FPGA IMPLEMENTATION OF AN IMPROVED WATCHDOG TIMER FOR SAFETY

Neelam Srikanth[1] , Dr S Kishore Reddy[2] , Dr G Chandrashekar Reddy[3] and E Nagesh

[1]M.Tech Student, ECE, VLSI System Design, Avanthi Institute of Engg. & Tech., Hyderabad, India.

[2]Associate Professor, HOD, ECE, VLSI System Design, Avanthi Institute of Engg. & Tech., India.

[3]Assistant Professor, ECE, VLSI System Design, Avanthi Institute of Engg. & Tech., Hyderabad, India.

[4]Assistant Professor, ECE, VLSI System Design, Avanthi Institute of Engg. & Tech., Hyderabad, India.

Abstract: *When it comes to embedded systems that are used in mission-critical applications, extreme dependability is absolutely necessary. By using external watchdog clocks, these kinds of systems are able to automatically manage and recover from issues that are associated with operational time. In terms of functionality, the majority of the external watchdog clocks that are now available on the market depend on additional circuitry to adjust the lengthsof their timeouts. Detailed information on an improved customisable watchdog timer that is suited for use in applications that are mission-critical is provided in this research. The fact that the watchdog integrates a large number of fault detection algorithms contributes to the increased dependability of the watchdog. Due to the fact that its capabilities and operations are very extensive, it may be used to monitor the actions of any processor-based real-time system. Another topic that is discussed in this article is a Field Programmable Gate Array (FPGA) in the construction of the recommended watchdog timer. Because of this, the overall cost of the system is decreased, and the design is made more easily adaptable to a variety of applications. The first step in determining how effective the recommended watchdog timer is in detecting and responding to problems is performed by conducting an analysis of the outcomes of the simulation. There is the possibility of doing real-time hardware validation of the design by injecting faults into the code while the processor is running.*

Keywords:*FPGA, Watchdog, Validation, Design, Injecting, Simulation.*

I. INTRODUCTION

A significant amount of independence is necessary for the majority of embedded systems. When a software freezes, it is not practical to wait for someone to restart it since this is not an option in most situations. When it comes to some embedded designs, such as space probes, human operators simply do not have the requisite access rights. In the case that the software on such systems encounters a hang, the system will become unusable. On the other hand, there are situations in which the reset speed of a human operator is not adequate to meet the uptime requirements of the product. A watchdog timer is an example of a piece of hardware that may be used automatically to reset the central processing unit (CPU) in the event that software anomalies occur. The operation of a watchdog timer may be

summarised as follows: it begins by counting down from a beginning number until it reaches zero. It is the embedded software that is responsible for initialising and restarting the counter at certain intervals. If the counter ever gets close to zero before the processor is restarted, the software is regarded to be flawed, and the reset signal of the CPU is activated. The central processing unit (CPU) and any embedded software that it is now running will be restarted, just as if someone had cycled the power. Some individuals refer to the action of resetting the counter on a watchdog timer as "kicking the dog." The best visual example would be a man being attacked by a dog that is aggressive. By kicking the dog, he can stop it from biting him endlessly and prevent it from doing so. However, to guarantee that the dog will not bite him, he must continue to kick the dog at certain intervals. When the watchdog timer is not restarted on a regular basis, the application runs the risk of restarting itself, which is another vulnerability. The watchdog timers will trigger control systems to enter a safety mode whenever a problem is identified. In this mode, motors, high-voltage electrical outputs, and any other subsystems that might potentially cause damage are disabled until the issue is resolved. In the event that a watchdog timer equipped with an x-bit counter is put in a system that is using a y MHz clock signal, the system will shut down after a certain length of timehas elapsed unless the timer is reset. In our research, we investigated and created watch dog timers for use in automated teller machines (ATMs), which is one of the many important applications for these devices.

1.1 Describing Structure

The inputs and/or outputs of a digital electronic system are present in each individual module. The electrical values that are shown on the outputs will be influenced by any particular set of input values when they are applied. This sort of digital system viewpoint is shown in Figure 1-1(a), which you may look at. The F module receives its inputs from A and B, and the output is denoted by the letter Y.In Volatile Hardware Description Language (VHDL), the inputs and outputs of module F are referred to as ports, whereas the unit itself is referred to as a design entity. One method of representing the function of a module is via the composition of sub-modules that make up the module. There are signals that connect the ports of the various instances of an entity, and each sub-module represents an instance of that entity. Figure 1-1(b) illustrates one of the potential compositions of entity F, which includes several occurrences of G, H, and I. When describing this sort of information, structural descriptions are the ones that are used. It is important to keep in mind that entities G, H, and I could potentially have a potential structural description.

1.2 Describing Behaviour

In many cases, it is not suitable to provide a structural description of a single module. As an example, take into consideration a module that is located at the very bottom of the hierarchy of another structural description. If you are developing a system that makes use of IC packages that you have purchased from an IC shop, you should not describe the internal structure of an integrated circuit. Under these circumstances, it is essential to provide an explanation of the purpose of the module, regardless of the structure of the module itself. The kind of descriptions that are used for this form of description are functional or behavioural descriptions. Taking into consideration the

exclusive-or function as the function of entity F in Figure1-1(a) will allow you to see this in action. As a consequence of this, the Boolean function might be helpful in providing an explanation for the behaviour of F.When it comes to explaining increasingly complex behaviours, a function of inputs alone is not sufficient. In the same way, the outputs of feedback systems are reliant on the passage of time. Through the implementation of the specification of behaviour as an executable program, VHDL is able to remove this problem.

II. WATCHDOG TIMER

In contemporary uses of microcontrollers, electromagnetic interference (EMI) and electrical noise are both things that are often encountered. When dealing with situations like these, it is beneficial to have system resources that helpensure that the system is running correctly. In many different kinds of systems, the incorporation of a watchdog timer is a common approach for verifying that the system is operating correctly. In essence, a watchdog timer is nothing more than a time measuring device that is used in conjunction with or as an integral component of a microprocessor. It has the capability to reset the microprocessor. If the system is well-designed, the watchdog will cause a reset to occur if the microprocessor is not functioning as it should, which will resolve the issue when it occurs. After a certain length of time has elapsed, the watchdog timer will often reset the central processing unit. In an ideal scenario, the central processing unit would resume the watchdog before the interval came to a close. After the program has been restarted, the watchdog will begin timing at a new interval that you choose. A watchdog timer that is external to the system has been included into the design of several systems. However, if you upgrade to the DS80C320, you won't need that additional piece of hardware. By itself, the watchdog timer that is integrated into theDS80C320 is very capable of performing its functions. The operation of a watchdog timer is described in this application note, which also provides instructions on how to utilise the timer. The more complex watchdog timers would attempt to store debug information onto a permanent medium. This information serves as data that can be used to identify and resolve the problem that caused the error. If the first watchdog timer fails to signal that it has completed saving the data within a certain amount of time, the system will reset regardless of whether or not the datawas stored. This is because the system was designed to do so. This is avoided by using a second watchdog timer thatis more straightforward. It is common practice for embedded systems to make use of watchdog timers, which are specialist timers that are often included into microcontrollers.

The watchdog timers will trigger control systems to enter a safety mode whenever a problem is identified. In this mode, motors, high-voltage electrical outputs, and any other subsystems that might potentially cause damage are disabled until the issue is resolved. In the event that a watchdog timer equipped with an x-bit counter is put in a system that is using a y MHz clock signal, the system will shut down after a certain length of time has elapsed unlessthe timer is reset.

2.1 General Use of Watchdog Timer

As will be seen in the next section, the primary function of a watchdog timer is to serve as a system monitor. Through a watchdog timer, it is feasible to build systems that are capable of providing good detection and correction capabilities for microprocessors that are operating outside of their control. Systems that make use of watchdog timers are advantageous in terms of their ability to detect bit errors. Possible causes of transient bit errors include soft memory failures and electromagnetic discharges into memory devices and their interfaces. Both of thesetypes of failures may occur simultaneously. Transient bit polarity flipping may occur as a result of these while datais entering or exiting the central processing unit (CPU). When anything like this occurs when the microprocessor is obtaining information about the program, it will begin to execute erroneous code.

2.2 The Watchdog as a System Supervisor

It is common practice watchdog timer included inside the High-Speed Micro as a system supervisor. Despite the fact that system supervisor is the most common application, this article will also discuss a great deal of different applications. During the period that the system supervisor mode is active, the central processing unit will restart the timer at certain intervals, as was described before. As an alternative to restarting, the watchdog will time out, which will result in the central processing unit (CPU) being reset in the case that the processor can no longer maintain control. The watchdog timer of the High-Speed Micro is driven by the core system clock, which is something that is shared by a number of different business divisions. You are able to choose the amount of time that passes between timeouts by altering the output of the division. Once the timeout has been reached, the interrupt flag is set to enablea reset of 512 clocks, provided that the timeout is enabled.

As seen, there is a wide range of timeout intervals that may be used. Before deciding on the interval, it is important to take into account a number of different elements. First and foremost, the objective is to choose a time period that represents the maximum amount of time that the processor may be left unattended. A timeout interval that is smaller than 500 milliseconds would be considered optimal for a system that directs the position of a robotic arm every 500 milliseconds. When the delay period is shortened, there is a greater possibility that the arm will only receive one wrong command. With regard to the adjustment of the watchdog timeout duration, another significant factor to take into account is the location within the system software where the restart instructions are located. This might be a fairly complicated issue, depending on the kind of software that is installed on the PC. To achieve the best possible configuration, there would be just one location inside the main loop of the system software where the watchdog timer may be restarted. The amount of time required to finish one cycle of the main program loop is directly proportional to the need for the timeout interval that must be used. It is possible to put the aforementioned method into practice, provided that the software flow of the system is sufficiently linear. It is not possible to apply this strategy to all programs because of the complexity and non-linearity of the flows of such systems.

2.3 The Watchdog Timer as an Interval Measuring Device

The watchdog timer included into the High-Speed Micro serves a role that is considerably distinct from that of a traditional interval timer. In this instance, we are activating the interrupt by employing the Enable Watchdog Timer Interrupt (EWDI=EIE.4) bit; however, we are not enabling the reset. If the global interrupt enable bit (EA=IE.7) is set at the time that the timeout occurs, the Watchdog Timer will set the WDIF bit (WDCON.3) and cause an interrupt to be triggered. The origin of the interrupt will be shown on the Watchdog Interrupt Flag, which must be cleared by the program. The data shown in the table above demonstrates that a crystal operating at 25 MHz is able togenerate intervals that span from 5.26 milliseconds to 2.68 seconds. In the event that you use the standard 16-bit clocks, this interval would be far less than it should be. In the following, you will find an extra little program that demonstrates the features of the watchdog timer. This piece of code demonstrates the procedures that must be taken to initialise the 32 interrupts and watchdog timer in such a way that a timeout will cause an interrupt to be triggered. There is also a short service protocol given for cases of disruptions.

III. PROPOSED METHOD

Applications that need the highest degree of dependability are those in which the failure of the system might potentially cause damage to human beings. In order for these systems to operate in a secure manner, they need fault tolerance strategies that are able to deal with unforeseen circumstances. In addition to that, these systems need to be able to recover themselves They have a crash. The ability of these fault tolerance approaches to identify when problems arise is essential for the management of defects and the reduction of system downtime to the greatest extent possible [1]. One method of fault tolerance is the implementation of system redundancy. Having several copies of the system's essential components contributes to an improvement in the overall reliability of the technology [2]. On the other hand, this increased system dependability may be achieved by increasing the complexity of both the hardware and the software, depending on the architecture. The watchdog is a mechanism thatis both low-cost and high-performance, and it is used in the creation of fault-tolerant systems to discover and find solutions to issues that are connected to operating time [3]. The term "watchdog timer" (WDT) refers to a piece of hardware that monitors the operations of the system and, in the event that it identifies a problem, initiates certain actions [4]. Typically, it is a timer circuit that requires frequent resets from the central processing unit (CPU). The expiry of the WDT is an extra warning sign that signifies that the system being monitored is not operating as it should [5]. The central processing unit is unable to reset the watchdog, the system will either restart or become in a known good situation from which it may recover, so preventing any more damage from occurring. A central processing unit (CPU) may be paired with either an on-chip or an external watchdog. An internal watchdog is not a dependable choice, despite the fact that it simplifies the hardware implementation and reduces the cost of the hardware. At the time that the program is running, it is under the program's control, and runaway code has the abilityto deactivate it [3]. Due to the fact that the watchdog is connected to the timeline of the CPU, it will be unable to identify any hardware issues The crystal fails [6]. When the reliability of an embedded system is of the utmost importance, external watchdogs are absolutely necessary. With an external watchdog, the clock is not shared with the central processing unit (CPU), and it functions independently. As a consequence of this, fault-tolerant system

topologies are far more robust, operating beyond the limitations imposed by internal watchdogs [7]. A particularkind of standalone watchdog timer microchip is less general than others since it only has timeout periods that have been preset. You have the ability to alter the timeout lengths by using a different group of devices that need additional external circuitry.

Although this strategy is feasible, it contributes to an increase in the overall cost of the system and makes the hardware more complicated. Through the realisation of the watchdog capability included inside a Field Programmable Gate Array (FPGA), it may be possible to maintain some degree of control over the ever-increasing expense and complexity of external watchdogs. There are a variety of FPGA chips that are used by many modern embedded systems to provide the functionality that is intended for the system [8]. The incorporation of the watchdogtimer into an FPGA has the potential to provide a system that is both efficient and dependable. A customised concurrent watchdog processor was researched by Giaconia et al. [9] for the purpose of real-time control systems. They looked at the potential of building the processor on FPGA. In place of a central processing unit timer, the design went through a simple program flow check and verified that a few variables were within acceptableparameters. In accordance with the recommendations made by El-Attar et al. [10], a sequenced watchdog timer examines time registers to determine whether or not a malfunction takes place. The fault detection capabilities that were provided, on the other hand, were limited, and there were not a lot of different configuration options to choose from. Despite the fact that they made the watchdog's design as simple as possible, the authors of [11] described the principles of a watchdog timer system that is based on FPGAs and various hardware components.

Within the scope of this study, we describe a revised windowed watchdog timer design as well as its FPGA implementation. It is possible to realise the idea in field-programmable gate arrays (FPGAs) with just minor modifications to the associated code in hardware description language (HDL) [8]. This would allow the same watchdog hardware to be connected to multiple processors and systems. Additionally, it makes it feasible to handle several watchdog timers for architectures that have many cores. When it comes to safety-critical embedded systems, the watchdog timer that was recommended is effective when it comes to increasing system dependability via the utilisation of redundant channels. The issue of component obsolescence is a problem that affects a great number of embedded systems, especially those that are used in aerospace and military applications. With the WDT being designed as a reusable IP core, it is possible to find a solution to this problem [12]. In this paper, the architecture of the recommended watchdog timer, as well as its fault detection capabilities and FPGA
implementation, are extensively discussed.

3.1 The Architecture of the Proposed Watchdog Timer

The capacity to recognise any abnormal software states and return the system to a specified state is an essential feature of a surveillance mechanism that is capable of performing its function effectively. The device ought to be equipped with an independent clock and should be able to conduct a hardware reset to all of the peripherals in the

event that a timeout occurs [3]. For its functions, the watchdog timer that is discussed in this article operates independently from the central processing unit (CPU) and makes use of a separate clock. It is possible for the program to determine the window lengths during the initialisation process thanks to the design's use of a windowed watchdog implementation. When the watchdog timer reaches its expiry point, a fail flag is raised, and a reset is automatically triggered when a specific amount of time has passed since the flag was raised.

The application could make advantage of the time that has passed to store important debugging information to a medium that is not volatile. It is possible for a typical watchdog timer to identify problems that are occurring inside the system, such as the freezing of the system that is brought on by endless loops in the execution of code. The most significant disadvantage of this watchdog, on the other hand, is that The system gets trapped in a fault state in which it constantly resets the timer, the issue situation would not be recognised. A normal watchdog timer is able to recognise slow mistakes, but it is unable to identify rapid faults that occur inside the duration of the watchdog timer [13]. In essence, a watchdog timer is able to identify slow errors. In spite of this, a design that incorporates windows is likely to be able to successfully handle this circumstance. To avoid a timeout, the watchdog is programmed tohave a certain amount of time during which it must be reset completely. This feature provides protections against both excessive speed and inadequate speed in systems [14], which ultimately results in an increase in the scope of error detection.

A. I/O Interface and Configuration

A representation of the input-output (I/O) interface of the recommended watchdog timer may be seen in Figure 1. Watchdogs have the capability of producing two different outputs: a watchdog failure signal, also known as WDFAIL, and a reset signal, also known as RSTOUT. The SYSRESET input is low, the WDFAIL and RSTOUT outputs will continue to be in an asserted and deasserted state, respectively inside the system. In addition, the design incorporates a configuration register, and the bit fields of this register are provided in the same manner as shown in the illustration. Adjustments may be made to the settings of the watchdog, and information about the current state can be acquired from the register. Using the WDRST and WDSRVC fields, respectively, you are able to reset the watchdog and service it. To accurately represent any changes that have occurred to the INIT input and WDFAIL output states, the configuration register is being automatically updated. The FLSTAT field is responsible for recording the failure mode of a watchdog, whereas the SWSTAT field is responsible for storing the current state of the service window. On the watchdog timer, the ENABLE and RD/WR control inputs allow you to read and write tothe configuration register. You may also read from the register. Both the address bus (ABUS) and the data bus (DBUS) signals are visible in the graphic shown here.

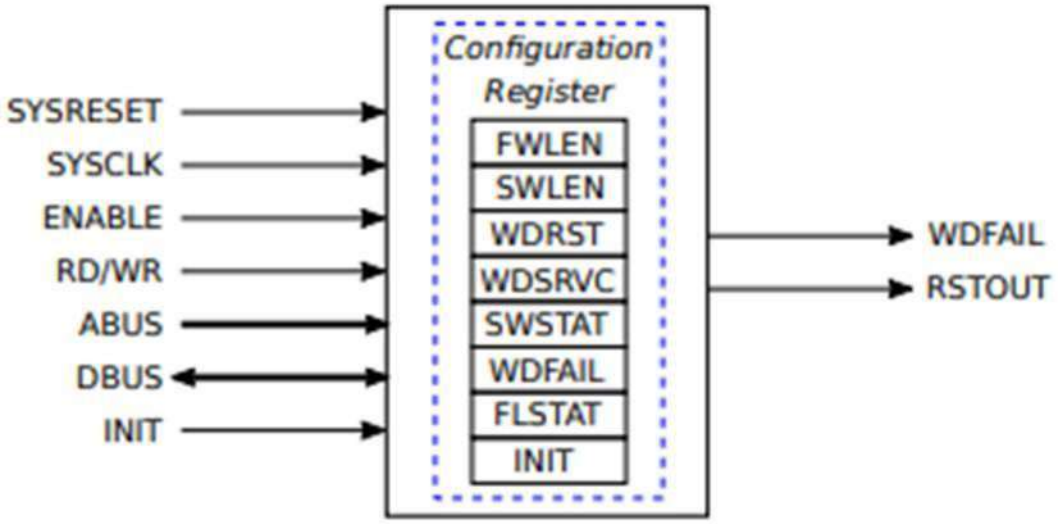


Figure 3.1 illustrates the configuration register as well as the input/output interface of the watchdog timer.

An example of the windowed watchdog idea that has been offered includes both a service window and a frame window. As opposed to the frame window, the service window will have a much shorter duration than the frame window. By writing to the bit fields SWLEN and FWLEN, the software is able to program the length of the two windows in the configuration register once the power is turned on. This is accomplished inside the configuration register. Following the window periods that have been established after power-up, the ability to make changes to the settings is blocked by design. The software feels the need to write to the configuration register once again, it will be required to go through a detailed unlock procedure. A consequence of this is that runaway code is unable to accidentally change the settings of the watchdog window. The INIT command is entered into the watchdog timer to initiate the process of initialising the service window. The service window will be activated if there is a transition from high to low on this input, provided that the fail flag (WDFAIL) is not currently active. It is necessary for the processor to perform maintenance on the watchdog inside the service window to prevent the watchdog from clocking out. The field known as watchdog service (WDSRVC) in the configuration register is used for the purpose of servicing the watchdog timer. In the event that this component inside the service window has an upward edge, the service window will be rendered closed, and the frame window will be initiated immediately. Through the frame window, the intervals between servicing the watchdog are specified. Performing maintenance on the watchdog once per cycle is a standard procedure, and the duration of this window should be maintained at a length that is considerably longer than the main loop of the embedded control system [15]. There are a variety of methods in which you may send the INIT signal to the watchdog timer. There is a method that involves performing certain sanity checks prior to terminating the main loop, and then thereafter activating the INIT signal [16]. An external interval timer may be used to avoid the central processing unit (CPU) from interfering with the creation of the INIT signal. The length of the frame window that is advised in this situation is somewhat longer than the duration of the main loop's execution. When it comes to task scheduling in frames, this particular setup strategy is effective for 56 embedded systems.

B. Watchdog Timer Initialization

The watchdog is switched on or reset, the WDFAIL output will display a high assert, which is an indication that the watchdog is in a failed condition. It is necessary for the software to initiate the watchdog and guarantee that it will continue to function. As shown in Figure 2, the waveform that represents the initialisation and general operation of the watchdog reset is shown. The first thing that has to be done to get the watchdog to function properly is to change the value of the watchdog reset (WDRST) field in the configuration register from low to high. A de-assertion of the WDFAIL flag and a subsequent servicing of the watchdog inside the service window are both necessary steps to restore its functionality. Bear in mind that another service window will start before the current frame window finishes since the frame window is kept larger than the system frame time. This is something that you should keep inmind. After the watchdog has been properly served once again, the frame window will be restored to its original state. As long as the frame window counts continue to function, the watchdog will not be able to identify any errors.

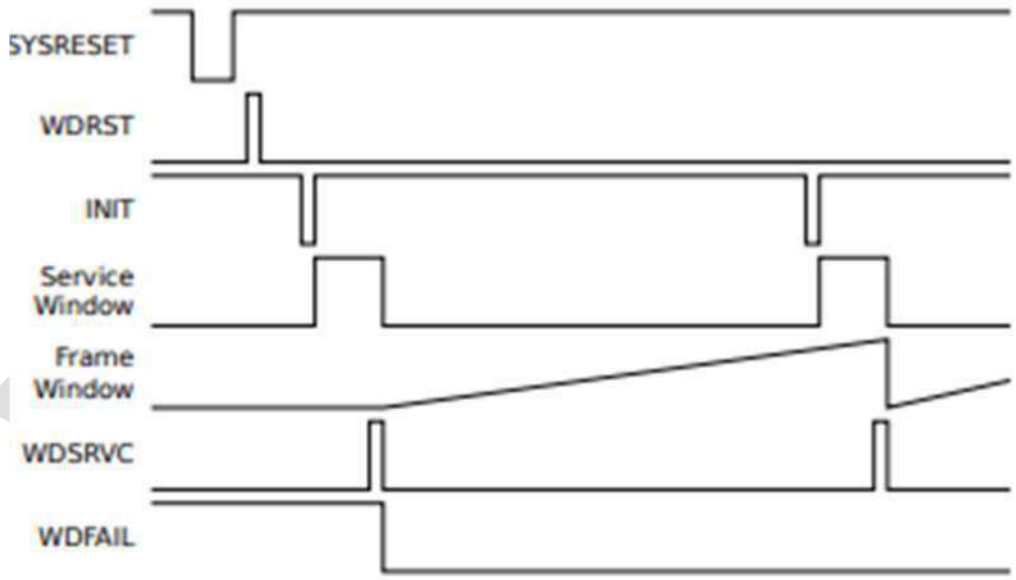


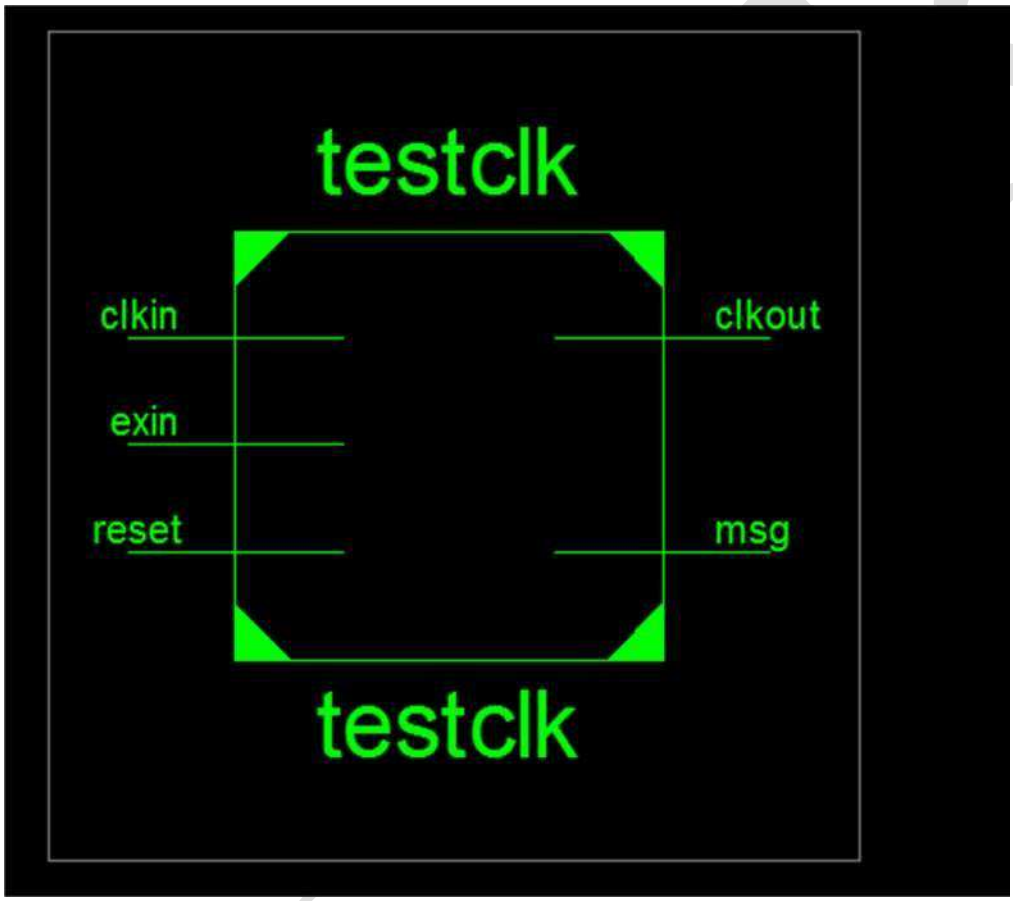
Fig.3.2. Startup and servicing of the watchdog timer

Critical embedded systems that operate in real time make use of redundancy and diversity to provide fault tolerance [17]. It turns out that in order for these systems to function properly, the watchdog fail signal should be asserted when the power is provided. It is possible fail status to indicate that a channel is unavailable for use in computationswhen it is not accessible. It is possible that the channel will be declared as online after the watchdog has been restored to a healthy state. In addition, the logic of redundancy management may cause the watchdog fail of a particular channel to occur if it is detected that the channel is operating badly throughout the course of ordinary

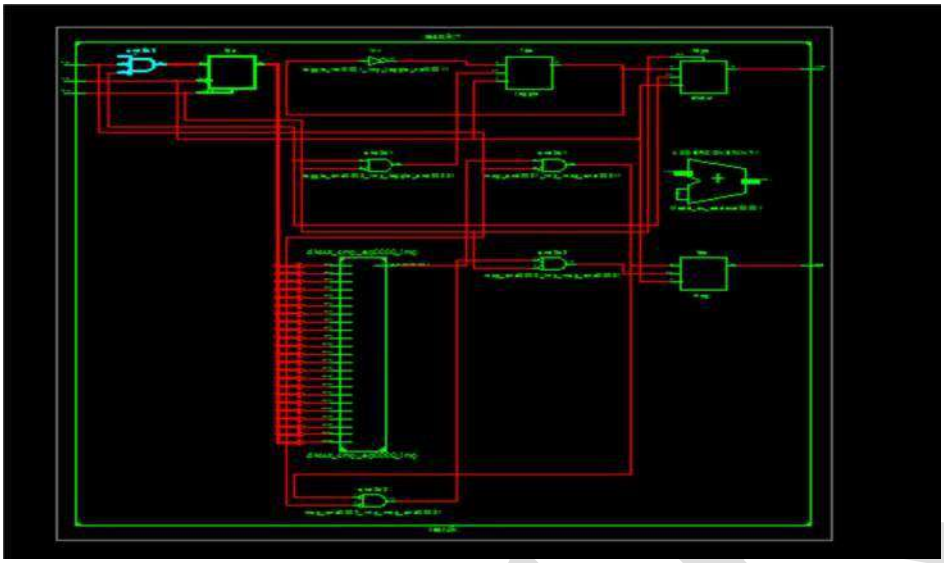
operations. If you do this, the faulty channel will be essentially removed from any computations that are performed in the future.

IV. SIMULATION RESULTS

- **RTL DIGARAM**



- **INTERNAL BLOCK**



- **Area**

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	26	9,312	1%	
Number of 4 input LUTs	33	9,312	1%	
Number of occupied Slices	30	4,656	1%	
Number of Slices containing only related logic	30	30	100%	
Number of Slices containing unrelated logic	0	30	0%	
Total Number of 4 input LUTs	55	9,312	1%	
Number used as logic	33			
Number used as a route-thru	22			
Number of bonded IOBs	5	232	2%	
Number of BUFGMUXs	1	24	4%	
Average Fanout of Non-Clock Nets	2.48			

Delay

```
    ----------------------------------------
    Total                      4.040ns (3.683ns logic, 0.357ns route)
                                       (91.2% logic, 8.8% route)

=========================================================================


Total REAL time to Xst completion: 8.00 secs
Total CPU time to Xst completion: 7.62 secs
```

- **Power**

Device	
Family	Spartan3e
Part	xc3s500e
Package	fg320
Temp Grade	Commercial
Process	Typical
Speed Grade	-5

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	0

Characterization	
PRODUCTION	v1.2,06-23-09

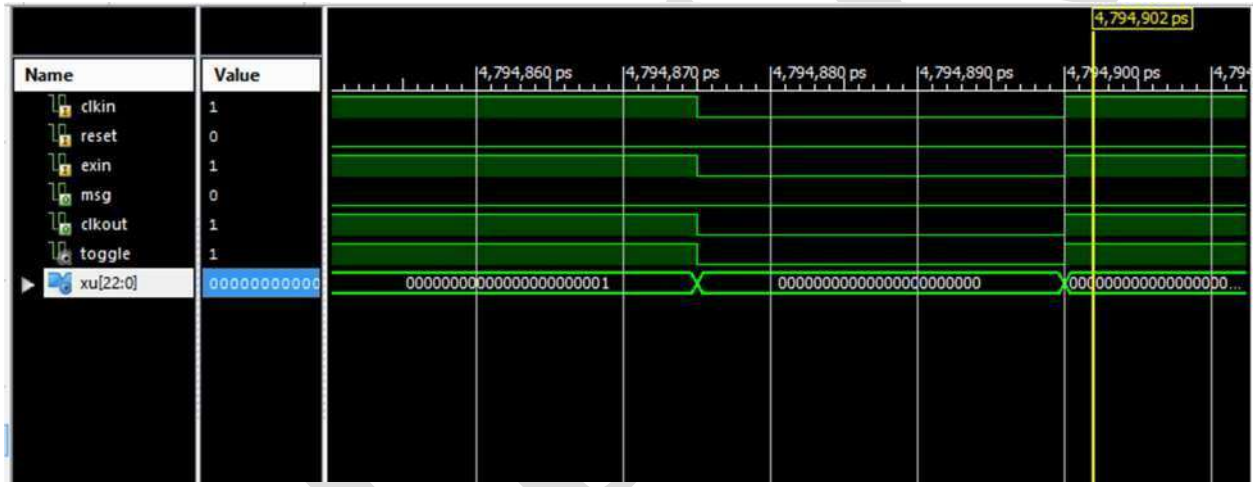
On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.000	1	---	---
Logic	0.000	55	9312	1
Signals	0.000	67	---	---
IOs	0.000	5	232	2
Leakage	0.081			
Total	0.081			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	26.1	82.9	27.1

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.200	0.026	0.000	0.026
Vccaux	2.500	0.018	0.000	0.018
Vcco25	2.500	0.002	0.000	0.002

	Total	Dynamic	Quiescent
Supply Power (W)	0.081	0.000	0.081

- **Simulation results**



V. CONCLUSION

The purpose of this research was to provide a full description of an improved windowed watchdog timer and its FPGA implementation. You are able to adjust the parameters of the watchdog timer to meet your requirements, andit functions independently from the central processing unit. A large number of defect detection techniques are included into the watchdog to facilitate the early identification of aberrant software modes. The fact that it is able to identify the kind of failure and record it is helpful for troubleshooting purposes. When a problem is identified, the watchdog timer provides the software with sufficient time to preserve the debug information before resetting the device. This occurs as soon as the fault is discovered. It is possible that the concept may be implemented in FPGA, which would make it more adaptable and reusable. There is no vendor-specificity associated with low-overhead HDL-based designs, which means they are interoperable with a broad range of FPGA devices. It is sufficient to make minor adjustments to the HDL to adapt the same design to a variety of processors and applications. The implementation of the concept in FPGA also provides a solution to the issue of component obsolescence that is

problematic for embedded systems with prolonged life cycles. In addition to the fact that the implementation requires a relatively low amount of hardware resources, its complexity is also quite low. After being tested in real- time safety-critical embedded hardware, the recommended design was able to effectively address a range of issues. This was accomplished via fault injection techniques.

REFERENCES

[1] S. N. Chau, L. Alkalai, A. T. Tai, and J. B. Burt, "Design of a faulttolerant 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] 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.

[5] B. Straka, "Implementing a microcontroller watchdog with a fieldprogrammable 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 internal and external watchdog timers application note," Maxim Integrated Products, 2008.

[8] 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.

[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 watchdog timer to enhance imaging system reliability in the presence of soft errors," in Signal Processing and Information Technology, 2007 IEEE International Symposium on. IEEE, Dec. 2007, pp. 1100–1104.

[11] M. Pohronska and T. Kraj ´ covi ˇ c, "FPGA implementation of multiple ˇ hardware watchdog timers for enhancing real-time systems security," in EUROCON-International Conference on Computer as a Tool (EUROCON), 2011 IEEE. IEEE, Apr. 2011, pp. 1–4.

[12] H. Guzman-Miranda, L. Sterpone, M. Violante, M. A. Aguirre, and M. Gutierrez-Rizo, "Coping with the obsolescence of safety- or missioncritical 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 Electrotechnical Conference, 2002. MELECON 2002. 11th Mediterranean. IEEE, May 2002, pp. 231–234.

[14] A. M. El-Attar and G. Fahmy, "A study of fault coverage of standard and windowed watchdog timers," in Signal Processing and Communications, 2007. ICSPC 2007. IEEE International Conference on. IEEE, Nov. 2007, pp. 325–328.

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

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

[17] F. Afonso, C. A. Silva, A. Tavares, and S. Montenegro, "Applicationlevel fault tolerance in real-time embedded systems," in Industrial Embedded Systems, 2008. SIES 2008. International Symposium on. IEEE, Jun. 2008, pp. 126–133.