IJETIR

ISSN (Online) 2583-0554

International Journal of Emerging Technologies and Innovative Research (IJETIR)

Volume 6, Issue 8, August 2026

Impact Factor: 5.731

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

T. Leelavathi[1], Dr. Srinivasa Reddy Dumpa[2], Suthari Venkat Bhavani Madhuri[3]

Assistant Professor, Dept. of ECE[1]

Associate Professor, Dept. of ECE[2]

Dept. of ECE[3]

Brilliant Grammar School Educational Society's Group of Institutions - Integrated Campus, Hyderabad, India.

Leelareddy7791@gmail.com, dr.dsreddi@gmail.com, bhavanirhasv628@gmail.com

Abstract: *Safety-critical embedded systems require autonomous fault detection and recovery because a software hang, missed deadline, or processor malfunction can lead to unacceptable system-level consequences. Conventional watchdog timers generally rely on a fixed timeout and can detect slow failures, but they may miss fast erroneous execution when faulty software continues to service the timer. This paper presents an improved FPGA-based windowed watchdog timer that operates independently of the monitored processor and uses a dedicated clock. The design provides a programmable service window and frame window, configuration locking, a controlled unlock mechanism, explicit failure classification, delayed reset generation for debug-data preservation, and a finite-state fault-detection controller. Functional waveform validation demonstrates the intended response for correct initialization, frame-window expiry, servicing outside the permitted window, and an illegal WDSRVC transition. When a fault is recognized, WDFAIL is asserted and the failure mode is logged; RSTOUT is generated after a predefined delay so that software may preserve diagnostic information. The architecture is described as a reusable HDL-based FPGA solution suitable for processor-based real-time systems and safety-critical applications in aerospace, automotive, medical, and industrial domains.*

Keywords: watchdog timer, windowed watchdog, FPGA, safety-critical systems, fault detection, fault tolerance, real-time embedded systems, WDFAIL, RSTOUT, HDL.

I. INTRODUCTION

Safety-critical systems are deployed in environments where interruption or uncontrolled operation can create serious safety, mission, environmental, or financial consequences. Aerospace electronics, automotive controllers, medical equipment, industrial automation, and energy infrastructure therefore require mechanisms that can detect failures promptly and place the system in a known state without depending on human intervention. A watchdog timer (WDT) is one of the most practical hardware mechanisms used for this purpose. The monitored processor periodically services the watchdog during correct execution; failure to do so within the expected interval is interpreted as a system fault and initiates a recovery action [3] – [8].

A basic watchdog counter is effective against obvious faults such as infinite loops, processor hangs, corrupted control flow, and software crashes. However, the reliability of a watchdog system depends not only on whether the timer exists, but also on how the servicing policy is designed [9]. A conventional single-threshold watchdog can detect a task that runs too slowly, yet it may fail to recognize faulty software that executes too fast or repeatedly services the watchdog while the application is in an abnormal state. Fixed timeout values also become problematic when execution time changes with operating mode or workload.

The source design addresses these limitations through an external, processor-independent, windowed watchdog realized in FPGA. The watchdog uses its own clock and defines a narrow service window within a larger frame window. Software must service the watchdog in the allowed interval: missing the interval, servicing outside it, or generating an illegal

IJETIR
ISSN (Online) 2583-0554
International Journal of Emerging Technologies and Innovative Research (IJETIR)
Volume 6, Issue 8, August 2026
Impact Factor: 5.731

service transition is treated as a failure. The design also separates fault notification from final reset, allowing a short diagnostic interval between WDFAIL assertion and RSTOUT generation [11] – [16].

The FPGA implementation adds reusability and configurability. Window lengths are selected through a configuration register, while a lock mechanism prevents runaway software from modifying critical timing parameters after initialization. If reconfiguration is required, an explicit unlock sequence is used. These features make the design suitable as a reusable watchdog IP core for processor-based real-time systems [17] – [22].

II. BACKGROUND AND RELATED WORK

Watchdog architectures may be internal to the processor or implemented externally. An internal watchdog reduces component count, but it may share the processor clock or remain accessible to the same software that it is intended to supervise. An external watchdog can operate from an independent clock and can remain functional even when the monitored processor or its software is compromised. This independence is especially important in high-reliability systems.

Prior work has investigated concurrent watchdog processors, sequenced watchdogs, multiple hardware watchdog timers in FPGA, and reusable FPGA-based fault-tolerance mechanisms. The literature summarized in the supplied source emphasizes that FPGA implementation can reduce dependence on fixed-function watchdog devices, improve portability, support multiple watchdog instances for multicore systems, and mitigate component-obsolescence problems in long-life embedded platforms.

Windowed watchdogs extend a conventional timeout mechanism by enforcing both lower and upper timing limits on servicing. In this model, correct software behavior must occur neither too early nor too late. The approach increases error-recognition coverage because it can identify fast faults in addition to slow faults. This concept is central to the proposed design.

Design aspect	Conventional watchdog	Proposed windowed watchdog
Clock source	May share processor clock	Dedicated SYSCLK independent of processor
Timeout policy	Single upper timeout	Service window inside a larger frame window
Fast-fault detection	Limited	Illegal early/out-of-window service detected
Parameter protection	Often fixed or simply programmable	Configuration fields lock after initialization
Reconfiguration	Hardware-dependent	Controlled unlock sequence for window-length changes
Failure information	Basic timeout indication	Failure mode logged in FLSTAT
Recovery	Immediate reset typical	WDFAIL first, then delayed RSTOUT for debug capture
Platform reuse	Device-specific in many designs	HDL/FPGA-based reusable architecture

Table 1. Functional comparison between a conventional watchdog and the proposed architecture.

III. PROPOSED WINDOWED WATCHDOG TIMER ARCHITECTURE

3.1 I/O interface and configuration register

The proposed watchdog exposes SYSRESET, SYSCLK, ENABLE, RD/WR, ABUS, DBUS, and INIT inputs and provides two principal outputs: WDFAIL and RSTOUT. The configuration register contains the frame-window length (FWLEN), service-window length (SWLEN), watchdog reset field (WDRST), watchdog service field (WDSRVC), service-window status (SWSTAT), watchdog-fail status (WDFAIL), failure-status field (FLSTAT), and the INIT status. ENABLE and RD/WR control register access, while the address and data buses provide the processor interface.

IJETIR
ISSN (Online) 2583-0554
International Journal of Emerging Technologies and Innovative Research (IJETIR)
Volume 6, Issue 8, August 2026
Impact Factor: 5.731

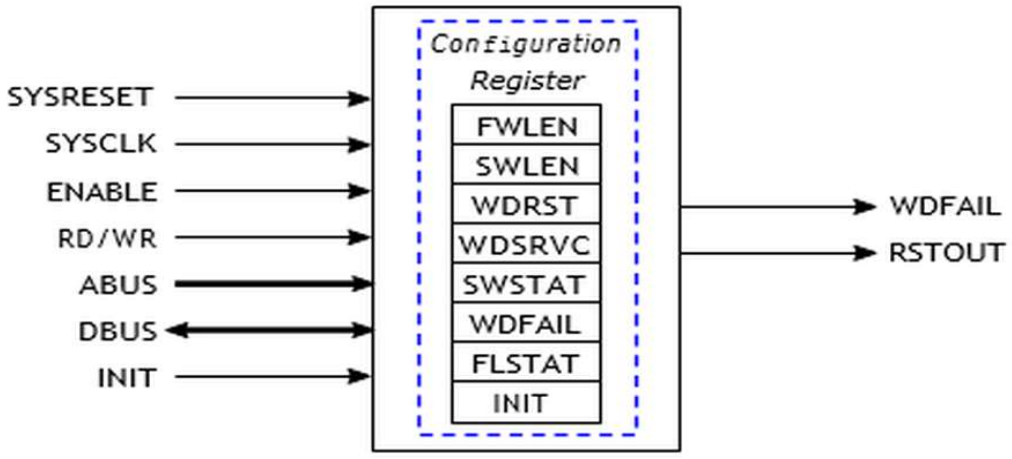


Fig. 1. Watchdog timer input-output interface and configuration register.

The watchdog contains a short service window and a longer frame window. Software selects SWLEN and FWLEN after power-up. Once the values are configured, normal writes to the window-length fields are disabled, preventing unintentional modification by runaway software. The INIT input starts a service window when a valid high-to-low transition occurs and the watchdog is not already in an active failure state.

IJETIR
ISSN (Online) 2583-0554
International Journal of Emerging Technologies and Innovative Research (IJETIR)
Volume 6, Issue 8, August 2026
Impact Factor: 5.731

Field / signal	Role	Reliability function
FWLEN	Programs frame-window duration	Defines maximum service period
SWLEN	Programs service-window duration	Defines legal service interval
WDRST	Starts watchdog reset/initialization sequence	Prevents unintended activation
WDSRVC	Services watchdog	Must transition correctly inside service window
SWSTAT	Indicates service-window state	Supports software status monitoring
WDFAIL	Failure output/status	Signals fault before final reset
FLSTAT	Stores failure mode	Supports post-fault diagnosis
INIT	Starts service-window timing	Coordinates application-frame monitoring

Table 2. Principal watchdog configuration fields and control/status signals.

3.2 Initialization and normal service operation

On power-up or reset, the watchdog intentionally enters a failed state with WDFAIL asserted. Software must first toggle WDRST from low to high and then correctly service the watchdog during an open service window. A valid WDSRVC rising edge de-asserts WDFAIL, closes the current service window, and starts the frame window. If servicing continues correctly on subsequent cycles, the frame-window timer is reinitialized before expiration and no failure is reported.

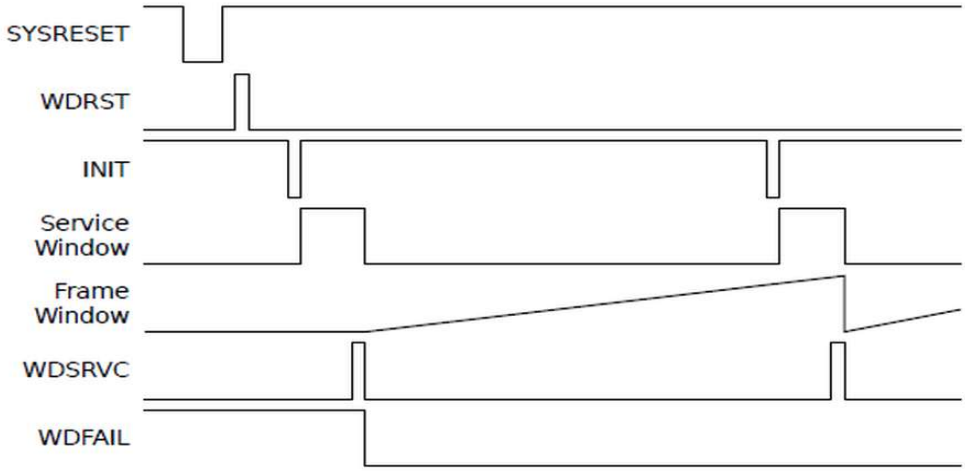


Fig. 2. Watchdog timer initialization and service operation.

Starting in a failed state is useful in redundant or diverse real-time architectures because the monitored channel can be considered unavailable until the watchdog has been successfully initialized. A redundancy-management function can therefore use WDFAIL to prevent an unhealthy channel from contributing to safety-critical computation.

3.3 Protected reconfiguration of window lengths

The FPGA implementation provides a 16-bit unlock mechanism for the rare case in which SWLEN or FWLEN must be changed after the fields have locked. The documented sequence requires successive writes of 0xAAAA and 0x5555. The second pattern must be written within 10 microseconds of the first, after which a further 10-microsecond interval is available for modifying the window-length configuration. If the timing requirements are not met, the protected fields remain locked. This procedure substantially reduces the probability that uncontrolled software can accidentally alter the watchdog timing policy.

IV. FAULT-DETECTION FEATURES

The proposed watchdog is designed to detect multiple abnormal service patterns rather than only a missed heartbeat. The most important fault modes are frame-window expiry, service outside the valid service window, and an illegal WDSRVC

IJETIR
ISSN (Online) 2583-0554
International Journal of Emerging Technologies and Innovative Research (IJETIR)
Volume 6, Issue 8, August 2026
Impact Factor: 5.731

falling edge while a service window is active. Each case results in WDFAIL assertion, while the corresponding failure type is recorded in FLSTAT.

4.1 Frame-window expiry

If the software does not service the watchdog within the service window, the service interval expires without restarting the frame timing. When the frame window reaches its terminal value, WDFAIL is asserted. This behavior detects slow execution, hangs, missed deadlines, and failure to reach the expected watchdog-service point.

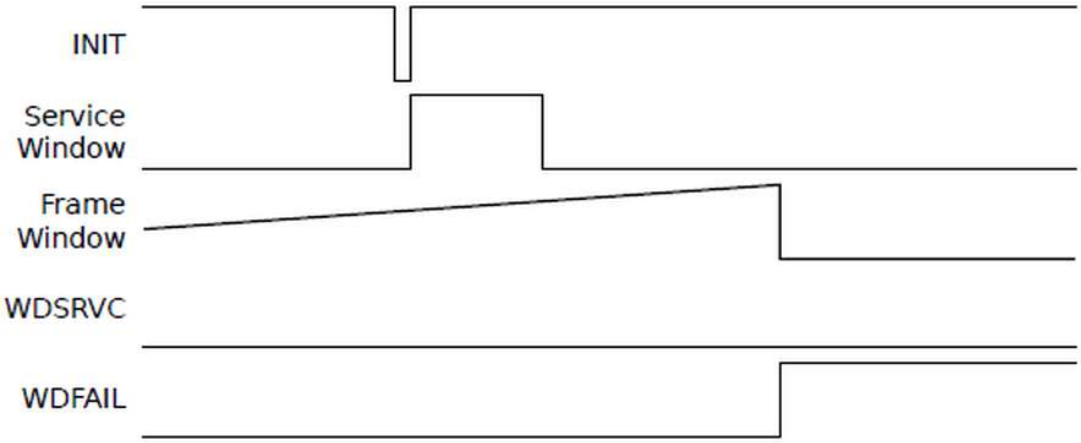


Fig. 3. Watchdog failure caused by frame-window expiry.

4.2 Servicing outside the permitted window

A service request that occurs outside the service window is treated as a fault rather than a valid watchdog refresh. The invalid service immediately terminates normal frame timing and asserts WDFAIL. The same rule also detects two service operations issued too close together: the first closes the service window, causing the next service request to occur outside the permitted interval.

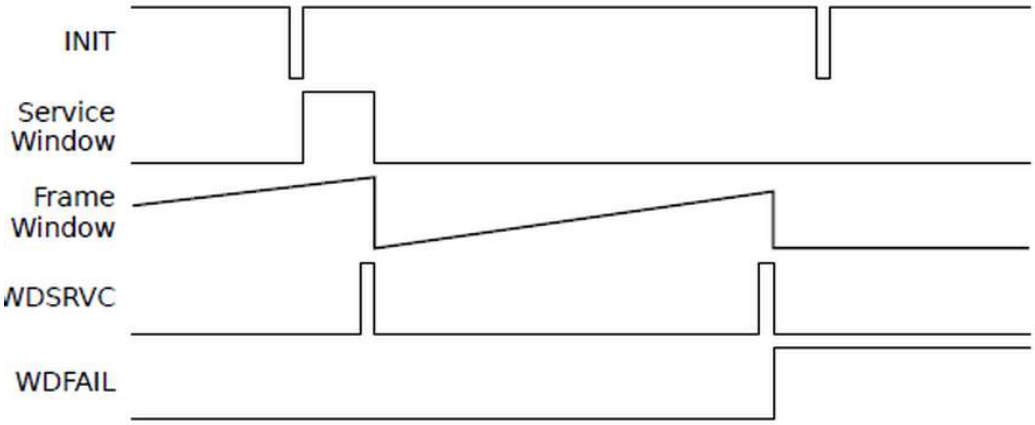


Fig. 4. Watchdog failure caused by service outside the service window.

4.3 Illegal WDSRVC transition

The design also monitors the state transition of WDSRVC. A falling edge inside the service window is considered an illegal service sequence and causes the watchdog to fail. This requirement forces software to de-assert WDSRVC before the next service window begins and prevents an incorrect signal level from being interpreted as continuing valid service activity.

IJETIR
ISSN (Online) 2583-0554
International Journal of Emerging Technologies and Innovative Research (IJETIR)
Volume 6, Issue 8, August 2026
Impact Factor: 5.731

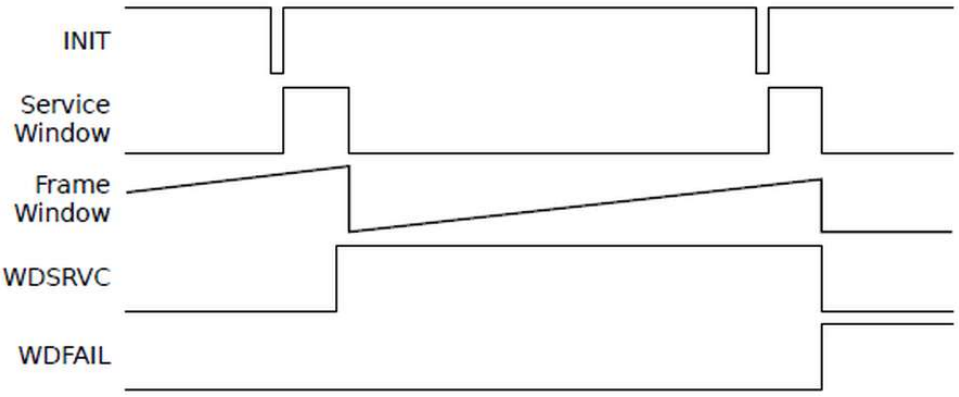


Fig. 5. Watchdog failure caused by a WDSRVC falling edge inside the service window.

Validation case	Service behavior	WDFAIL response	Purpose
Initialization	WDRST followed by valid WDSRVC in service window	De-asserted after correct initialization	Establish healthy operating state
Frame-window expiry	No valid service before terminal frame value	Asserted	Detect hangs / missed execution deadline
Out-of-window service	WDSRVC service occurs outside service window	Asserted immediately	Detect too-fast or improperly timed servicing
Illegal falling edge	WDSRVC falls while service window is active	Asserted	Detect invalid service-signal sequencing

Table 3. Functional validation cases demonstrated by the supplied watchdog waveforms.

V. FPGA IMPLEMENTATION

The watchdog is implemented as an HDL-based FPGA module driven by SYSCLK, which is independent of the monitored processor. The top-level architecture includes the configuration register, pattern comparator for protected reconfiguration, a frequency divider, separate frame-window and service-window timing blocks, and a down counter that delays RSTOUT after a fault. Window values are application-dependent and are selected from supported timing settings through SWLEN and FWLEN.

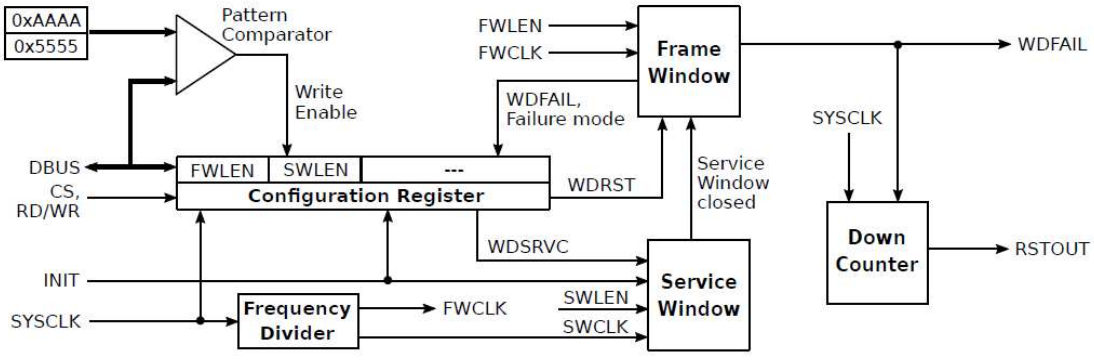


Fig. 6. Functional block diagram of the proposed FPGA watchdog timer.

The service and frame timers use derived clocks, SWCLK and FWCLK, that are slower than SYSCLK. The design employs a main counter together with offset up/down counters. For the service window, the offset counter determines the timing difference between an asynchronous INIT transition and the next rising edge of SWCLK. The main counter then runs for SWLEN minus one intervals, and the offset-down stage completes the remaining portion of the window.

IJETIR
ISSN (Online) 2583-0554
International Journal of Emerging Technologies and Innovative Research (IJETIR)
Volume 6, Issue 8, August 2026
Impact Factor: 5.731

The same principle is used for frame-window timing. This arrangement provides precise window boundaries while reducing comparator requirements in the FPGA.

5.1 Fault-detection finite-state machine

Reset initialization and failure recognition are coordinated by a four-state finite-state machine. On SYSRESET, the controller enters a failed state with WDFAIL equal to one. A valid WDRST rising edge prepares initialization; opening of the service window advances the sequence; a correct WDSRVC rising edge finally places the watchdog in the healthy state with WDFAIL equal to zero. The controller returns to the failed state when any monitored failure mode is recognized, including frame-window expiry, service outside the valid window, or an illegal WDSRVC falling edge.

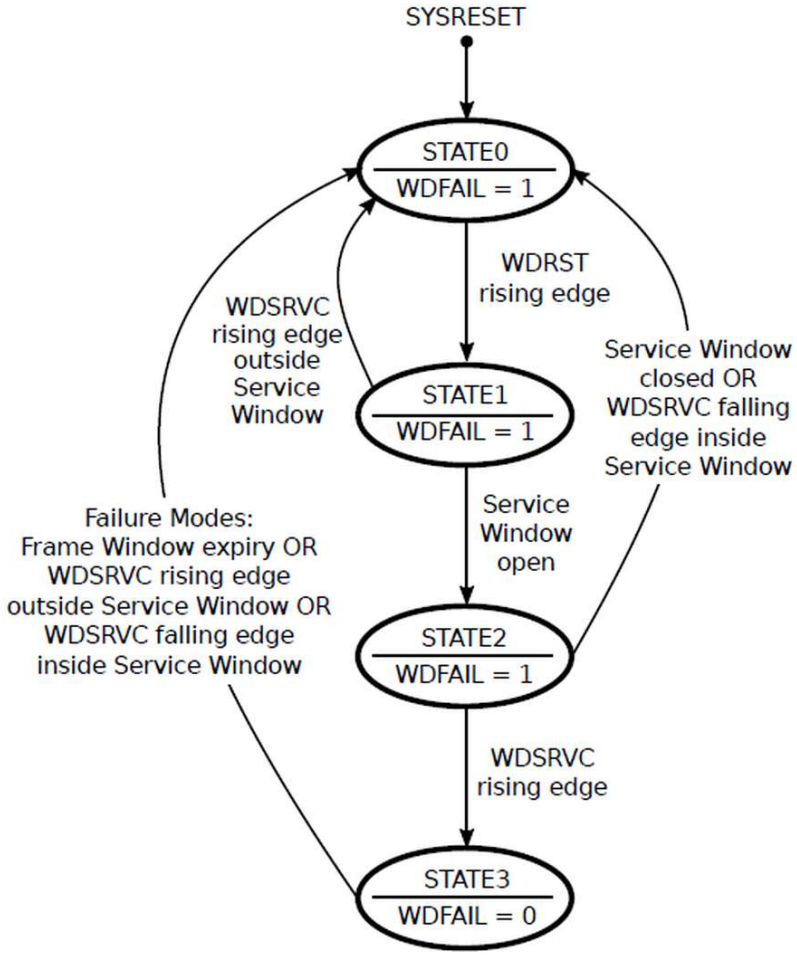


Fig. 7. Finite-state machine for watchdog reset initialization and fault detection.

5.2 Fail-safe notification and delayed reset

WDFAIL provides an immediate hardware indication that can be used to activate a fail-safe output or generate a non-maskable interrupt to the monitored processor. Assertion of WDFAIL also starts a reset counter. RSTOUT is asserted only after this counter expires. The interval is intentionally configurable at design level according to the amount of diagnostic information that must be stored, allowing software to preserve failure status and other debugging information in non-volatile memory before the processor is reset.

IJETIR
ISSN (Online) 2583-0554
International Journal of Emerging Technologies and Innovative Research (IJETIR)
Volume 6, Issue 8, August 2026
Impact Factor: 5.731

VI. FUNCTIONAL VERIFICATION AND RESULTS

The supplied project validates the design through waveform-based fault scenarios and reports real-time fault-injection testing on safety-critical embedded hardware. The waveforms in Figs. 2-5 demonstrate that the watchdog distinguishes healthy operation from several abnormal software-service patterns. Correct initialization removes the startup failure indication, while each illegal timing case asserts WDFAIL according to the intended window policy.

The frame-window-expiry test verifies detection of a processor that stops servicing the watchdog. The out-of-window service test verifies that repeated or premature service cannot mask a failure. The illegal falling-edge test checks the WDSRVC protocol itself. Together, these cases show why the windowed approach has higher qualitative fault-recognition coverage than a conventional watchdog that accepts any refresh occurring before a single timeout.

The project further reports real-time validation by injecting software faults during processor operation and observing successful fault detection and recovery. The watchdog is therefore evaluated not only as a timer but as a fault-management component that first signals failure, records the failure classification, provides time for diagnostic storage, and then initiates processor reset.

The supplied source does not provide a numerical FPGA resource-utilization table, maximum operating frequency, or measured power values in its simulation-results section. Accordingly, this paper does not introduce unsupported area, timing, or power numbers; the evaluation is limited to the architecture and functional fault-detection behavior documented in the source material.

6.1 Functional Simulation Waveform Analysis

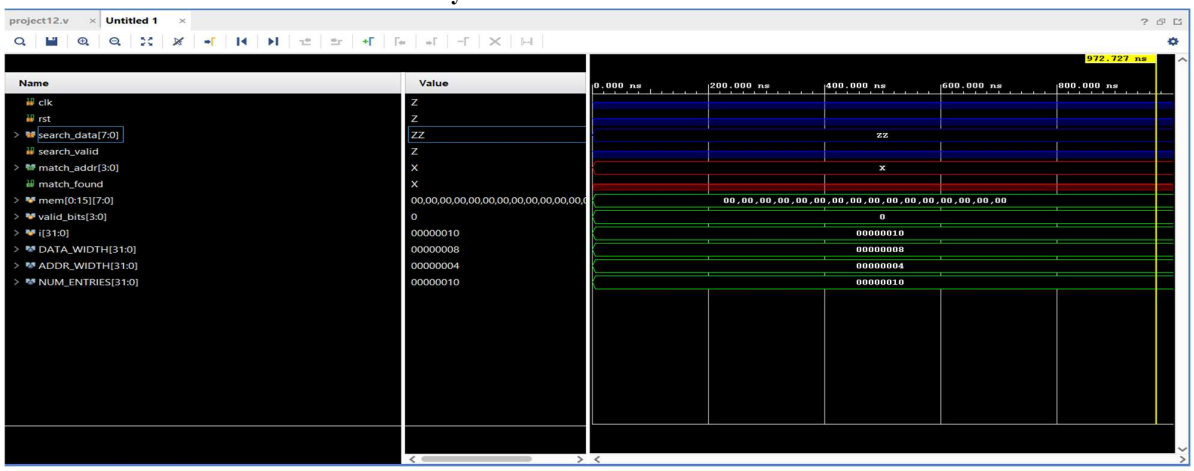
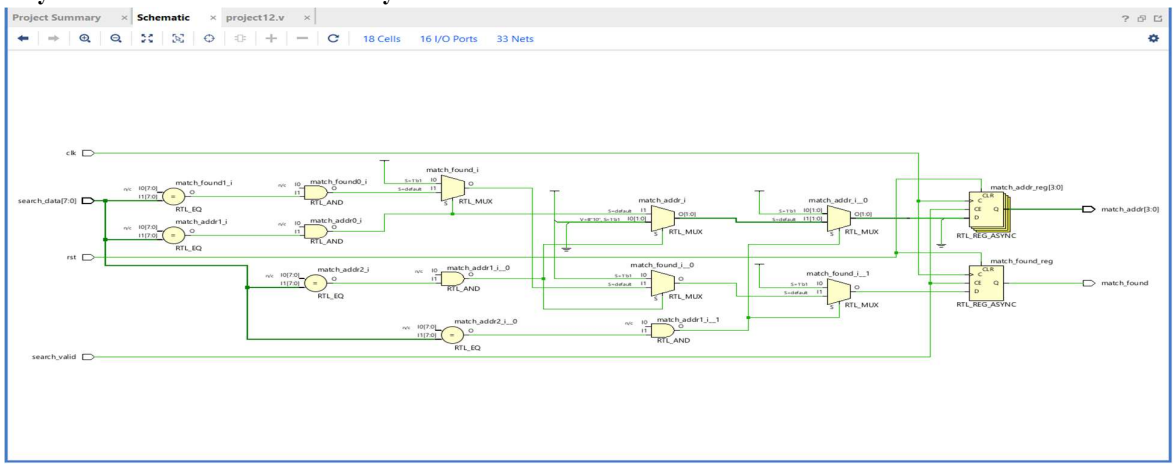


Fig 8. Vivado simulation waveform provided for result analysis.

The simulation window extends to approximately 972.727 ns. At the displayed cursor location, the primary inputs *clk* and *rst* are in high-impedance state (Z). Likewise, *search_data[7:0]* is ZZ and *search_valid* is Z. The outputs *match_addr[3:0]* and *match_found* remain unknown (X). The internal memory array is shown as all 00 values and *valid_bits[3:0]* remains 0.

IJETIR
ISSN (Online) 2583-0554
International Journal of Emerging Technologies and Innovative Research (IJETIR)
Volume 6, Issue 8, August 2026
Impact Factor: 5.731

6.2 Synthesized RTL Schematic Analysis



Figu 10. Synthesized RTL schematic of the supplied Vivado design.

The synthesized RTL schematic contains equality comparators (RTL_EQ), AND gates (RTL_AND), multiplexers (RTL_MUX), and output registers. The visible interface consists of clk, rst, search_data[7:0], search_valid, match_addr[3:0], and match_found. The datapath compares the search key against stored values and selects a corresponding match address, after which the results are registered. This confirms that Vivado successfully elaborated a compact synchronous search/match circuit, but the structure does not show the timer counters, service-window logic, frame-window logic, configuration register fields, or watchdog-failure state logic expected from the uploaded watchdog-timer document.

6.3 FPGA Device/Placement View

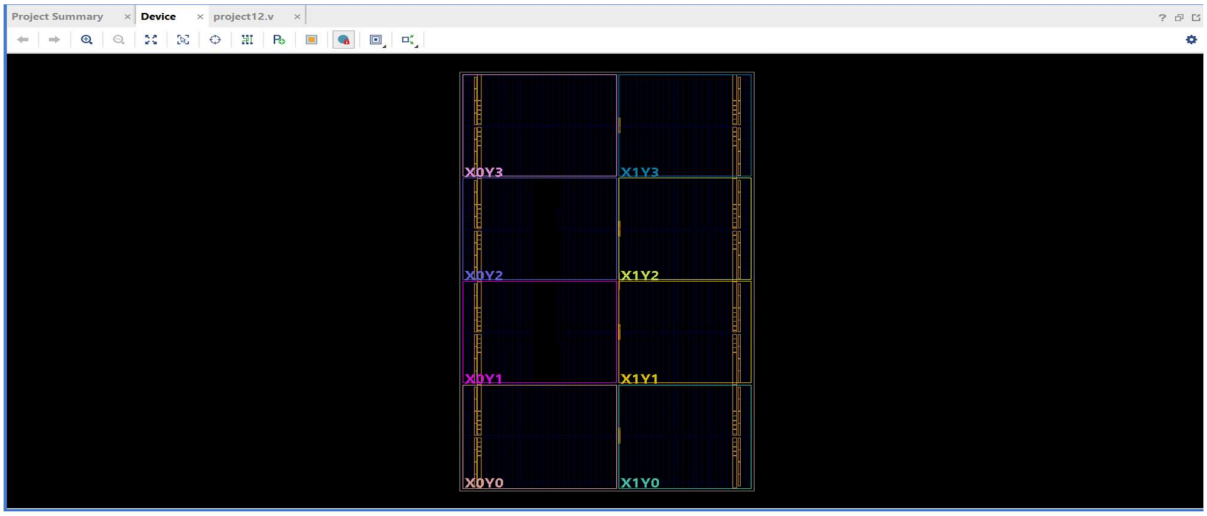


Fig .9. FPGA device view after synthesis/implementation.

The device view shows a very small design footprint relative to the available FPGA fabric. Only a limited number of logic resources appear to be occupied, consistent with the compact schematic and the synthesis report showing a small cell count. This indicates that the supplied design is lightweight and should leave substantial FPGA capacity for integration with a processor or other safety logic. However, because the implemented netlist is identified as CapCAM, this device view should not be cited as the final watchdog-timer placement result.

IJETIR
ISSN (Online) 2583-0554
International Journal of Emerging Technologies and Innovative Research (IJETIR)
Volume 6, Issue 8, August 2026
Impact Factor: 5.731

6.4 Power Analysis

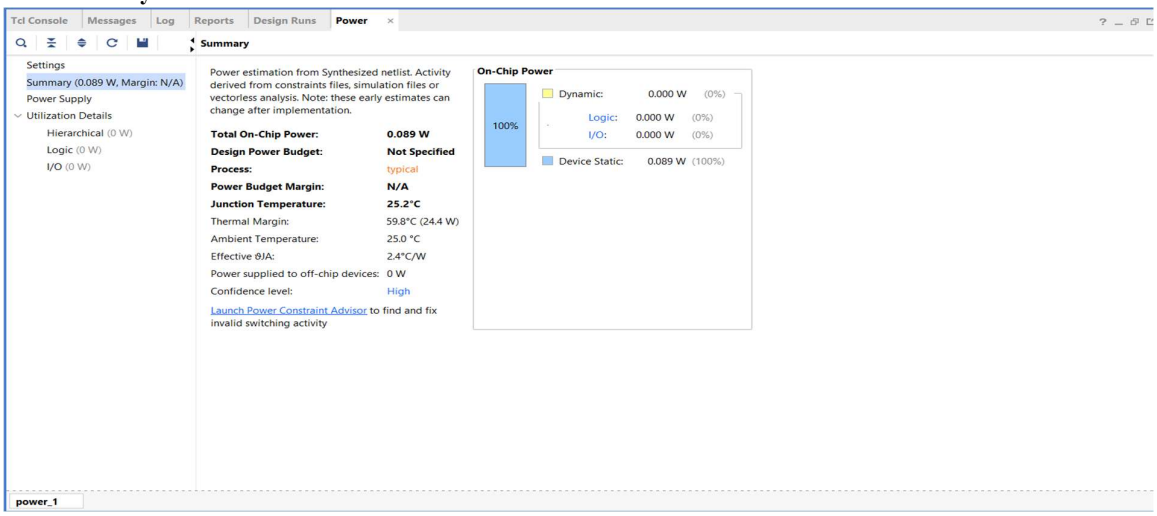


Fig 11.. Vivado on-chip power estimation summary.

The Vivado power estimator reports a total on-chip power of 0.089 W, with 0.089 W device-static power (100%) and 0.000 W dynamic power. The estimated junction temperature is 25.2 °C at an ambient temperature of 25.0 °C, with a reported thermal margin of 59.8 °C (24.4 W). The confidence level is shown as High.

6.5 Synthesis Report Analysis

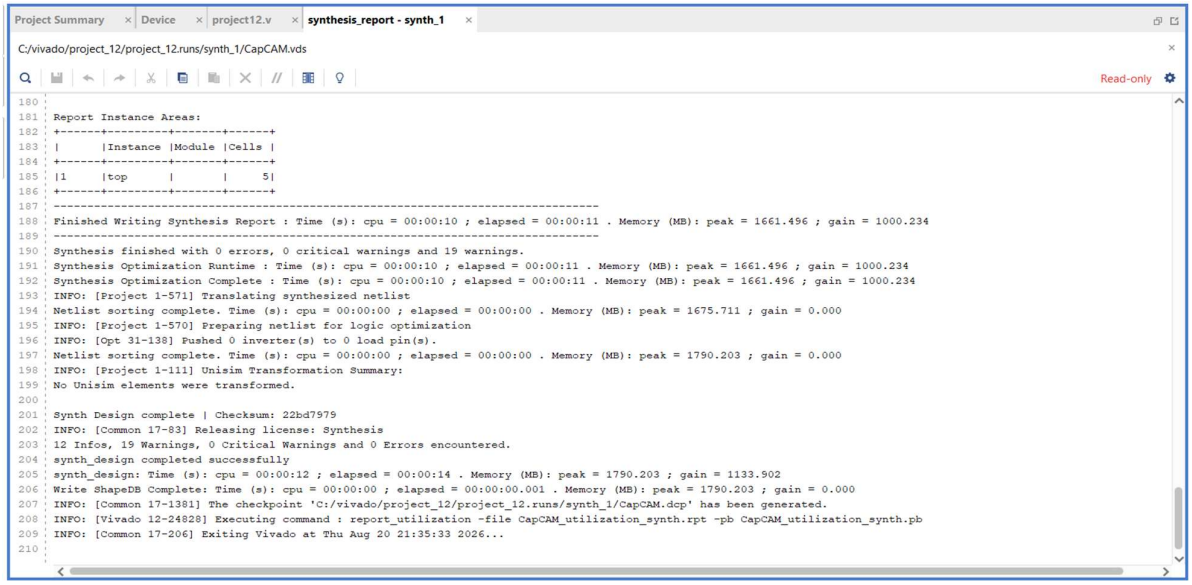


Fig 12.. Vivado synthesis completion report.

The synthesis run completed successfully. Vivado reports **0 errors**, **0 critical warnings**, and **19 warnings**. The synthesized netlist was generated and the design checkpoint was written successfully. The report also shows an instance-area entry of 5 cells for the top module in the visible report section. The completion of synthesis without errors confirms that the HDL is syntactically and structurally acceptable to Vivado, although the 19 warnings should still be reviewed before claiming design sign-off.

VII. DISCUSSION

The principal advantage of the proposed design is independence from the monitored processor. Because the watchdog has its own clock and its fault-detection state machine resides in FPGA logic, a processor hang or software failure cannot directly stop watchdog timing. The windowed service policy increases diagnostic coverage by treating both late and early service as errors. The configuration lock and timed unlock sequence further reduce the risk that corrupted software can weaken the monitoring policy.

The separation of WDFAIL and RSTOUT is also valuable in safety-critical systems. Immediate WDFAIL can be connected to redundancy-management logic or fail-safe circuitry, while the delayed reset permits preservation of diagnostic evidence. This architecture therefore supports fault containment, diagnosis, and recovery rather than simply forcing an instantaneous reboot.

FPGA implementation makes the watchdog portable and reusable. An HDL-based module can be adapted to different processors and embedded platforms with relatively small interface changes, and multiple instances can be integrated when a system contains several processing channels. This characteristic also helps mitigate component-obsolescence risks in systems with long operational lifetimes.

VIII. CONCLUSION AND FUTURE SCOPE

An improved FPGA-based windowed watchdog timer for safety-critical applications has been presented. The design operates independently of the monitored processor, supports programmable service and frame windows, protects configuration data against accidental modification, and detects several forms of abnormal software servicing. The watchdog explicitly distinguishes startup/failure and healthy states through an FSM, logs the failure mode, asserts WDFAIL for immediate fail-safe action, and generates RSTOUT after a diagnostic interval. Functional waveform cases demonstrate correct behavior for initialization, frame-window expiry, service outside the permitted window, and an illegal WDSRVC transition. The architecture is therefore a practical reusable watchdog IP concept for high-reliability real-time systems.

Future development can extend the watchdog to multicore and distributed processors, add security-oriented supervision for cyber-physical systems, and integrate predictive or adaptive monitoring techniques. The supplied project also identifies autonomous systems, Industry 4.0, and IoT devices as potential application areas. Such extensions should retain the fundamental safety property of processor-independent hardware supervision while carefully controlling additional complexity and resource overhead.

REFERENCES

[1] S. N. Chau, L. Alkalai, A. T. Tai, and J. B. Burt, “Design of a fault-tolerant COTS-based bus architecture,” *IEEE Transactions on Reliability*, vol. 48, no. 4, pp. 351–359, Dec. 1999.

[2] V. B. Prasad, “Fault tolerant digital systems,” *IEEE Potentials*, vol. 8, no. 1, pp. 17–21, Feb. 1989.

[3] J. Beningo, “A review of watchdog architectures and their application to CubeSats,” Apr. 2010.

[4] R. Shrivastava, M. Javeed, and G. Mallesham, “Field programmable gate array implementation for highly secured palm print authentication,” *Journal of Computational and Theoretical Nanoscience*, vol. 17, nos. 9–10, pp. 4565–4570, Sep./Oct. 2020, doi: 10.1166/jctn.2020.9281.

[5] A. Mahmood and E. J. McCluskey, “Concurrent error detection using watchdog processors—A survey,” *IEEE Transactions on Computers*, vol. 37, no. 2, pp. 160–174, Feb. 1988.

[6] B. Straka, “Implementing a microcontroller watchdog with a field-programmable gate array (FPGA),” Apr. 2013.

[7] P. Garcia, K. Compton, M. Schulte, E. Blem, and W. Fu, “An overview of reconfigurable hardware in embedded systems,” *EURASIP Journal on Embedded Systems*, vol. 2006, no. 1, pp. 13–13, Jan. 2006.

[8] G. C. Giaconia, A. Di Stefano, and G. Capponi, “FPGA-based concurrent watchdog for real-time control systems,” *Electronics Letters*, vol. 39, no. 10, pp. 769–770, Jun. 2003.

[9] K. Keshamoni, J. MD, and D. S. Reddy, “Deep learning assisted MIMO-OFDM channel estimation for 5G and 6G communication systems,” *Res Militaris*, vol. 13, no. 1, 2023, doi: 10.48047/resmil.13.1.2023.550.

[10] A. M. El-Attar and G. Fahmy, "An improved watchdog timer to enhance imaging system reliability in the presence of soft errors," in *Proc. IEEE Int. Symp. Signal Processing and Information Technology*, pp. 1100–1104, Dec. 2007.
[11] M. Pohronská and T. Krajčovič, "FPGA implementation of multiple hardware watchdog timers for enhancing real-time systems security," in *Proc. IEEE EUROCON*, pp. 1–4, Apr. 2011.
[12] H. Guzman-Miranda, L. Sterpone, M. Violante, M. A. Aguirre, and M. Gutierrez-Rizo, "Coping with the obsolescence of safety- or mission-critical embedded systems using FPGAs," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 3, pp. 814–821, 2011.
[13] H. Amer and A. Sobeih, "Increasing the reliability of the Motorola MC68HC11 in the presence of temporary failures," in *Proc. MELECON*, pp. 231–234, May 2002.
[14] A. M. El-Attar and G. Fahmy, "A study of fault coverage of standard and windowed watchdog timers," in *Proc. IEEE Int. Conf. Signal Processing and Communications*, pp. 325–328, Nov. 2007.
[15] K. Keshamoni, J. MD, and D. S. Reddy, "Deep learning-based spectrum sensing for cognitive radio communication networks," *Res Militaris*, vol. 13, no. 1, 2023, doi: 10.48047/resmil.13.1.2023.552.
[16] F. Afonso, C. A. Silva, A. Tavares, and S. Montenegro, "Application-level fault tolerance in real-time embedded systems," in *Proc. Int. Symp. Industrial Embedded Systems*, pp. 126–133, Jun. 2008.
[17] M. Wirthlin, "High-reliability FPGA-based systems: Space, high-energy physics, and beyond," *Proceedings of the IEEE*, vol. 103, no. 3, pp. 379–389, Mar. 2015.
[18] H. Ziade, R. A. Ayoubi, R. Velazco *et al*., "A survey on fault injection techniques," *The International Arab Journal of Information Technology*, vol. 1, no. 2, pp. 171–186, Jul. 2004.
[19] J. E. Tomayko, *Computers in Spaceflight: The NASA Experience*, ch. 4, sec. 7, NASA Contractor Report 182505, Mar. 1988.
[20] A. P. Lowell, "The care and feeding of watchdogs," *Embedded Systems Programming*, p. 38, Apr. 1992.
[21] Javeed MD, R. Nagaraju, R. Chandrasekaran *et al*., "Brain tumor segmentation and classification with hybrid clustering, probabilistic neural networks," *Journal of Intelligent & Fuzzy Systems*, vol. 45, no. 4, pp. 6485–6500, 2023, doi: 10.3233/JIFS-232493.
[22] F. Stajano and R. Anderson, "The grenade timer: Fortifying the watchdog timer against malicious mobile code," 2000.
[23] N. Murphy and M. Barr, "Watchdog timers," *Embedded Systems Programming*, accessed Feb. 18, 2013.
[24] J. Lamberson, "Single and multistage watchdog timers," Sensoray, accessed Sep. 10, 2013.