VLSI design Of Improved Watchdog Timer for Safety-critical Applications

Kathi Ramababu (1)
rambabukathi9@gmail.com[1]

Krishnamohan.G (2)
gkmr09@gmail.com[2]

[1] M Tech, Dept of ECE, Abdul Kalam Institute of Technological Sciences.
[2] Assistant Professor, Dept of ECE, Abdul Kalam Institute of Technological Sciences.

ABSTRACT: Embedded devices used in safety-critical apps involve maximum reliability. In such schemes, external watchdog timers are used to automatically manage and recover errors associated with operating moment. Most of the external watchdog timers available use extra circuitry to modify their timeouts and only provide restricted feature characteristics. This document discusses the architecture and layout of an enhanced configurable watchdog timer that can be used in critical security apps. The watchdog incorporates several fault detection systems, which adds to its robustness. The features and activities are quite general and can be used to monitor the activities of any real-time processor-based scheme. This document also discusses the application in a Field Programmable Gate Array (FPGA) of the suggested watchdog timer. This enables for easy adaptation of the model to distinct apps while lowering the overall price of the scheme. First, by analyzing the simulation outcomes, the efficacy of the suggested watchdog timer to identify and react to faults is explored. In a real-time hardware, the layout is validated by injecting faults through the software while the processor is running and drawing results.

INDEX TERMS— deep learning, prediction process, accelerator, neural network

1. INTRODUCTION

For applications where a system crash could lead to human injury, highest reliability is required. Such systems should have fault tolerance mechanisms that account for the unexpected to ensure proper safety of operation. These systems should also be able to recover from a crash without any human assistance. These fault tolerance mechanisms detect when a fault occurs in order to handle the fault and to limit the system downtime [1]. One way to achieve fault tolerance is by implementing system redundancy. By using multiple copies of the critical components of the system, the overall system reliability is enhanced [2]. However, this improved system reliability is achieved through increased hardware and software complexity, depending on the type of architecture used. When developing a fault-tolerant system, one of the most cost effective ways of detecting and handling operation time related failures is the watchdog [3]. A watchdog timer (WDT) is a hardware subsystem that monitors the operations of the system and takes certain actions in the event of detecting a fault [4]. It typically consists of a timer circuit and the processor is required to periodically reset the timer. If the WDT expires, it is a secondary indication of some problem with the system under observation [5]. When the processor fails to reset the watchdog, a decision is made to restart the system or put the system into a known state from which it can recover, thus preventing further damages. A watchdog can be internal (on-chip) or external to the processor. Internal watchdog reduces the hardware complexity and cost, however, is not a robust solution. The software has control over it during runtime and a runaway code can disable the watchdog timer [3]. Moreover, since it is connected to the processor clock, a crystal failure will make the watchdog incapable of monitoring the hardware for faults

[6]. When the reliability of an embedded system is crucial, external watchdogs become unavoidable. An external watchdog runs independent of the processor and does not share its clock with the processor. This overcomes the limitations of internal watchdogs and leads to much more robust fault-tolerant system architectures [7]. A class of standalone watchdog timer microchips offers only fixed timeout periods, which make them less generic. Other set of devices allow adjusting the timeout periods by using additional external circuitry. Though useful, this method adds to the complexity of the hardware and increases the overall system cost. The increased cost and complexity of external watchdogs can be managed to a certain extend by realizing the watchdog functionality within a Field Programmable Gate Array (FPGA). Many of the modern embedded systems incorporate one or more FPGA devices to accomplish the desired system functionality [8]. Accommodating the watchdog timer within a FPGA can yield an efficient and robust solution. The work done by Giaconda et al. [9] considered the implementation of a custom concurrent watchdog processor in FPGA for real-time control systems. The design did not provide a timer for the processor; rather, it performed a reasonableness check on some variables and a basic program flow check. El-Attar et al. [10] proposed a sequenced watchdog timer that used time registers to determine whether or not a fault has occurred. However, it did not offer much configuration options and the fault detection features implemented were limited. In [11] the authors addressed the basic concepts of a multiple hardware watchdog timer system in FPGA, but kept the design of the watchdog simple. In this paper, we propose the design of an improved windowed watchdog timer and its implementation in FPGA. Realizing the design in FPGA means that the same watchdog hardware can be interfaced to different processors and systems, with only minor modifications of the associated Hardware description language (HDL) code [8]. It also allows for accommodating multiple watchdog timers for multicore architectures. The proposed watchdog timer is well suited for safety-critical embedded systems, where redundant channels are employed to enhance the system reliability.

Designing the WDT as a reusable IP core also addresses the component obsolescence issues faced by many embedded systems, especially those in the aerospace and military applications [12]. The paper describes the architecture of the proposed watchdog timer, the fault detection features, and its implementation in FPGA. The remainder of this paper is organized as follows. The following section introduces the architecture of the proposed watchdog timer. The fault detection mechanisms built into the watchdog is discussed in III. Section IV describes the implementation of the watchdog timer in FPGA. Simulation results and evaluation of the design in hardware are detailed in section V. Finally, section VI concludes this paper.

II. PROPOSED WATCHDOG TIMER ARCHITECTURE

An effective watchdog should be able to detect all abnormal software modes and bring the system back to a known state. It should have its own clock and should be capable of providing a hardware reset on timeout to all the peripherals [3]. The watchdog timer proposed in this paper operates independently of the processor and uses a dedicated clock for its functions.

The architecture follows a windowed watchdog implementation, where the window periods can be configured by the software during initialization. A fail flag is raised when the watchdog timer expires and after a fixed amount of time from raising the flag, a reset is triggered. The time in-between can be used by the software to store valuable debugging information to a non-volatile medium. A standard watchdog timer can catch problems in the system such as hanging because of endless

loops in code execution. However, the main disadvantage of this watchdog is that if the system enters a fault state in which it continually resets the timer, the error state will never be detected. In other words, a standard watchdog timer can detect slow faults, but cannot detect fast faults which occur within the watchdog timer period [13]. However, a windowed architecture can handle this properly.

Here the watchdog defines a small time window within which the watchdog must be reset in order to avoid a timeout. This provides protection against systems from running too fast and too slow [14], thus increasing the error recognition coverage. A. I/O Interface and Configuration Fig. 1 shows the input-output (I/O) interface of the proposed watchdog timer. The watchdog has two outputs, namely the watchdog fail output (WDFAIL) and the reset output (RSTOUT). When the SYSRESET input is low, the WDFAIL output remains asserted and the RSTOUT output stays deserted.

The design also consists of a configuration register with bit fields defined as in the figure. The register enables adjustments to the watchdog parameters and also provides status information. The WDRST and WDSRVC fields are used respectively for resetting and servicing the watchdog. The state of the INIT input and the WDFAIL output are automatically updated in the configuration register.

The SWSTAT field holds the state of the service window and the FLSTAT field logs the watchdog failure mode, if any. The control inputs to the watchdog timer, ENABLE and RD/WR, permit the read and write to the configuration register. The ABUS and DBUS signals in the figure indicate address bus and data bus, respectively.
The proposed windowed watchdog design constitutes a service window and a frame window. The service window duration will be much smaller compared to that of the frame window. The length of the two windows can be programmed by the software after power-up by writing to the bit fields, SWLEN and FWLEN, in the configuration register.

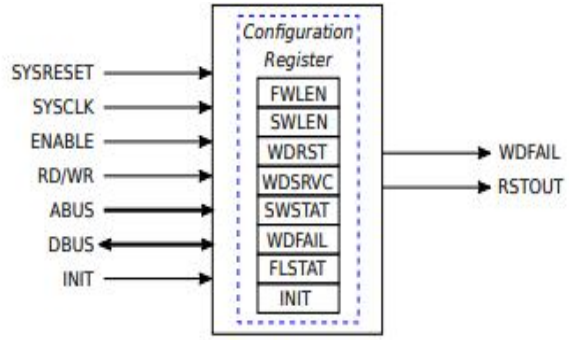


Fig. 1. Watchdog timer input-output interface and configuration register

Once the window periods are configured after power-up, modifying the values is disabled by design. If needed, the software will have to go through a stringent unlock procedure in order to be able to once again write to the configuration register. This prevents any accidental modification of the watchdog window parameters by a runaway code. The INIT input to the watchdog timer initializes the service window.

A high-to-low transition on this input will start the service window, provided the fail flag (WDFAIL) is not active. The processor is required to service the watchdog within the service window, in order to prevent a timeout. The watchdog timer is serviced using the watchdog service (WDSRVC) field in the configuration register. A rising edge on this bit inside the service window will immediately close the window and start the frame window.

The frame window defines how periodically the watchdog should be serviced. Typically, the duration of this window is kept slightly more than the main loop of the embedded control system and the watchdog is serviced once in every cycle [15]. The INIT signal to the watchdog timer can be driven in different ways. One way is to trigger the

INIT signal at the end of the main loop, after performing some sanity checks [16]. An external interval timer may be used to avoid any processor intervention in generating the INIT signal. The frame window in this case should be set for a length slightly more than the execution time of the main loop. This mode of configuration is 56 particularly suitable for embedded systems that schedule their tasks in frames.

B. Watchdog Timer Initialization

On power-up or reset the watchdog wakes up in a failed state, i.e., the WDFAIL output will be asserted high. It is the responsibility of the software to initialize the watchdog and keep it running. Fig. 2 illustrates the waveform for watchdog reset initialization and general operation. In order to bring the 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 inside the service window, will de-assert the WDFAIL flag and make it operational. Since the frame window is kept larger than the system frame time, another service window will start before the current frame window expires. When the watchdog is again properly serviced, the frame window will be reinitialized. As long as the frame window counters keep running, no failures will be flagged by the watchdog.

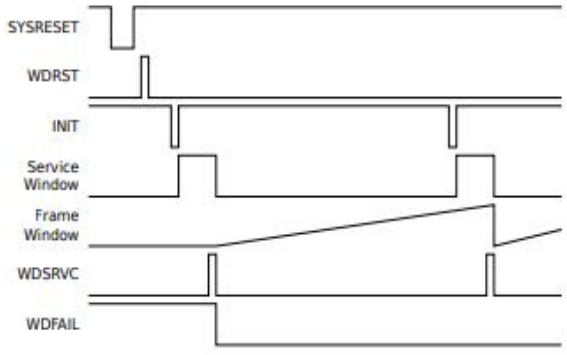


Fig. 2. Watchdog timer initialization and service operation

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. Once the watchdog is brought to a healthy state, 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 activate the watchdog fail of that channel. This can effectively withdraw the faulty channel from taking part in any further computations.

III. FAULT DETECTION FEATURES

Several fault detection mechanisms are built into the proposed 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 window the watchdog asserts its WDFAIL signal, indicating a failure. This failure mode is depicted in Fig. 3.

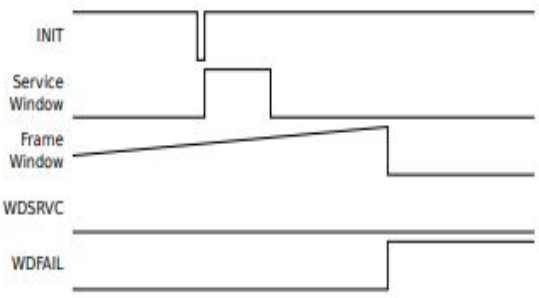


Fig. 3. Watchdog fail due to frame window expiry

A watchdog fail will occur when the software services the watchdog outside the service window, as shown in Fig. 4. It can be seen that the invalid service operation instantly terminates the frame window and asserts the WDFAIL signal. A favorable consequence of this feature is that two successive service operations will also lead to a watchdog fail. Here, the first service operation will immediately close the service window and the next one will invariably occur outside the window. This becomes equivalent to servicing the

watchdog outside the service window and leads to a watchdog failure.

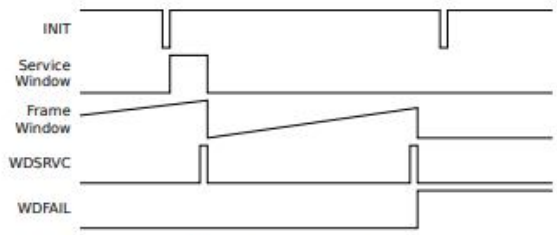


Fig. 4. Watchdog fail due to service outside the service window

Fig. 5 illustrates a scenario where the WDSRVC falling edge is occurring inside the service window. This is also considered as an illegal service operation and the watchdog fail signal is asserted. This implies that, after servicing the watchdog, the software is required to de-assert the WDSRVC signal before the start of the next service window. All of these fault detection mechanisms ensure that a software running haywire will not go undetected by the proposed watchdog timer

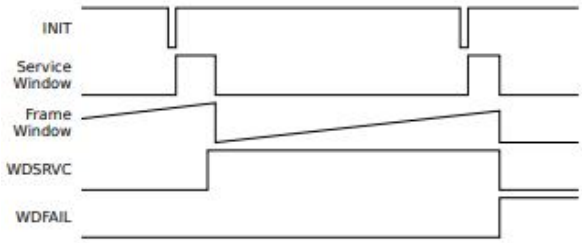


Fig. 5. Watchdog fail due to WDSRVC falling edge inside service window

The WDFAIL output from the watchdog timer can be used to activate a fail-safe state, or warn the processor of the fault by issuing a non-mask able interrupt (NMI) signal. After a predefined amount of time from asserting the WDFAIL output, the watchdog will assert its RSTOUT output. This signal can be tied to the reset pin of the processor, causing it to reset the embedded system automatically. The intervening time will give the software an opportunity to save information that can be valuable for debugging. In the event of a failure, the corresponding failure mode will be logged in the FLSTAT field in the watchdog configuration register. The software can attempt to save this information also to a non-volatile memory for debugging purposes.

IV. WATCHDOG TIMER IMPLEMENTATION IN FPGA

This section details the realization of the proposed watchdog timer in FPGA. The high-level diagram of the watchdog hardware is shown in Fig. 6. The design is clocked by its SYSCLK input, which is independent of the processor clock.

The possible sets of window lengths are arrived based on the application and hard-coded in the design. These values can be selected by writing to the appropriate bits in the configuration register - SWLEN for the service window and FWLEN for the frame window - after power-on. Once the values are selected, the window length configuration fields get locked automatically; i.e., writes to these bits are disabled. For the cases where the window lengths have to be modified again, a 16-bit unlock register is provided in the design.

In order to change the window lengths, the software will have to perform two successive writes to this register with data 0xAAAA and 0x5555. Subsequent to writing the first pattern the second one must be written within 10 µs, after which the software gets a 10 µs period to modify the length configuration fields. If these timings are not strictly met, writes to these bits will remain disabled. The service window is started when a high-to-low transition is detected on the INIT signal.

The service window uses a derived clock (SWCLK) that is much slower than the SYSCLK. The slower clock helps in reducing the number of comparators required, thus minimizing the resource utilization in FPGA. The service window has an offset up/down counter that are clocked by the SYSCLK, and a main counter that runs at SWCLK. The offset up counter finds the offset (Tof f set) between the INIT input and the next rising edge of the SWCLK. This is necessary as the INIT signal may be asynchronously driven

and can come at any time within the SWCLK period, Tswclk. The offset value is saved and the main counter is started, which then runs for (SWLEN - 1) times. Once the main counter expires, the offset down counter runs for a duration Tswclk−Tof f set. This counting procedure allows for a precise control over the window length.

The running status of the service window is also updated in the watchdog configuration register periodically. When the watchdog is correctly serviced, the counters in the service window stop immediately and the frame window starts. The frame window also uses a derived slower clock (FWCLK) for its operations. It has an offset up/down counter and a main counter with functionalities similar to that of the service window. The offset up counter here finds the offset between the termination of the service window and the next rising edge of the FWCLK.

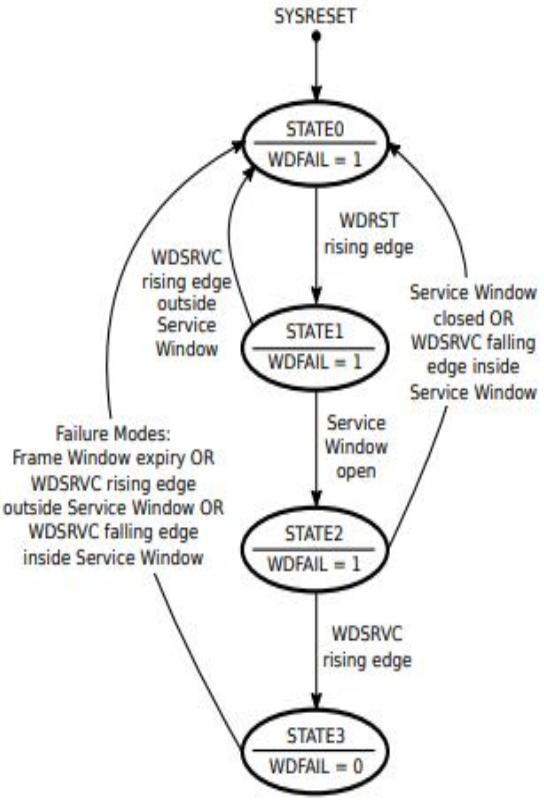


Fig. 7. Finite state machine design of watchdog fault detection logic

The main counter counts for (FWLEN - 1) times which is then followed by the offset down counter. The frame window counters reset when a watchdog service operation occurs within the next service window duration, before the frame window expires. A. Reset Initialization and Fault Detection The diagram in Fig. 7 shows the finite state machine (FSM) implementation of watchdog reset initialization and fault detection logics.

On power-up the WDFAIL output is asserted, indicating a watchdog failure. A rising edge on the WDRST bit prepares the watchdog timer for initialization. When the service window opens, a rising edge on the WDSRVC bit desserts the WDFAIL output and the window counters start running. However, if the watchdog is serviced incorrectly, the whole initialization process is discarded and the software will have to repeat the entire procedure.

The WDFAIL signal gets de-asserted only when the watchdog is properly initialized. While the watchdog is up and running if any of the failure modes described in section III occurs, the WDFAIL output is again asserted. The configuration register is updated with the failed status and the nature of the failure. Assertion of the watchdog fail also triggers a reset counter that runs for a predefined amount of time.

The duration of the counter can be determined by considering the amount of debug information that needs to be stored. On the expiry of the counter, the WDT asserts its RSTOUT output high. The reset counter will be nonfunctional during power-up and the RSTOUT output will be set to low at this point. When the watchdog is initialized for the first time, the counter gets automatically enabled.

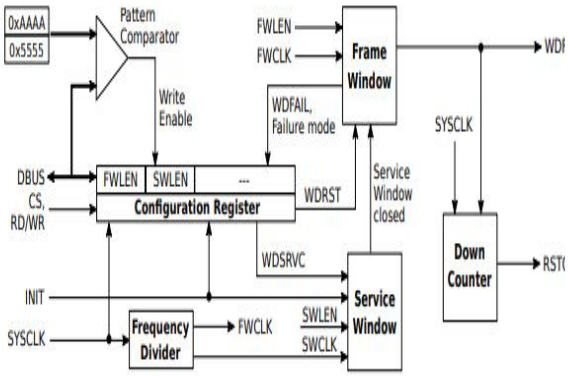
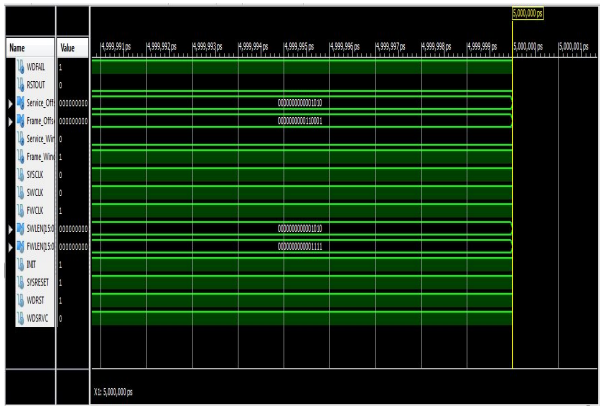


Fig. 6. Functional block diagram of the proposed watchdog timer

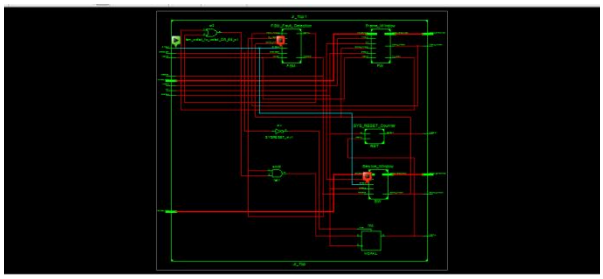
2. RESULTS

Various parts of the suggested scheme are stored in VERILOG HDL, displayed in I simulator and Xilinx ISE is the FPGA synthesis software tool.

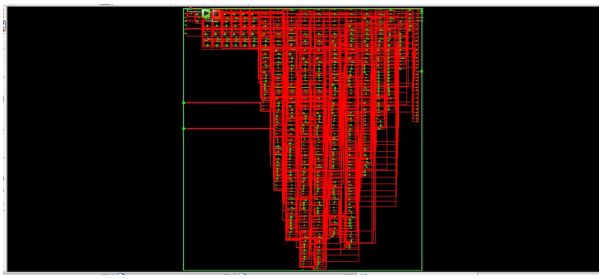
SIMULATION:



RTLSCHEMATIC:



TECHNOLOGYSCHEMATIC:



DESIGNSUMMARY:

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

TIMINGREPORT:

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

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

CONCLUSION:

This paper presented in detail the architecture and design of an improved windowed watchdog timer and its implementation in FPGA. The watchdog timer runs completely independent of the processor and permits adjusting the timer parameters according to the application. Several fault detection techniques are built into the watchdog for the early detection of erratic software modes. It has the capability to identify the failure type and log it, which can become valuable while debugging.

Upon detecting a failure, the watchdog timer also allows the software sufficient time for saving the debug information, before initiating a reset .Implementing the entire design in FPGA has the advantage of making it adaptable and reusable. HDL based designs are vendor-independent and can be used on different FPGA devices with low overhead. The same design can also be customized for different processors and applications with only minor HDL modifications. In addition, realizing the design in FPGA addresses the component obsolescence issues

presenting long life cycle embedded systems. The implementation has low complexity and takes up very less amount of hardware re-sources. The proposed design was tested in a real-time safety-critical embedded hardware using fault injection techniques and proved to be effective in handling various faults.

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 Reliabil-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, "A review of watchdog architectures and their applicationto Cubesats," Apr. 2010.

[4] A. Mahmood and E. J. McCluskey, "Concurrent error detection usingwatchdog 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 field-programmable gate array (FPGA)," Apr. 2013.

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

[7] E. Schlaepfer, "Comparison of internal and external watchdog timersapplication note,"Maxim Integrated Products, 2008.

[8] P. Garcia, K. Compton, M. Schulte, E. Blem, and W. Fu, "An overviewof reconfigurable hardware in embedded systems,"EURASIP Journalon Embedded Systems, vol. 2006, no. 1, pp. 13–13, Jan. 2006.

[9] G. C. Giaconia, A. Di Stefano, and G. Capponi, "FPGA-based concur-rent 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 toenhance imaging system reliability in the presence of soft errors," inSignal Processing and Information Technology, 2007 IEEE InternationalSymposium on. IEEE, Dec. 2007, pp. 1100–1104.

[11] M. Pohronsk´a and T. Kraj˘covi˘c, "FPGA implementation of multiplehardware watchdog timers for enhancing real-time systems security," inEUROCON-International Conference on Computer as a Tool (EURO-CON), 2011 IEEE. IEEE, Apr. 2011, pp. 1–4.

[12] H. Guzman-Miranda, L. Sterpone, M. Violante, M. A. Aguirre, andM. Gutierrez-Rizo, "Coping with the obsolescence of safety- or mission-critical embedded systems using fpgas,"IEEE Transactions on IndustrialElectronics, vol. 58, no. 3, pp. 814–821, 2011.

[13] H. Amer and A. Sobeih, "Increasing the reliability of the motorolaMC68HC11 in the presence of temporary failures," inElectrotechnicalConference, 2002. MELECON 2002. 11th Mediterranean. IEEE, May2002, pp. 231–234.

[14] A. M. El-Attar and G. Fahmy, "A study of fault coverage of standard andwindowed watchdog timers," inSignal 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, "Application-level fault tolerance in real-time embedded systems," inIndustrialEmbedded Systems, 2008. SIES 2008. International Symposium on.IEEE, Jun. 2008, pp. 126–133.

[18] M. Wirthlin, “High-reliability fpga-based systems: Space, high-energyphysics, and beyond,”Proceedings of the IEEE, vol. 103, no. 3, pp.379–389, Mar. 2015.

[19] H. Ziade, R. A. Ayoubi, R. Velazcoet al., “A survey on fault injectiontechniques,”The International Arab Journal of Information Technology,vol. 1, no. 2, pp. 171–186, Jul. 2004.

BIOGRAPHIES

G Krishna Mohan working as an assistant professor(Hod Ece dept)born in 17/09/1987 b.tech from sri ks Raju institute of science and technology from ece.M tech es & vlsi from malla Reddy engg college.from 2014.jntu hyderabad.

kathi Rambabu was born in kothagudem , Telangana , india . in 1993 . recevied B-Tech Degree in Elevtronics and Communication Engineering from JNTU hyderabad in 2015