time modification of the watchdog parameters.The configuration register contains multiple fields such as FWLEN (Frame Window Length), SWLEN (Service Window Length), WDRST (Watchdog Reset), WDSRVC (Service Status), SWSTAT (Service Window Status), WDFAIL (Watchdog Failure Indication), FLSTAT (Failure Status), and INIT (Initialization Control). These registers provide full control over timing windows, operational modes, and system fault status. The watchdog continuously observes processor responses within specified time constraints. When any anomaly occurs—like delayed servicing or processor hang—status signals such as WDFAIL and RSTOUT are triggered to either log the fault or reset the system. Thus, the architecture ensures fault tolerance, improved reliability, and autonomous system recovery suitable for safety-critical embedded applications.Windowed inspector style is a service window and a screen window. Compared to the frame window the operation period length would be significantly less. After power-up, the device will program the duration of the two windows by writing to the Bit fields SWLEN and FWLEN, In Log of Setup.

VI.IMPLEMENTATION

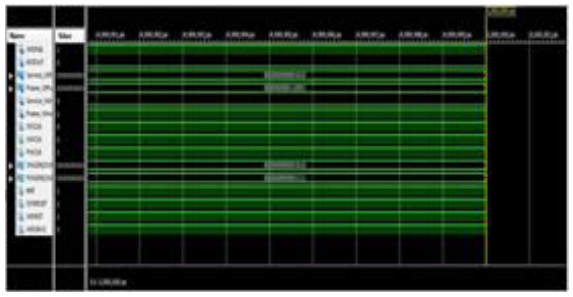


Fig 6.1 SYNTHESIS RESULT

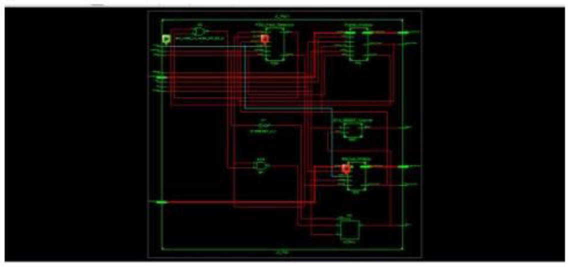


Fig 6.2 RTL SCHEMATIC

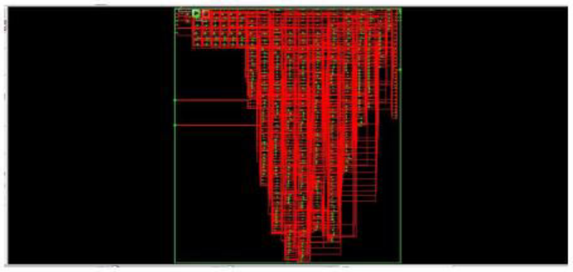
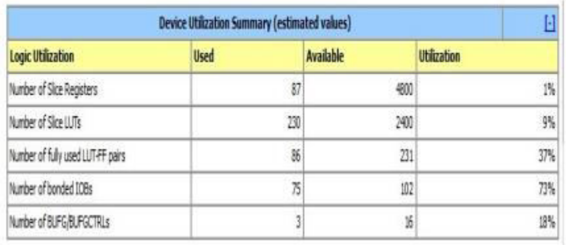
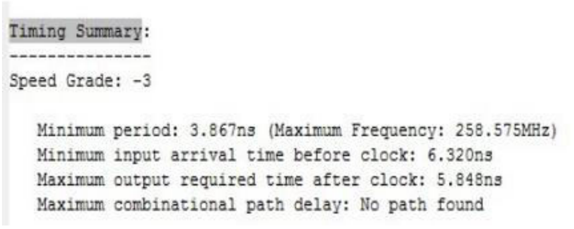


Fig 6.3 TECHNOLOGY SCHEMATIC



Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	87	4800	1%
Number of Slice LUTs	230	2400	9%
Number of fully used LUT-FF pairs	86	231	37%
Number of bonded IOBs	75	102	73%
Number of BUFG/BUFGCTRLs	3	16	18%

Fig 6.4 DESIGN SUMMARY



```
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: 3.867ns (Maximum Frequency: 258.575MHz)
   Minimum input arrival time before clock: 6.320ns
   Maximum output required time after clock: 5.848ns
   Maximum combinational path delay: No path found
```

Fig 6.5 TIMING REPORT

VII.CONCLUSION

The proposed system introduces an enhanced Window Watchdog Timer (WWDT) architecture implemented on FPGA to ensure reliable operation in safety-critical embedded applications. Unlike traditional internal watchdogs that rely on the processor clock, the improved watchdog functions independently, thereby eliminating the risk of failures due to clock malfunction or software runaway conditions. The configurable design allows the system to dynamically adjust timing parameters such as the frame window and service window based on application requirements, offering high flexibility and adaptability.To achieve strong fault tolerance, multiple fault-detection mechanisms are integrated, enabling early identification of abnormal processor behavior such as late servicing, early servicing, or missed servicing events. The watchdog is capable of classifying different fault types and logging detailed error status, which greatly aids in fault diagnosis and system recovery analysis. When a fault is detected, the WWDT generates reset or fault indication signals, ensuring that the system enters a safe state and resumes operation without human intervention.Additionally, the architecture provides a grace period before system reset, allowing the processor to preserve critical failure data and system logs. This supports efficient troubleshooting and rapid restoration. The FPGA-based implementation enables high-speed parallel processing, low-latency response, and simplified integration into a wide range of heterogeneous designs. The watchdog IP core is reusable, scalable for multicore systems, and addresses long-term support and obsolescence challenges common in aerospace, automotive, defense, and industrial safety systems. Overall, the enhanced WWDT delivers robust protection against runtime failures, significantly improving the reliability, safety, and maintainability of real-time embedded systems.

VIII.FUTURE SCOPE

The enhanced window watchdog timer designed in this work offers strong protection for safety-critical systems; however, several improvements can be incorporated in future developments. Advanced fault-classification algorithms can be integrated to not only detect but also accurately predict failures before they occur, supporting predictive maintenance strategies. The watchdog can be expanded to support network-based monitoring, enabling remote supervision and system health tracking in distributed embedded environments such as IoT, autonomous vehicles, and industrial automation.

Further improvements may include the addition of self-diagnostic capabilities and adaptive timing control based on real-time workload patterns, enhancing flexibility and efficiency in dynamic system conditions. The design can be extended into a fully compliant hardware security module to detect malicious activities or cyberattacks attempting to disable safety mechanisms. Support for radiation-hardened FPGA devices can also make the watchdog suitable for space and defense missions requiring extreme environmental resistance. Moreover, the reusable IP-core nature of the design enables integration into multicore processors and AI-enabled embedded platforms, making it a scalable and future-ready solution for next-generation safety-critical applications

IX.REFERENCES

[1].S. Zaidi, J. Sampedro, "Suspicious Human Activity Recognition From Surveillance Videos," IEEE Xplore, 2016.

[2].A. Virkhe, "AI in Video Surveillance: Shaping the Future of Intelligent Security," LinkedIn, 2025.

[3].Grand View Research, "AI In Video Surveillancence Market Size | Industry Report, 2030," Oct. 2024.

[4].MarketsandMarkets, "AI in Video Surveillance Market Size, Share and Trends," Dec. 2024.

[5].IJRPR, "Suspicious Activity Detection - Journal Paper," 2025.

[6].IJARSCT, "Suspicious Activity Detection System," 2025.

[7].ScienceDirect, "Surveillance Detection - an overview," 2025.

[8].McKinsey & Company, "Charting a path to the data- and AI-driven enterprise of 2030," Sept.

[9].NextMSC, "AI CCTV Trends: Smarter Surveillance for Safer Cities," 2025.

[10].Deloitte, "Surveillance and Predictive Policing Through AI," April 2025.
2021.

[11].W. Luo, W. Liu, S. Gao, "A Revisit of Sparse Coding Based Anomaly Detection in Videos," ICCV, 2017.

[12].Y. Cong, J. Yuan, J. Liu, "Sparse Reconstruction Cost for Abnormal Event Detection," CVPR, 2011.

[13].S. R. Basha, V. Bhagyasree, P. S. Prasad, "Deep Learning Models for Suspicious Activity Recognition in Videos: A Survey," IEEE Access, 2023.

[14].Y. Li, Y. Tian, "Deeply Learned Models for Anomaly Detection in Surveillance Video," ACM Computing Surveys, 2022.

[15].A. Sabokrou, M. Fayyaz, M. Fathy, E. Adeli, M. Klette, "Deep-Anomaly: Fully Convolutional Neural Network for Fast Anomaly Detection in Crowded Scenes," CVPR Workshops, 2017.

[16].H. Hasan, J. Choi, J. Neumann, A. Roy-Chowdhury, L.S. Davis, "Learning Temporal Regularity in Video Sequences," CVPR, 2016.

[17].C. Lu, J. Shi, J. Jia, "Abnormal Event Detection at 150 FPS in MATLAB," ICCV, 2013.

[18].I. Goodfellow et al., "Generative Adversarial Nets," NIPS, 2014.

[19].Z. Zhao, H. Zhang, "Anomaly Detection Using Generative Adversarial Networks," IEEE Transactions on Neural Networks, 2020.

[20].D. Tran, L. Bourdev, R. Fergus, L. Torresani, M. Paluri, "Learning Spatiotemporal Features with 3D ConvNets," ICCV, 2015.

[21].A. Divvala, A. Farhadi, C. Guestrin, "Learning Everything about Anything: Webly-Supervised Visual Concept Learning," CVPR, 2014.
[22].S. Ji, W. Xu, M. Yang, K. Yu, "3D Convolutional Neural Networks for Human Action Recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013.
[23].Y. Wang, J. Yuan, N. Vaswani, "Deep Learning for Automatic Anomaly Detection in Video Surveillance: Review and Prospects,"IEEE Access, 2021.
[24].T. Hasan, M. Johnson, "A Survey on Deep Learning in Video Surveillance," Journal of Visual Communication and Image Representation, 2024.