Design of Improved Watch dog Timer By using FPGA

M. KULLAYAPPA, P. RADHA, M. SATEESH KUMAR
ASSISTANT PROFESSOR [1,2,3]
manthri845@gmail.com, radhasvec@gmail.com, steeshkumar9f@gmail.com
Sri Venkateswara Institute of Technology,
N.H 44, Hampapuram, Rapthadu, Anantapuramu, Andhra Pradesh 515722

***Abstract*—** Maximum dependability is required of embedded systems used in safety-critical applications. Such systems automatically manage and recover from problems related to operating time using external watchdog clocks. When it comes to functionality, most of the external watchdog clocks on the market utilise extra circuitry to change their timeout durations and provide very limited functionalities. An enhanced adjustable watchdog timer which may be used in safety-critical applications is described in this work along with its architecture and design. The watchdog's resilience is enhanced by its several built-in fault detection systems. It may be used to monitor the activities of any processor-based real-time system since the capabilities and operations are relatively broad.In addition to discussing the suggested watchdog timer, this article delves into the development of a Field Programmable Gate Array (FPGA).As a result, the system cost is reduced and the design is readily adaptable to diverse applications. First, by looking at the simulation data, we can see how well the suggested watchdog timer detects and reacts to problems.By manipulating the software to introduce errors while the processor is running, the design is tested on real-time hardware, and the results are analysed.

INTRODUCTION

The most reliable system is needed for situations where human damage might result from a system crash. For these systems to function safely, they need fault tolerance techniques that can handle the unexpected. It is expected that these systems can also recover from crashes without any intervention from humans. As soon as a problem arises, these fault tolerance mechanisms kick in to fix it and keep the system running as smoothly as possible [1]. System redundancy is one approach to fault tolerance. Improving the system's overall dependability is possible with the use of numerous copies of its important components [2].But, depending on the design, this enhanced system dependability is accomplished by increasing the complexity of both the hardware and the software.
The watchdog is a low-cost, high-performance method for identifying and addressing operation-time related problems in fault-tolerant system development [3]. A hardware component known as a watchdog timer (WDT) keeps an eye on the system's activities and triggers certain actions when a malfunction is detected [4]. The CPU must regularly reset the timer, which is a common component. A secondary sign of an issue with the monitored system is when the WDT expires [5]. Restarting the system is decided upon when the CPU is unable to reset the watchdog. system, or restore it to a known-good condition from which it may recover; this will stop any more harm from happening. Both on-chip and off-chip versions of the watchdog are possible. Although it is not a strong solution, an internal watchdog may simplify and lower the cost of the hardware.A runaway code may deactivate the watchdog timer, and the software has control over it during runtime [3]. Also, the watchdog can't detect hardware problems if the crystal fails since it's linked to the CPU clock [6]. External watchdogs are essential when an embedded system's dependability is critical.An external watchdog does not share its clock with the CPU and operates independently. As a result, fault-tolerant system topologies are much stronger, beyond the constraints of internal watchdogs [7].
One kind of standalone watchdog timer microchip is less generic since it only has predefined timeout intervals. The timeout durations may be adjusted with the use of an extra set of devices that use external circuitry. Despite its practicality, this approach raises system costs and complicates hardware. By implementing the watchdog functionality inside a Field Programmable Gate Array (FPGA), the added complexity and expense of external watchdogs may be mitigated to some degree.To provide the required system functionality, many contemporary embedded systems include one or more FPGA chips [8]. By incorporating the watchdog timer into an FPGA, a reliable and effective solution may be achieved. For real-time control systems, Giaconia et al.[9] investigated the possibility of implementing a bespoke concurrent watchdog processor on FPGA.Instead of including a CPU timer, the design checked the reasonableness of many variables and the program's execution. To detect the occurrence of a malfunction, El-Attar et al. [10] suggested a sequenced watchdog timer that relies on time registers. Nevertheless, the defect detection features that were included were restricted, and there were not many configuration choices to choose from. The authors of [11] discussed the fundamental ideas of an FPGA-based multiple-hardware watchdog timer system, but they kept the watchdog's architecture simple.
This work presents the design and implementation of an enhanced windowed watchdog timer in FPGA. The architecture may be realised in FPGA such that the same watch-dog hardware can be interfaced to multiple systems and processors with only small alterations to the related HDL code [8]. For multicore designs, it also makes it possible to accommodate several watchdog timers.A suggested watchdog timer is ideal for

embedded systems that must prioritise safety, namely those that use redundant channels to improve system dependability.

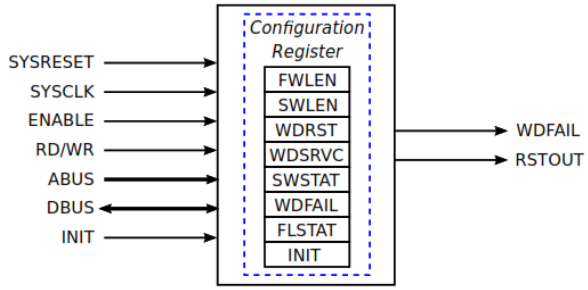
One solution to the problem of component obsolescence that plagues many embedded systems, particularly those used in aerospace and military applications, is to design the WDT as a reusable IP core [12]. The study delves into the design of the suggested watchdog timer, including its architecture, fault detection capabilities, and FPGA implementation. What follows is an outline of the rest of the paper. The suggested watchdog timer's design is detailed in the section that follows. In Section III, we covered the watchdog's built-in defect detection techniques. The watchdog timer is implemented in FPGA in Section IV. In section V, we discuss the simulation results and evaluate the hardware design. The article is concluded in section VI.

I. PROPOSEDWATCHDOGTIMERARCHITECTURE

A good watchdog will be able to restore the system to a known state the moment it detects any unusual software mode. It needs its own clock and the ability to send a hardware reset signal to any and all peripherals when a timeout occurs [3]. This article proposes a watch-dog timer that runs on a separate clock and is therefore not reliant on the CPU. During setup, the programme may specify the window periods, which follow a windowed watchdog implementation in the architecture.When the watchdog timer goes out, an alarm goes off, and after a certain period of time has passed, the system is reset. The programme may utilise that time to save important debugging data to a non-volatile memory. System hangs caused by infinite loops in code execution are catchable by a typical watchdog timer. Nevertheless, this watchdog's biggest drawback is that it won't ever notice a fault condition if the system enters one and keeps resetting the timer.To rephrase, although a regular watchdog timer is capable of detecting slow faults, it is unable to detect quick errors that happen within the watchdog timer period [13].Nevertheless, this can be handled well by a windowed design. To prevent a timeout, the watchdog specifies a short window of time in which it must be reset.This increases the coverage of mistake detection and protects systems from operating too quickly or too slowly [14]. Chapter A. I/O Interface and Configuration
As shown in Figure 1, the proposed watchdog timer has an input-output (I/O) interface. A watchdog failure signal (WDFAIL) and a reset signal (RSTOUT) are the two possible outputs from the watchdog. The WDFAIL and RSTOUT outputs are maintained in an assert and de-asserted state, respectively, while the SYSRESET input is low. A configuration register with the bit fields specified as shown in the picture is also part of the design. In addition to providing status information, the register allows for modifications to the watchdog's settings. Resetting and servicing the device are done using the WDRST and WDSRVC fields, respectively.

security dog.The configuration register is automatically updated with the current state of the INIT input and the WDFAIL output.
If there is a watchdog failure mode, it is recorded in the FLSTAT field, and the service window status is held in the SWSTAT field. You may read and write to the configuration register using the ENABLE and RD/WR control inputs to the watchdog timer.Address bus and data bus are represented in the figure by the signals ABUS and DBUS, respectively.

Figure 1: The configuration register and input/output interface of the watchdog timer
A service window and a frame window make up the suggested windowed watchdog concept.The duration of the service window will be much less than the frame window. The software has the ability to configure the length of the two windows in the configuration register after powering up by writing to the bit fields SWLEN and FWLEN. By design, you cannot change the settings of the window periods after they have been established after power-up. If the programme needs to write to the configuration register again, it will have to go through a strict unlock process.This prevents runaway code from inadvertently changing the parameters of the watchdog window.



The watchdog timer's INIT input is used to initialise the service window. Assuming the fail flag (WDFAIL) is not set to active, the service window will be initiated by a high-to-low transition on this input. To avoid a timeout, the CPU must service the watchdog during the service window. The configuration register's watchdog service (WDSRVC) field is used to service the watchdog timer. The service window will be closed and the frame window will be started immediately if this part inside of it has an upward edge. How often the watchdog needs servicing is defined by the frame window. The embedded control system's main loop typically uses a somewhat longer time for this window, and the watchdog is serviced once per cycle [15]. Various methods exist for driving the INIT signal to the watchdog timer. One approach is to do some sanity tests before ending the main loop and then trigger the INIT signal [16]. To prevent the CPU from interfering with the INIT signal generation, an external interval timer may be used. Here, you want to make sure the frame window is set for a little longer than the main Loop execution period. If your embedded system uses frames to arrange its duties, this configuration option is tailor-made for you.

A. WatchdogTimerInitialization

Onpower-uporresetthewatchdogwakesupinafailed state, i.e., the WDFAIL output will be asserted high. It is the responsibility of the software to initialize the watchdogand keep it running. Fig. 2 illustrates the waveform for watchdog reset initializationand general operation.In order tobringthe watchdog to a working state, first the watchdog reset (WDRST) field in the configuration register must be toggled from low-to-high. This, followed by servicing the watchdog insidethe service window,will de-assertthe WDFAIL flag and make it operational.Since the frame window is keptlarger than the system frame time, another service windowwill start before the current frame window expires. When the watchdogisagainproperlyserviced,theframewindowwillb e reinitialized. As long as the frame window counters keep running, no failures will be flagged by the watchdog.

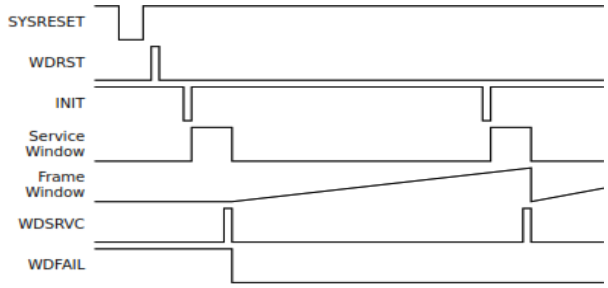


Fig. 2. Watchdog timer initialization and service operation

Critical real-time embedded systems make use of redundancy or diversity to achieve fault-tolerance [17]. Asserting the watchdog fail signal on power-up proves to be a useful feature for such systems. The fail state can be used to indicate that a particular channel is unavailable for computations.Oncethewatchdogisbroughttoahealthystate , the channel can be declared online. Moreover, during normal operations if a particular channel is found to be functioning abnormally, the redundancy management logic can activatethe watchdog fail of that channel.This can effectively withdraw the faulty channel from taking part in any further computations.

II. FAULTDETECTIONFEATURES

Several fault detection mechanisms are built into the pro- posed watchdog timer in order to improve its effectiveness in capturing erratic software modes. When the software fails to service the watchdog inside the service window, the window expires and sets a fail flag internally. In this case, the frame window does not reinitialize and expires upon reaching its terminal value. On the expiry of the frame windowthe watchdog asserts its WDFAIL signal, indicatingafailure.ThisfailuremodeisdepictedinFig.3

When the programme serves the watchdog outside of the service window, a watchdog fail will occur (Fig. 4)

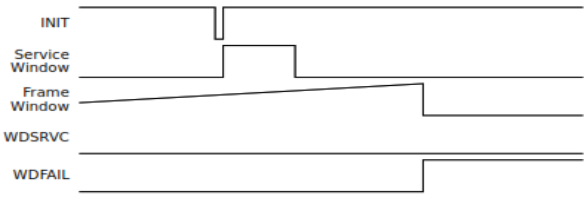


Fig. 3. Watchdog fail due to frame window expiry

It is clear that the frame window is immediately terminated and the WDFAILsignal is asserted due to the invalid service activity. A positive side effect of this feature is that it will also cause a watchdog failure if two service activities are executed consecutively. In this case, the service window will be closed instantly upon the initial action, and the subsequent operation will always take place outside of the window.The result is a watchdog that doesn't work since it's the same as trying to service it outside of its service window.

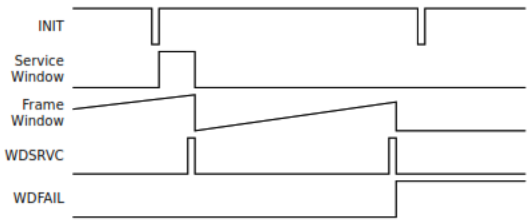


Fig. 4. Watchdog fail due to service outside the service window

If the WDSRVC falling edge happens inside the service window, as shown in Fig. 5, then... This is likewise seen as an unauthorised service action, which triggers the assertion of the watchdog fail signal. This means that prior to the next service window starting, the programme must de-assert the WDSRVC signal after servicing the watchdog. With all these problem detection systems in place, the suggested watchdog timer will not miss any programme that is behaving erratically.

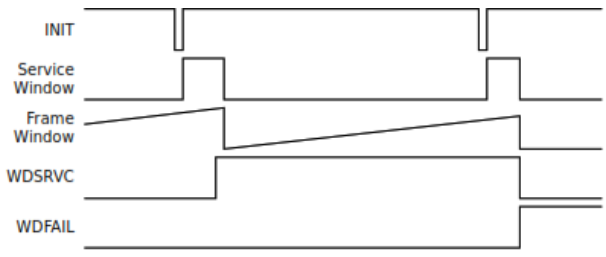


Fig. 5. Watchdog fail due to WDSRVC falling edge inside service window

One possible application for the WDFAIL output of the watchdog timer is to trigger a fail-safe condition or to alert the processor to the fault by sending an NMI signal.The watchdog will assert its RSTOUT output after a preset period of time after asserting the WDFAIL output. By connecting this signal to the processor's reset pin, the embedded system may be reset automatically. Software has a chance to preserve information that can be useful for troubleshooting during this period. The watchdog configuration register's FLSTAT field will record the associated failure mode in the case of a failure. For debugging reasons, the programme may also try to store this data to non-volatile memory.

III. Implementing Watchdog Timer in FPGA

The implementation of the suggested watchdog timer in FPGA is described in this section. This is the schematic of the watchdog hardware at a high level.Picture 6.The design's SYSCLK input keeps it timed apart from the processor's internal clock. The design comes up with the potential sets of window lengths depending on the application. After powering on, you may choose these values by writing to the matching bits in the configuration register: SWLEN for the service window and FWLEN for the frame window. After the settings are chosen, the fields for configuring the window length are automatically locked, meaning that writes to these bits are disabled. A 16-bit unlock register is included in the design in case the windowlengths need to be changed again. The programme has to write the values 0xAAAA and 0x5555, in that sequence, to this register in order to modify the window widths. The second pattern has to be typed within 10 µs after the first one, and then the programme has 10 µs to change the length configuration fields. These bits will not be able to be written to unless these times are satisfied precisely. When the INIT signal detects a change from high to low, the service window is initiated.A far slower derived clock (SWCLK) than the SYSCLK is used by the service window. The slower clock helps to minimise resource use in FPGA by lowering the number of comparators needed. The service window has a primary counter that runs at SWCLK and an offset up/down counter that is timed by the SYSCLK.Between the INIT input and the next rising edge of the SWCLK, the offset upcounter detects the offset (Toffset). Given that the INIT signal could arrive at any point within the Tswclk period of the SWCLK, which is asynchronously driven, this is mandatory. We begin by saving the offset value and then start the main counter, which will run for (SWLEN - 1) times. The period of the offset down counter is Tswclk−Toffset, and it begins after the main counter ends. The window length may be precisely controlled using this counting approach.The watchdog configuration register is also updated on a periodic basis with the operating state of the service window.

As soon as the watchdog is serviced properly, the counters in the service window stop and the frame window begins.

starts. For its activities, the frame window additionally makes use of a derivative slower clock (FWCLK).Similar to the service window, it contains a primary counter and an offset up/down counter. This is where the offset up counter comes in handy; it detects the offset between when the service window ends and when the FWCLK next rises. Before the offset down counter is followed, the main counter counts for (FWLEN-1) times. When the next service window period passes without a watchdog service action, the frame window counts reset.

A. Resetting the System and Identifying Problems

Figure 7 shows the final state machine (FSM) diagram.

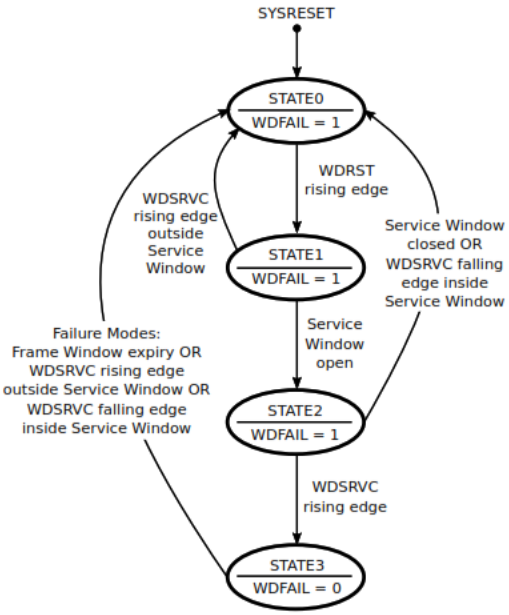


Fig. 7. Finite state machine design of watchdog fault detection logic

launch of the watchdog's reset initialization and fault detection techniques. A watchdog failure is indicated by the asserted WDFAIL output at power-up.In order to initialise the watchdog timer, a rising edge on the WDRST bit is used. The WDFAIL output is de-asserted and the window counters begin running as soon as the service window opens, which is caused by a rising edge on the WDSRVC bit. The programme will have to start from the beginning of the startup process all over again if the watchdog is serviced erroneously. Once the watchdog is initialised correctly, the WDFAIL signal is de-asserted.

Even if the watchdog is operational, it will assert the WDFAIL output again in the event that any of the failure types listed in section III happens.Notification of the failed status and failure type is updated in the configuration register.In addition, a reset counter that runs for a preset period of time is triggered when the watchdog fail assertion is made. Taking the quantity of debug information into account will decide the length of the counter.

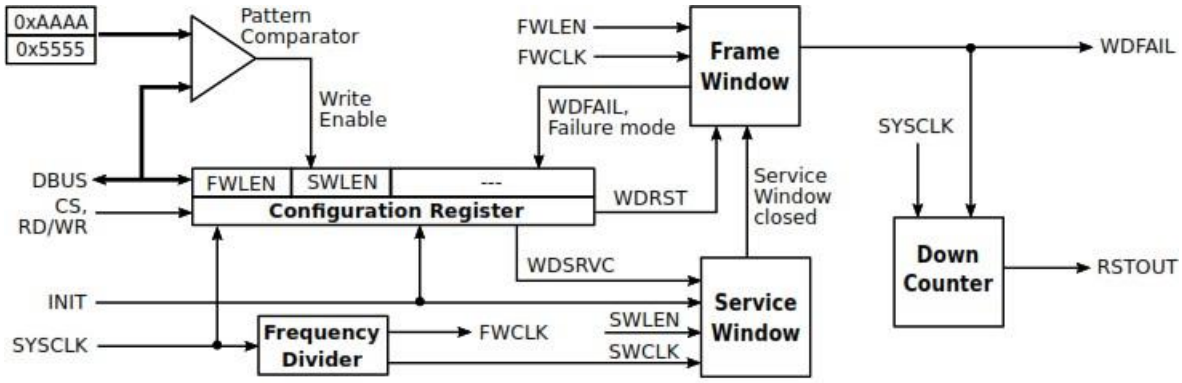


Fig. 6. Functional block diagram of the proposed watchdog timer

thatneedstobe stored. Onthe expiryofthecounter,the WDT asserts its RSTOUT output high. The reset counter will be non-functional duringpower-upandtheRSTOUToutputwill besettolowatthispoint.Whenthewatchdogisinitialized for the first time, the counter gets automatically enabled.

III. EXPERIMENTALRESULTS

The proposed watchdog timer architecture has been implemented using VHDL and realized in a FPGA device. A dedicated 25 MHz clock signal was used for the SYSCLK input. Possible values for the window lengths were calculated based on the application and embedded in the design. In one particular implementation of WDT for an embedded control system, the service window duration could be 100µs, 200µs, 400µs or 800µs.The frame window had eight selectable options - 1ms, 2ms, 5ms and then up to 30ms in steps of 5ms. The processor could select the desired window lengths by writing the appropriate value to the configuration register. A programmable interval timer was implemented in the FPGA and the expiry of the timer was used to drive the INIT signal. The WDFAIL output from the watchdog was used as an interruptrequesttotheprocessorandtheRSTOUToutputwas connected to the reset pin of the processor. The reset counter wasdesignedtorunfor 3milliseconds. This value wasarrived after calculating the amount of fault log information that will have tobewrittentotheNVRAMpresentinthesystem,inthe case of a failure. The duration of the reset pulse from the watchdog timer was also set according to the reset input requirements of the processor.

Theproposedwatchdogtimerdesignhasbeen simulated using ModelSim software by creating adequate test benches andrunningtheacceptance test procedures (ATP).A processor bus function model was used to access the configuration register and service the watchdogas per the ATP. Fig. 8 shows the simulated waveformfor WDT reset initialization. On power-up, the WDFAILoutput of the watchdog is asserted high to indicate a failure. It can be seen from the waveform that the servicewindow opens (SWSTAT=1) when the INIT signal goes low. Inside the servicewindow, the WDRST bit is set to high and then the WDSRVCbitistoggledfromzerotoone.Thisclosesthe service window immediately and de-asserts the WDFAIL output.

The functionality of the watchdog for all possible combi- nations of window lengths were simulated and verified. Using emulation based fault injection techniques, faults were introduced in the test bench models. All the three failure scenarios mentioned in section III were created andthe response of the design was analyzed. In all the cases the watchdog detected the fault, asserted the WDFAIL signal, classified the failure mode and logged the fault in the configuration register, before initiating a system reset. The simulated waveform in Fig. 9 shows the response of the watchdog timer to an improper service operation. It can be seen that the watchdog fail signal, WDFAIL, is asserted within a short time of 81 ns.

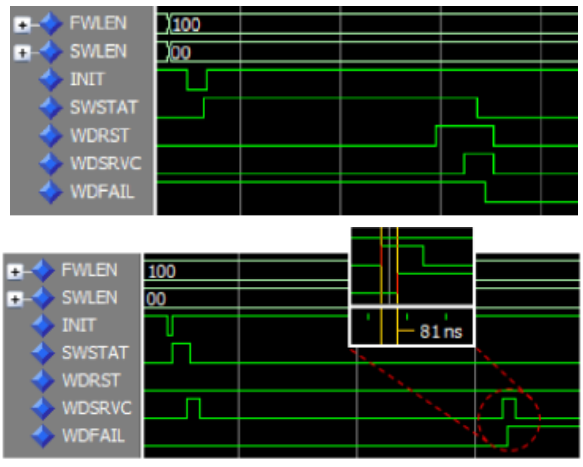


Fig. 9. Declaration of watchdog fail for an improper service operation

A. DesignVerificationinFPGA

The design has been synthesisedand implementedon a MicrosemiProASIC3E series Flash based FPGA. A Flash basedFPGAdevicewaschosenforitsgreaterim munityto

while yet enabling in-system reprogrammability, and handling single event upsets (SEUs)[18].Only 1% of the capacity of the chosen device was used by the implementation, which utilised 648 logic elements, which is comparable to a 3-input LUT. There is a little more intricacy in the design compared to the work in [11]. In this case, the authors kept the watchdog's architecture basic and utilised the down counter's expiration to show that the watchdog failed. In addition, they demonstrated how to create such many WDTs in a single FPGA.However, unlike current systems, the one suggested in this study has a windowed architecture and includes many defect detection capabilities. It is also possible to expand the architecture such that a single FPGA may house several watchdog timers. Concerning setup possibilities and fault detection characteristics, the suggested architecture outshines industry-standard microprocessor supervisory circuits like MAX693/MAX6323/TPS381X. A 32-bit NXP microcontroller-equipped real-time safety-critical embedded system has shown the design's implementation. System needs informed the selection of the watchdog window configurations. For the purpose of design validation, a software-based fault injection approach was used. This technique allows one to change the system state by modifying the software running on the system [19]. When the CPU neglects to service the watchdog timer, it usually indicates a hardware problem. Faulty memory readings or a software flaw might be the cause of this.Overloading, intermittent failures, or transitory errors might cause the CPU to mistime the maintenance of the watchdog.The CPU servicing the watchdog too often is another example.These real hardware failures served as the basis for the creation of fault injection models.We were able to enable fault injection in the programme by inserting sufficient instructions. Invoking them and simulating different failure situations was done by raising a hardware exception to the CPU during runtime.

The watchdog service activity was updated to skip during programme execution, enabling the frame window to expire and raise the fail flag for the watchdog. Additionally, the programme was designed to switch the WDT outside of the service window, which resulted in a rapid increase in the watchdog fail output. In order to satisfy the watchdog two times in a row, an alternate scenario was devised. The watchdog and asserteditsfail flag also saw this right away.After making the adjustments shown in Figure 5, we were able to see that the design performed as planned.The INIT input to the watchdog was finally failed by introducing a hardware-based fault injection. As a consequence, the watchdog failed since the service window could not start, which led to the frame window expiring.Each of these instances demonstrates that the suggested watchdog timer accurately identified the failure mode and recorded it in its configuration register.Additionally, the system's reset occurred when the watchdog asserted its RSTOUT output three milliseconds after the reset counter had been activated.

IV. CONCLUSION

The architecture and design of an enhanced window watchdog timer, as well as its implementation in

FPGA. The CPU is not involved in the operation of the watchdog timer, which allows for application-specific parameter adjustment.In order to identify abnormal software modes early on, the watchdog incorporates many defect detection mechanisms.It can detect the kind of failure and record it, which is useful for troubleshooting. When the watchdog timer detects a failure, it gives the programme enough time to save the debug information before starting the reset. The design may be made more versatile and reusable by implementing it in FPGA. With little overhead, HDL-based designs may be implemented on a variety of FPGA devices, regardless of manufacturer.By making little adjustments to the HDL, the same design may be adapted to many processors and applications. Also, the problem of component obsolescence in embedded systems with a lengthy life cycle may be solved by realising the design in FPGA. There is little hardware resource consumption and the implementation is simple. Using fault injection methods, the suggested design was tested in real-time safety-critical embedded hardware and successfully handled a variety of problems. It handles errors in this way.

REFERENCES

[1] S. N. Chau, L. Alkalai, A. T. Tai, and J. B. Burt, "Design of a fault- tolerant COTS-basedbus architecture," IEEE TransactionsonReliabil-ity,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,"Areviewofwatchdogarchitecturesandtheir application to Cubesats," Apr. 2010.

[4] A. Mahmood and E. J. McCluskey,"Concurrent error detection using watchdog processors - a survey," IEEE TransactionsonComputers, vol. 37,no.2,pp.160–174, Feb. 1988.

[5] B. Straka, "Implementing a microcontrollerwatchdog with afield- programmable gate array (FPGA)," Apr. 2013.

[6] J. Ganssle, "Great watchdogs," V-1.2, The Ganssle Group,updated January 2004, 2004.

[7] E. Schlaepfer,"Comparison of internaland external watchdog timers application note," Maxim Integrated Products, 2008.

[8] P. Garcia, K. Compton, M. Schulte, E. Blem, and W. Fu, "An overview of reconfigurablehardware in embedded systems,"EURASIP Journal on Embedded Systems, vol. 2006, no. 1, pp. 13–13, Jan. 2006.

[9] 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.

[10] A.M.El-Attar and G.Fahmy, "An improved watchdogtimer to enhance imaging system reliabilityin the presenceof softerrors,"inSignalProcessingandInformation Technology, 2007IEEEInternationalSymposiumon.IEEE, Dec. 2007, pp. 1100–1104.

[11] M.Pohronska´ andT. Krajcˇovicˇ,"FPGA implementationofmultiplehardwarewatchdogtimersfor enhancingreal-timesystemssecurity,"inEUROCON-InternationalConference on Computer as a Tool (EURO-CON), 2011 IEEE.IEEE, Apr 2011 pg 1-4

[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 usingfpgas,"IEEETransactionsonIndustrialElectronics,vol. 58, no. 3, pp 814-821, 2011

[13] H.AmerandA.Sobeih,"Increasingthereliabilityofthe Motorola MC68HC11 in the presence of temporary failures," in Electrotechnical Conference, 2002. MELECON 2002. 11th Mediterranean.IEEE, May 2002, ph no. 231-234

[14] A.M.El-AttarandG.Fahmy, "Astudyoffaultcoverage of standard and windowed watchdog timers," in Signal Processing and Communications, 2007. ICSPC 2007. IEEE International Conference on.IEEE, Nov. 2007, pg no: 325-328

[15] M. Barr, "Introduction to watchdog timers," Embedded Systems Design, 2001

[16] N. Murphy, "Watchdogstimers,"Embedded Systems Programming, p. 112, 2000

[17] F. Afonso, C. A. Silva, A. Tavares, and S. Montenegro, "Application-level fault tolerance inreal-time embedded systems," inIndustrial EmbeddedSystems,2008.SIES 2008.InternationalSymposiumon. IEEE, Jun. 2008, pp 126-133

[18] M. Wirthlin, "High-reliability fpga-based systems: Space,high-energyphysics,andbeyond,"Proceedingsofthe IEEE, vol. 103, no. 3, pp 379-389, 2015

[19] H. Ziade, R. A. Ayoubi, R. Velazco et al., "A survey on fault injection techniques," The International Arab Journal of InformationTechnology, vol.1,no.2, pp 171-186, jul 2004.