International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 05 | May 2019 www.irjet.net p-ISSN: 2395-0072

FPGA IMPLEMENTATION OF AN IMPROVED WATCHDOG TIMER FOR SAFETY CRITICAL APPLICATIONS

M. Jamuna[1], Mrs. S. Jayashree[2]

[1]Department of Electronics and Communication Engineering, Sri Ramanujar Engineering College, Chennai
[2]Assistant Professor, Department of Electronics and Communication Engineering, Sri Ramanujar Engineering College, Chennai

---

ABSTRACT - *Embedded systems that are employed in safety critical applications require highest reliability. This paper describes the architecture and design of an improved configurable windowed watchdog timer that can be employed in safety-critical applications. Several fault detection mechanisms are built into the watchdog, which adds to its robustness. This allows the design to be easily adaptable to different applications, while reducing the overall system cost and also the timing constrain is less in proposed watchdog than the existing due to the processor dependant. The effectiveness of the proposed watchdog timer to detect and respond to faults is first studied by analyzing the simulation results. Thus after designing the watchdog it is implemented in ATM and space launch vehicle and verified. The design is coded in Verilog and implemented in Xilinx 14.5 and implemented in FPGA Spartan 6. The design is validated in a real-time hardware by injecting faults through the software while the processor is executing.*

1. INTRODUCTION

A watchdog timer (sometimes called a computer operating properly or COP timer, or simply a watchdog) is an electronic timer that is used to detect and recover from computer malfunctions. During normal operation, the computer regularly resets the watchdog timer to prevent it from elapsing, or "timing out". If, due to a hardware fault or program error, the computer fails to reset the watchdog, the timer will elapse and generate a timeout signal. The timeout signal is used to initiate corrective action or actions. The corrective actions typically include placing the computer system in a safe state and restoring normal system operation.

Watchdog timers are commonly found in embedded systems and other computer-controlled equipment where humans cannot easily access the equipment or would be unable to react to faults in a timely manner. In such systems, the computer cannot depend on a human to invoke a reboot if it hangs; it must be self-reliant. For example, remote embedded systems such as space probes are not physically accessible to human operators; these could become permanently disabled if they were unable to autonomously recover from faults. A watchdog timer is usually employed in cases like these. Watchdog timers may also be used when running un-trusted code in a sandbox, to limit the CPU time available to the code and thus prevent some types of denial-of-service attacks.

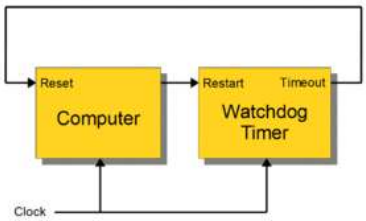


Fig 1. Block diagram of watchdog timer

2. EXISTING WATCHDOG TIMER:

In the existing system, a watchdog timer with no windowed watchdog is executed. The input is directly sent into the memory, from the memory instructions are processed into the processor, this watchdog will not detect the fault immediately. If there is any error occurrence in between them, it will sequentially wait for its time to trigger the CPU that error has occurred. It is totally dependent on the CPU. Then after CPU, getting the error information it will reset the whole process. It is stated as slow watchdog fault mechanism. The time it takes to reach the error mechanism to rectify is more than the proposed system. Since it is not clock independent, this sequential watchdog is a failure to embedded system. It is rectified during this proposed system.

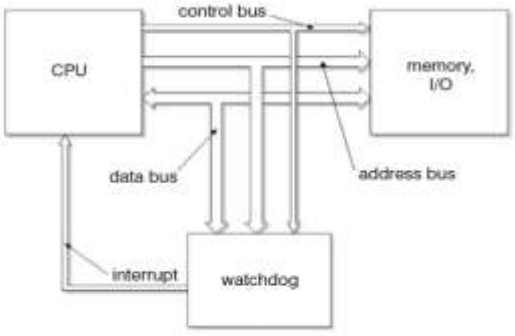


Fig2. Existing block diagram

DISADVANTAGES OF EXISTING SYSTEM:

- It doesn't have any window, thus if there is any error in between the processing time, the watchdog will observe the error only after the full processor design completes, thus fast faults cannot be detected with this method .
- Process is slow in this method.
- Data cannot be retrieved back in this method.

3. PROPOSED SYSTEM

The watchdog timer proposed in this paper operates independently of the processor and uses a dedicated clock for its functions of three windows such as service window, frame window and controller window. 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.

ADVANTAGE

- Since it has the window, it will detect faults immediately after the fault occurs, thus efficient than the existing system
- It is more efficient in detecting fast faults. Data can be retrieved during this process.
- Processing time is less than the windowless technique.

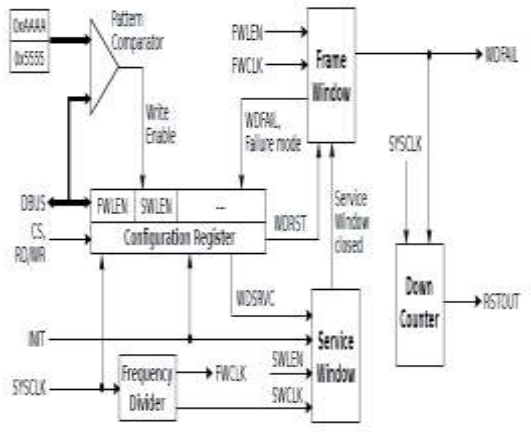


Fig 6. Functional block diagram of the proposed watchdog timer

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. 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. When the watchdog is correctly serviced, the counters in the service window stop immediately and the frame window starts.. The offset up counter here finds the offset between the termination of the service window and the next rising edge of the FWCLK. The frame window counters reset when a watchdog service operation occurs within the next service window duration, before the frame window expires.

4. PROPOSEDBLOCK DIAGRAM WITH FAULT INJECTION BLOCK:

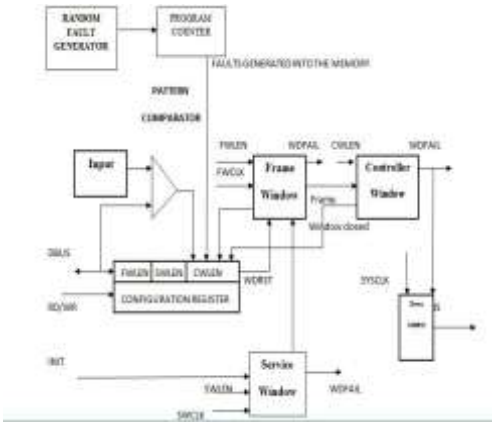


Fig3. Fault injection block diagram

The random numbers generated are injected into the program counter at random periods of time. A random pulse is driven from a second PN sequence generator. This pulse controls a multiplexer whose output is connected to the Program counter. The inputs are either an incrementer or the random generator. At all times the incrementer is selected as this is the normal operation of the system. Simultaneously the watchdog

timer is running and a counter records the number of times the watchdog is able to detect the injected faults.

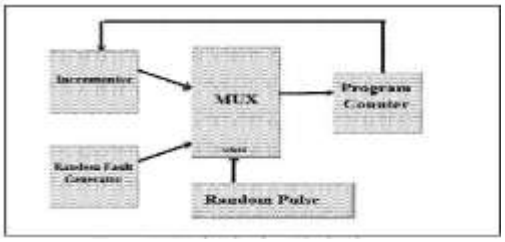


Fig4. Random fault generator block diagram

5. IMPLEMENTATIONOF WATCHDOG IN SPACE LAUNCH VEHICLE

Maintaining of space launch vehicle is the most important parameter to be fixed .if there is any parameter not checked or not built; it leads to large loss of money and time. Thus our watchdog timers will do this job in a perfect way. In our existing system, all the parameters are checked all at a time and if there is any fault in checking of temperature, pressure and heat explosion, then these parameters are made with some terminal value, checked with the parameters pre-recorded, thus leading to watchdog fault if there is an overflow value range. Similarly in our proposed system, we find an efficient way to check our parameters individually, thus leading to windowed watchdog timer. Each window goes on checking with each parameter, thus leading to wdfail in each stage.

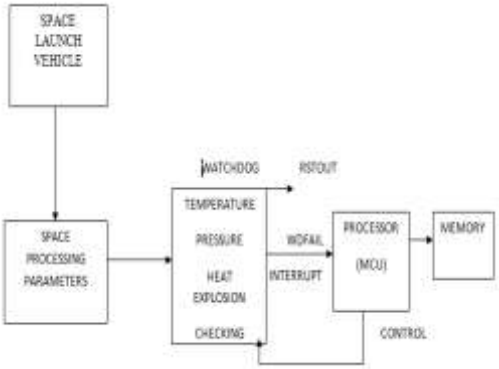


Fig5. Existing application of space launch vehicle

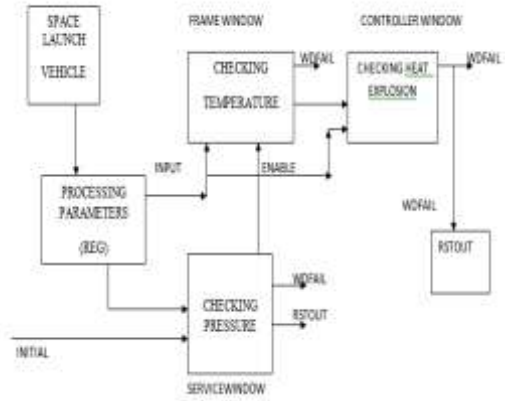


Fig6. Proposed application of space launch vehicle

6. RESULTS AND SIMULATION

IF pressure, temp and heat values obtained from the sensor is subjected to test and maintain by watchdog, thus the terminal values of all three are maintained and processed, in windowless technique, if there is a parameter meeting its extreme me limit, wdfail = 0, thus watchdog making an interrupt request to processor, thus processor takes the action over it to reset.

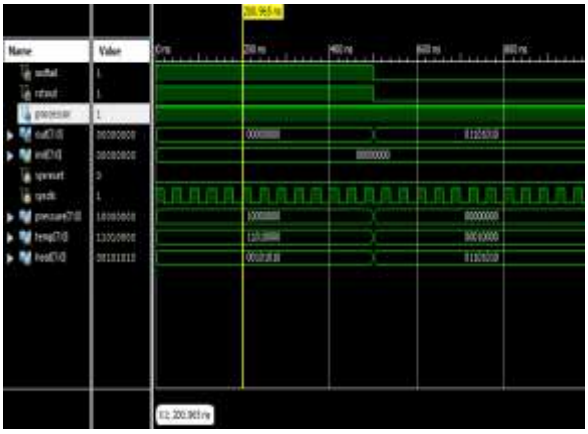
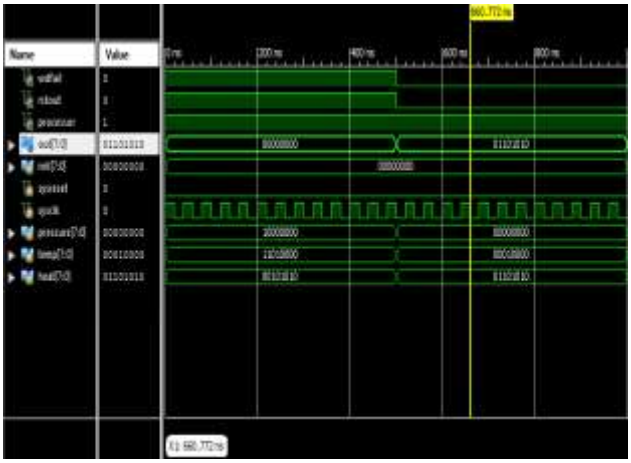


Fig7. Simulation of existing application of space launch vehicle

if there is no extreme limit met , then wdfail = 0; thus rstout = 0;



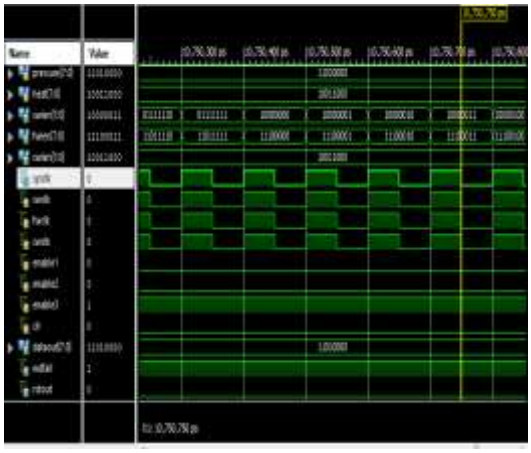
DEVICE SUMMARY:

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	13	5720	1%
Number of fully used LUT-FF pairs	0	13	1%
Number of bonded IOBs	49	102	49%

TIMING REPORT

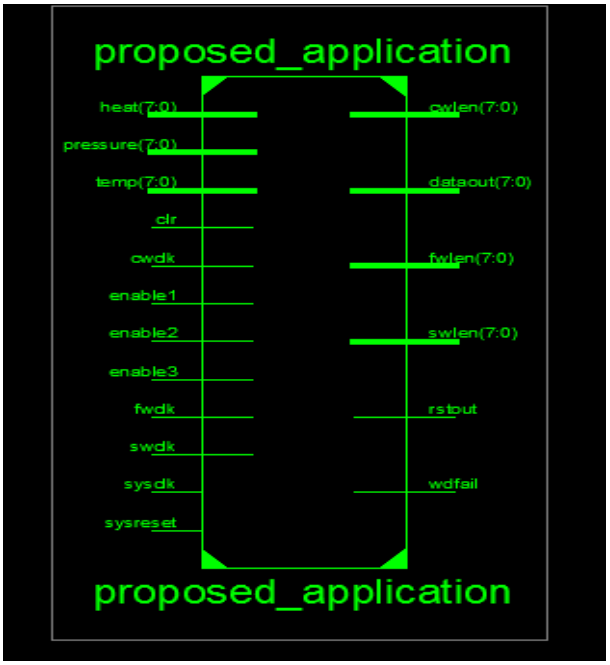
```
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: No path found
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 7.821ns

Timing Details:
---------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default path analysis
  Total number of paths / destination ports: 256 / 10
-------------------------------------------------------------------------
Delay:               7.821ns (Levels of Logic = 5)
  Source:            temp<1> (PAD)
  Destination:       out<7> (PAD)
```

PROPOSED SYSTEM:



IF RSTOUT, IT GOES BACK TO THE ORIGINAL INITIAL VALUE OF THE SPACE LAUNCH VECHICLE:

RTL SCHEMATIC:



DEVICE SUMMARY:

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	13	5720	1%
Number of fully used LUT-FF pairs	0	13	1%
Number of bonded IOBs	49	102	47%

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

   Minimum period: 3.318ns (Maximum Frequency: 301.432MHz)
   Minimum input arrival time before clock: 3.580ns
   Maximum output required time after clock: 4.900ns
   Maximum combinational path delay: 14.818ns

Timing Details:
---------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default period analysis for Clock 'cwclk'
  Clock period: 3.318ns (frequency: 301.432MHz)
  Total number of paths / destination ports: 144 / 16
-------------------------------------------------------------------------
Delay:               3.318ns (Levels of Logic = 3)
  Source:            w8/temp_2_P_2 (FF)
  Destination:       w8/temp_7_C_7 (FF)
  Source Clock:      cwclk rising
  Destination Clock: cwclk rising
```

CONCLUSION

DESCRIPTION	AREA ANALYSIS	TIMING REPORT
EXISTING APPLICATION WATCHDOG TIMER	13 LUT's	7.821ns
PROPOSED APPLICATION WATCHDOG TIMER	13 LUT's	3.318 ns

Thus from the results we prove that timing taken by the windowed watchdog timer is taking much slower time than the existing without window technique. This project 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.

REFERENCES

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

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

[3] J. Beningo, "A review of watchdog architectures and their application to Cubesats," Apr. 2010.

[4] 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.