International Journal of Research (IJR)
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Vol. 9 Issue 07
July 2022

Area Optimized VLSI Architecture of an Improved Using Watchdog Timer sklansky adder

Mamedi Saidheeraj[1], G. Dhanalakshmi[2], K. Rajesh[3], D. Srinivas [4]
[1] Research Scholar, Siddhartha Institute of Technology and Sciences Koremulla Road, Narapally, R.R. District, Telangana.
[2] Professor, Siddhartha Institute of Technology and Sciences Koremulla Road, Narapally, R.R. District, Telangana.
[3,4]Asistant Professor, Siddhartha Institute of Technology and Sciences Koremulla Road, Narapally, R.R. District, Telangana.

ABSTRACT

Embedded systems in safety-critical applications must be extremely reliable. Such systems utilize external watchdog clocks to manage and recover from operational time-related issues. The overwhelming majority of external watchdog timers need extra hardware to change their timeout durations and provide very limited capabilities. This article describes the structure and construction of a sophisticated configurable watchdog timer. Using the sklansky adder adder, which is suitable for safety-critical applications. The watchdog has many defect detection algorithms built in, which contributes to its resilience. Because of its generic capabilities and operations. You may use it to keep tabs on the functioning of any real-time system that relies on a CPU. In addition, the implementation of the recommended watchdog timer is explored. This makes the design adaptable to a number of applications and reduces the overall cost of the system. The simulation results are used to evaluate the error detection and correction capabilities of the suggested watchdog timer. Errors are introduced into the program while the processor is running to validate the design in real-time hardware, and conclusions are formed.

Index terms: FSM, sklansky adder, error detection and correction.

I. INTRODUCTION

A watchdog timer is an electrical circuit that detects a computer hardware failure or software error and takes remedial action. Because it offers a mechanism to automatically recover from transitory defects, It is essential to systems that are difficult or impossible to access physically. A watchdog timer is helpful in circumstances when a human operator would be too slow to react. Watchdog timers are found in a broad range of embedded and remote devices, from microwave ovens to Mars rovers. Every watchdog timer, no matter how basic or complex, must take two corrective actions. To begin, it must configure the computer's control outputs to safe levels, ensuring that potentially harmful components like motors and heaters do not endanger people or equipment.

This is a high-priority activity that must be taken as soon as a problem is discovered. The next step is to restore regular system functioning once the outputs have been adjusted to safe levels. This may be as easy as restarting the computer as if a human operator had clicked the reset button, or it might be a series of procedures that culminate in a computer restart.

Operation and Structure A watchdog timer does exactly what its name suggests: it keeps track of time. A circuit performs the timing function by producing a delayed output signal in response to an input trigger signal. The circuit may be constructed as an analogue delay circuit, which utilizes a

monostable multivibrator, or as a digital delay circuit, which controls the delay duration using a digital counter. Although many of the techniques addressed here also apply to analogue delay circuits, this application note concentrates on digital watchdog clocks.

A watchdog timer (or simply "watchdog") is a digital counter that counts from an initial value to a terminal value at a fixed-frequency clock-determined pace. Typically, the counter counts down from a starting number to zero, with the starting value configurable so that the software may control how long it takes to reach zero. When the counter hits zero, the watchdog "times out," asserting its timeout signal and stopping counting; this is known as a watchdog "event." External circuitry is attached to the timeout signal to conduct remedial action.

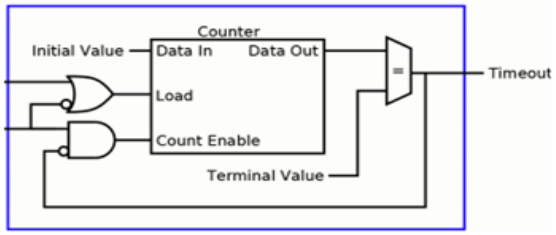

Fig 1: A basic digital watch dog timer

By putting the starting value into the counter, a software may restart the timer at any moment; this is known as "kicking" the watchdog. The watchdog is restarted by asserting its restart input for a brief period of time, commonly by writing to a watchdog "kick" port. The application software frequently kicks the timer to prevent it from approaching 0 during normal operation, usually as part of a control loop. The watchdog will timeout and take remedial action if a fault situation prevents the application from triggering the timer. Because the WDT is the final line of defense, it must be designed to account for each possible failure scenario. "What are the traits of an excellent watchdog?" one may wonder. The WDT has to be CPU-independent. The watchdog timer must work no matter what strange mode the CPU is in. Furthermore, nothing the CPU performs after activation should be able to deactivate or reprogram the watchdog. Otherwise, a malicious software may unintentionally deactivate this safeguard, making it ineffective.

With the exception of a hardware failure, the WDT must always bring the system back to life. This entails giving the CPU a hard reset. There is no certainty that any other technique will bring a wrecked CPU back to life. Instead of a reset, certain WDTs send a non-mask able interrupt. The concept is that the NMI's service procedure will be able to take a snapshot of the stack and record debugging data. Unfortunately, there's no reason to think that a CPU in an arbitrary malfunctioning state would react to any interrupt; there's a lot of processing to be done before the service routine is called.

Many processors will not start an interrupt service function if the stack pointer contains an odd number or unaligned addresses; in fact, they may enter a double-bus fault state, in which the CPU shuts down and only a hard reset will bring it back to life. However, the NMI method is intriguing. An NMI and a timer are two options that are sometimes employed. The timer then resets the CPU after a few milliseconds. If the NMI service procedure succeeds, it records debugging information, but the device is always brought back to life by a hard reset.

When the system manages risky hardware, it is vital that the watchdog, independent of a possibly crippled CPU, puts the system into a safe state. Because the reset may not operate if the processor has

been wrecked by a hardware problem, moving equipment, harmful radiation, and so on must be deactivated, parked, or otherwise disengaged. In today's systems, the peripherals are often sophisticated; in some situations, the I/O is even more complicated than the CPU.

These devices must be brought back to a known condition through the WDT reset routine. When code breaks, strange streams of data may be sent to the peripherals. If the peripherals are designed in such a manner that the CPU is not always able to put them into known right states, the WDT must do a hard reset. Finally, if at all feasible, leave debugging breadcrumbs behind. NMI/deferred-reset, as previously indicated, is an example.

Developers may save the stack and other key settings in a non-volatile memory section. Unfortunately, a reset wipes out all processor state information, but there is typically application-specific data, such as references to state machine tables, that might aid in troubleshooting. Save them before re-initializing them after a reset. Save the time of the reset if there is a real-time clock.

II. LITERATURE SURVEY

"Fault tolerant digital systems," by V. B. Prasad. Embedded systems must be very reliable in safety-critical applications. This study describes an updated windowed watchdog timer for safety-critical applications. The watchdog has many defect detection algorithms built in, which contributes to its resilience. The recommended watchdog has fewer temporal limitations than the current watchdog due to processor reliance. Simulation results are utilized to evaluate the proposed watchdog timer's error detection and response. As a consequence, the watchdog is deployed and validated in ATMs and space launch vehicles once it is built. The design is developed in Verilog and implemented on a Spartan 6 FPGA using Xilinx 14.5J. Benin go studied watchdog designs and their application to Cube sats. Safety-critical applications need trustworthy embedded systems. External watchdog clocks monitor and recover from time-related problems in such systems. Most external watchdog timers need additional hardware to modify timeouts and have restricted functionality.

This paper describes an updated programmable watchdog timer for safety-critical applications. The watchdog has many defect detection algorithms built in, which contributes to its resilience. Because of its generic capabilities and operations, It monitors processor-based real-time systems. This makes the design flexible to a variety of applications while also lowering the total cost of the system. The simulation results are used to assess the proposed watchdog timer's error detection and repair capabilities.

III. WINDOWED WATCHDOG TIMER

Any good monitor should be able to detect and undo any rogue program states. It must have its own clock and be able to reset all devices' hardware when timeout happens. The watchdog timer presented in this study is self-contained and runs on a separate clock from the CPU. The design uses a windowed watchdog implementation, in which the programme may set the window periods at startup. When the watchdog timer expires, a fail flag is raised, and a reset is triggered when a predetermined length of time has passed since the flag was raised. The programme may utilize the interval in between to save crucial debugging data to a non-volatile media.

A conventional watchdog timer may detect system issues such as the system stalling due to infinite loops in code execution.

If the system enters a fault situation in which the timer is continually reset, the problem status is never recognized by this watchdog. To put it another way, a normal watchdog timer may detect slow errors but not quick problems that occur within the timer interval. A windowed architecture, on the other hand, can deal with this effectively To avoid a timeout, the watchdog specifies a window of opportunity during which it must be reset. This protects systems from operating too quickly or too slowly, hence boosting the error recognition coverage.

I/O Interface and Configuration

Figure 2 depicts the recommended watchdog timer's input/output (I/O) interface. WDFAIL (output on watchdog failure) and RESET (output on reset) are the two signals that come from the watchdog (RSTOUT). In the event that the SYSRESET input is set to 0, the WDFAIL output will remain asserted while the RSTOUT output will become deasserted. As can be seen in the figure, the architecture also includes a configuration register that has switchable bit fields.

In addition to altering and reporting status information, the register enables the watchdog settings to be modified. To reset and service the watchdog, use the corresponding WDRST and WDSRVC entries. Making automatic adjustments to the configuration register by altering the INIT input and WDFAIL output. In the FLSTAT column, the status of the service window is recorded, and in the SWSTAT field, the failure mode of the watchdog is noted.

The configuration register of the watchdog timer can be read from and written to via the ENABLE and RD/WR control inputs. The address bus (ABUS) and data bus (DBUS) are represented in the diagram by the corresponding signals.

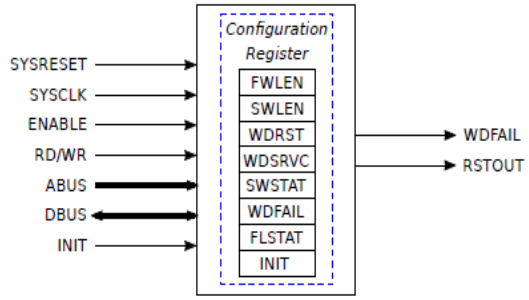

Fig2. The watchdog timer's I/O port and programmable register

There are two types of windows in the proposed windowed watchdog architecture: Both the service window and the frame window are referred to here. Construction time will be much longer than service time. Software can set the width and height of the two windows after powering on by programming the configuration register's bit fields SWLEN and FWLEN. The inability to change the settings once the window periods have been set is intentional.

It must first clear the configuration register before it can write to it again. the software must, if required, undergo a long unlocking procedure. This prevents a runaway code from accidentally changing the watchdog window's parameters. Beginning with the INIT input to the watchdog timer, the service window may begin. Unless the fail flag (WDFAIL) is set, the service window cannot begin until this input goes from high to low. If the CPU does not service the watchdog within the designated service window, the timer will time out.

Setting a value in the configuration register's watchdog service (WDSRVC) field keeps the watchdog timer running. If the edge of the frame is rising, the service window will automatically shut. When the frame window opens up is how often the watchdog needs attention. The watchdog is serviced once each cycle, and this window is usually longer than the embedded control system's main loop.

Multiple mechanisms exist for triggering the watchdog timer's INIT signal. Sending the INIT signal at the end of the main loop after a battery of validity checks has been made is one such strategy. To eliminate any CPU involvement in the generation of the INIT signal, an external interval timer might be employed. In this scenario, the frame window should be set to a length that is somewhat longer than the main loop's execution duration. Incorporating this preparation procedure into frame-based job planning in embedded systems is very beneficial.

Watchdog Timer Initialization

On power-up or reset, the WDFAIL assertion is set to true, indicating that the watchdog has awakened in a failed state. The programme is responsible for launching and maintaining the watchdog. Figure 3 depicts the waveform during regular operation and initialization of the watchdog reset. Therefore, in order to make the watchdog work, you must first adjust the configuration register's watchdog reset (WDRST) value from low to high. After that time period has elapsed, the WDFAIL flag will be reset and the watchdog will be serviced.

A new service window will start before the current one closes because the frame window was kept open for longer than the system frame time. The frame window will be reinitialized after the watchdog has been properly serviced again. As long as the frame window counts are functioning, the watchdog will not report any failures. Important real-time embedded systems need redundancy or variety to tolerate failure. The ability to assert the watchdog fail signal during power-up is useful in such systems. The fail state indicates that a particular channel is no longer accessible for calculations.

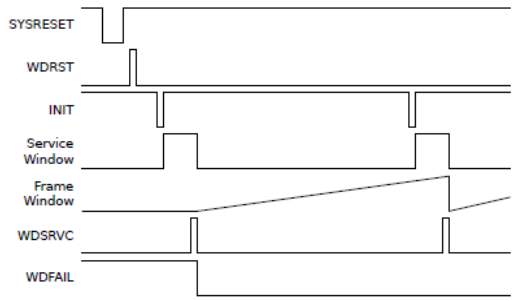

Fig.3. Initialization and servicing of the watchdog timer

The channel may be deemed online after the watchdog has been restored to a healthy condition. Furthermore, if a specific channel is determined to be malfunctioning during regular operations, the redundancy management logic may trigger the channel's watchdog fail. This basically prevents the defective channel from participating in any further calculations.

FAULT DETECTION FEATURES

To boost its efficiency in catching unpredictable programme modes, the proposed watchdog timer includes many defect detection techniques. The service window ends and an internal fail flag is set if the programme fails to service the watchdog within the allotted time In this scenario, the frame window does not reset when it reaches its maximum value but rather just ends. The watchdog asserts its WDFAIL signal when the frame window expires, signifying a failure. Figure 3 depicts this failure mode. As can be seen in Figure 5, a watchdog fail happens when the application attempts to service the watchdog outside of the service window. As can be seen, the frame window is promptly terminated, and the WDFAIL signal is asserted, upon the occurrence of the invalid service activity.

This feature has the advantage of causing a watchdog failure if two consecutive service activities are performed.

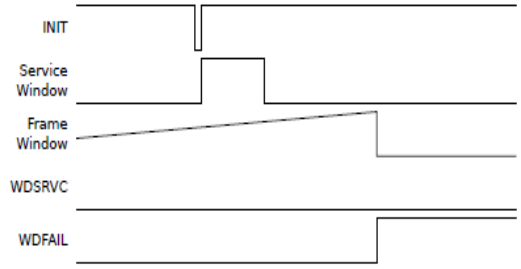

Fig4. Due to the expiration of the frame window, watchdogs fail.

The first maintenance process will instantly close the service window, and the second will almost probably take place outside of it. This is akin to doing watchdog maintenance outside of the service window and it results in the failure of the watchdog.

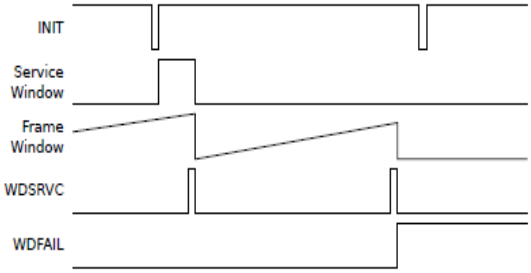

Fig.5. Service beyond the service window causes watchdogs to fail.

In the case shown in Figure 6, the falling edge of the WDSRVC occurs inside the service window. The watchdog failure signal is generated since this is likewise deemed an unauthorized service action. Before the next service window begins, the application must de-assert the WDSRVC signal after servicing the watchdog. All of these fault detection algorithms guarantee that the suggested watchdog timer will not go unnoticed if software goes wild. A fail-safe state may be triggered via the watchdog timer's WDFAIL output, or a non-mask able interrupt (NMI) signal can be sent to the CPU to alert it to an issue. After a certain amount of time after the WDFAIL signal has been asserted, The RSTOUT output of the watchdog will be asserted.

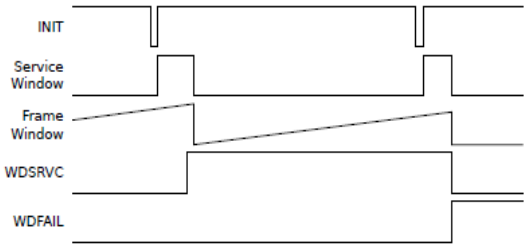

Fig6. Due to WDSRVC's dropping edge within the service window, the watchdog fails.

Embedded systems may be reset automatically by connecting this signal to the reset pin of the CPU. At this point, the program may save data that will be used later in the debugging process. The FLSTAT field is updated in the watchdog configuration register to reflect the failure mode. The programme may make an effort to log this data to non-volatile memory for further inspection during development.

WATCHDOG TIMERS IN FPGA

Here we will discuss the FPGA implementation details of the proposed watchdog timer. Figure 6 is a schematic of the hardware watchdog. The SYSCLK input is utilized to time the design as it is not tied to the CPU clock. The programme determines the potential window lengths, which are hard-coded in the design once the electricity is on, Set the SWLEN and FWLEN bits in the configuration register to acquire this information.

The window length configuration settings are automatically locked after the values are specified; to rephrase: access to these parts for writing purposes has been denied. In case further adjustments to the window widths are necessary, a 16-bit unlock register has been included into the design. To modify the window lengths, the programme must send values 0xAAAA and 0x5555 to this register twice in a row. The second pattern must be input within 10 seconds after the first, the programme then has 10 seconds to adjust the length configuration parameters. Writes to these

bits will be refused if these timings are not carefully followed.

The service window is activated when the INIT signal transitions from high to low. SWCLK is slower than SYSCLK. Slower clock decreases comparators, saving FPGA resources. An up/down counter (SYSCLK) and a primary counter (SYSWATCH) are used in the service window (SWCLK).To generate Toffset, an offset up counter adds INIT to the time that passes between the rising edge of SWCLK and INIT. INIT generation might occur asynchronously during SWCLK (denoted by Tswclk).The main counter is reset to (SWLEN - 1) times, and the offset value is stored.

The offset down counter runs for Tswclk Toffset seconds after the main counter has expired. The window length may be precisely controlled using this counting approach. The watchdog configuration register additionally receives periodic updates detailing the service window's operational condition.

The watchdog's service window counter stops incrementing and the frame window loads after it has been properly serviced. The frame window utilizes a slower clock (FWCLK). Like a service counter, it features both a primary and an up/down counter. The FWCLK rising edge is used by the offset up counter to determine the distance from the service window's end. Finally, the primary counter adds up (FWLEN - 1).Finally, the primary counter adds up (FWLEN - 1).

Reset Initialization and Fault Detection

Fault detection and initialization logics for the FSM watchdog are shown in FIGURE 7. WDFAIL is asserted upon power-on if the watchdog fails. The watchdog timer starts when WDRST rises. After the service window starts and the window counters begin ticking, the WDFAIL output is followed by a rising edge on the WDSRVC bit. The whole starting procedure is rejected and the program must be restarted if the watchdog isn't serviced correctly at the outset. WDFAIL is de-asserted after the watchdog has been successfully initialized.

If any of the failure types specified while the watchdog is active again asserts WDFAIL. The configuration register shows the failed status and kind. In the case of a watchdog failure, a reset counter is triggered. The quantity of debug information that has to be saved may be used to calculate the counter's duration. Whenever the WDT's countdown timer reaches zero, it will assert a high state on the RSTOUT pin. While the power supply is being connected, the RSTOUT output will be low, rendering the reset counter inoperable. The counter is automatically activated when the watchdog is started for the first time.

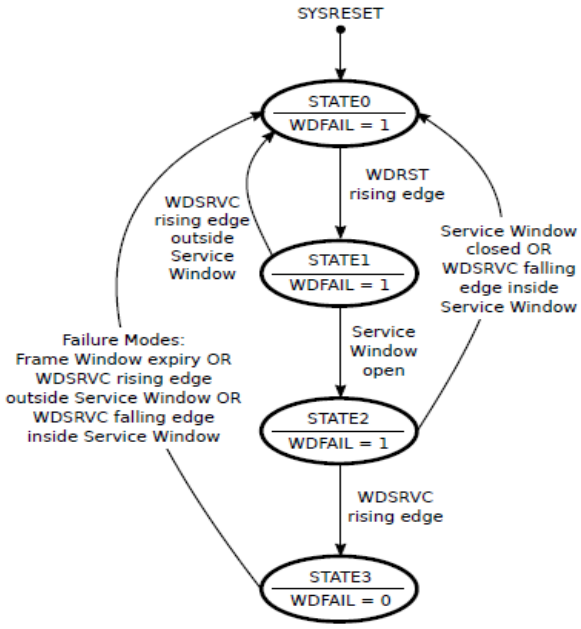

Fig. 7. The creation of a fault-detection "watchdog" using a finite state system

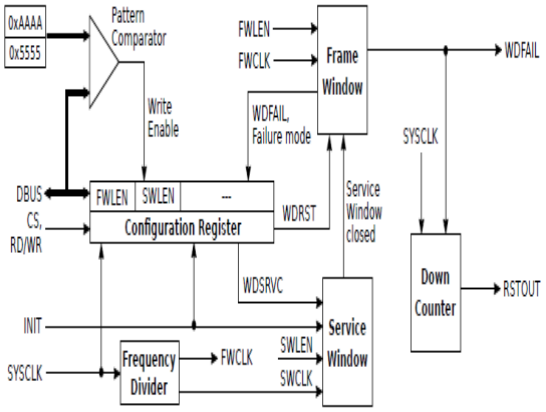

Fig.8.The suggested watchdog timer's functional block diagram

SKLANSKY ADDER (ADDER DESIGN USED IN PROPOSED WATCH DOG TIMER)

In this project SKA was used as incremented in service window and frame window of proposed watch dog timer. As the most efficient adder available, the SKA is frequently used in high-performance arithmetic circuits. Carry is generated in O(log n) time. By performing the calculations in parallel, SKA is able to quickly determine carriers at the expense of additional storage space. The following diagram, which corresponds to 16-bit SKA, shows how SKA works.

The schematic of a 16-bit Sklansky adder is shown in the diagram. Divide-and-conquer tree is another name for the Sklansky adder. Sklansky (1960) introduced conditional sum addition logic for prefix addition, which provides a minimal depth prefix network at the penalty of higher fan-out for particular calculation nodes. The longest lateral fanning wires link a node to n/2 more nodes. The Sklansky adder's fan-out grows dramatically from inputs to outputs along the crucial route, accounting for a considerable amount of delay. When the number of bits in the adder becomes huge, this reduces the structure's performance.

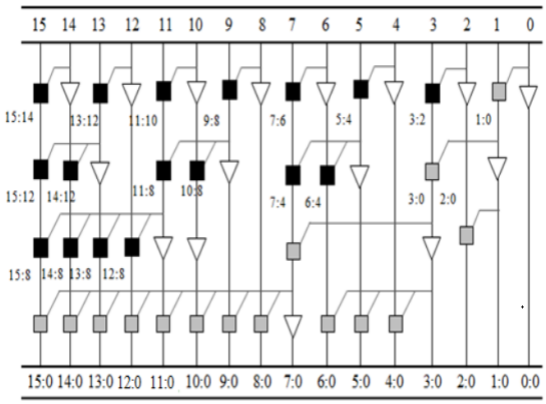

Fig9:16bit sklansky adder (Proposed adder)

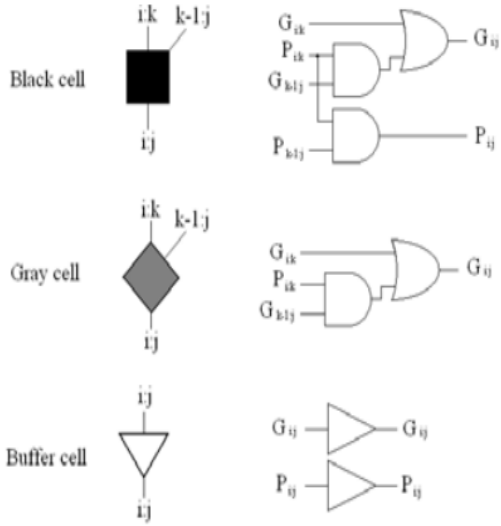

Fig10: gray cell, black cell and buffer.

III. RESULTS

RTL SCHEMATIC: The register transfer level (RTL) schematic denotes the architecture's blueprint and is used to compare perfect architecture that we must create from the intended architecture. The HDL language is used to transform the architecture's description or summary into the functioning summary using a coding language like verilog or vhdl. The internal connection blocks are even specified in the RTL schematic for easier analysis. Below is a schematic representation of the design's RTL implementation.

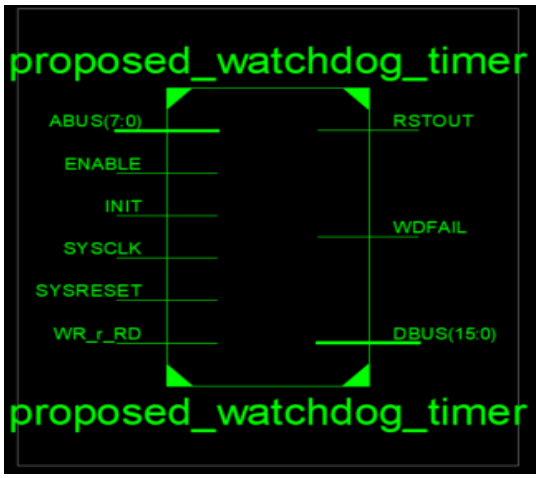

Fig 11: RTL Schematic of Proposed watch dog timer

TECHNOLOGY SCHEMATIC:

Technology Schematic With the LUT area parameter being used to estimate architecture design in VLSI, this diagram shows the technology's architecture in LUT format. The FPGA's LUTs, which are square units, represent the code's memory allocation.

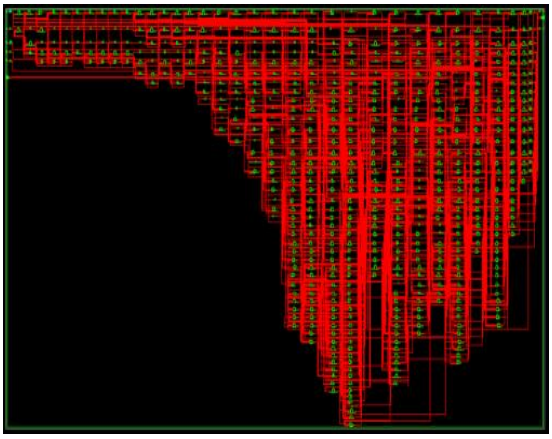

Fig12: View Technology Schematic of proposed watch dog timer

SIMULATION: Unlike the schematic, which only verifies the connections and blocks of a circuit, a simulation verifies the circuit's workings. Waveforms are the only form of output that can be viewed in the simulation window because it is launched by shifting from implementation to simulation. Since it can support multiple radix number systems, this is a useful feature.

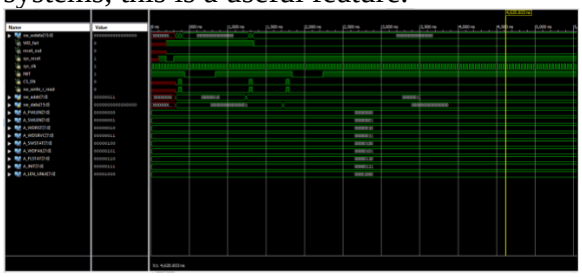

Fig13: Simulated Waveforms of proposed watchdog timer

PARAMETERS: Consider that in VLSI, the factors considered are area, delay, and power; using these metrics It's possible to make comparisons between several designs. XILINX 14.7 is used to acquire the parameter, while verilog is used as the HDL language.

Parameter	Existed Watchdog Timer	Proposed Watchdog Timer
Number of LUTs	407	324

Table1: parameter comparison

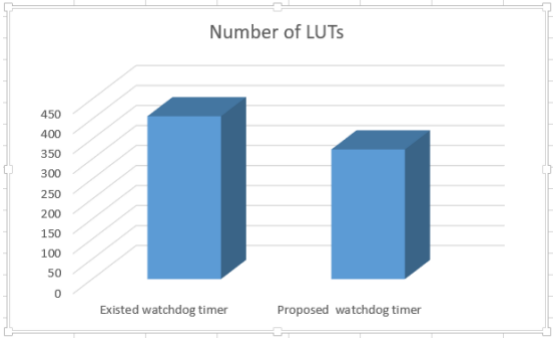

Fig14: LUT comparison bar graph

IV. CONCLUSION

The architecture and construction of an enhanced (windowed) watchdog timer employing the sklansky adder were detailed in this study. The watchdog timer is fully independent of the CPU and allows you to customize the timer settings to fit your needs. The watchdog includes many defect detection mechanisms for early identification of abnormal programme modes. It provides the ability to detect and record the failure type, which might be useful for troubleshooting. When the

watchdog timer detects a malfunction, it also gives the programme enough time to save debug information before performing a reset.

With very modest HDL changes, the same architecture may be modified for numerous processors and applications. The implementation is simple and consumes a little amount of hardware resources. The proposed approach has been proved to be effective in resolving various problems via testing using fault injection techniques in operational, safety-critical embedded hardware in real-time.

REFERENCES

[1] S. N. Chau, L. Alkalai, A. T. Tai, and J. B. Burt, "Design of a faulttolerantCOTS-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 fieldprogrammablegate 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 timer's application note," *Maxim Integrated Products*, 2008.

[8] P. Garcia, K. Compton, M. Schulte, E. Blem, and W. Fu, "An over view 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.