DESIGN AND IMPLEMENTATION OF HIGH SPEED FPGA CONFIGURATION USING SBI

Mr. B.GOWRI SHANKAR[1], V.J.JAYAPARKAVI[2]

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

[2]*Department of Electronics and Communication Engineering, Sri Ramanujar Engineering College, Chennai*

---***---

ABSTRACT:- FPGA technology is used in all kinds of high speed devices now, which also need so many demands in Quick configuration of FPGA at runtime is the latest need. Scheduling in FPGAs is increasingly being employed in modern real-time embedded systems, which often impose strict timeliness constraints. The existing system aims at tackling such a problem with a self aware approach. The security module checks the course of the proceedings at each intermittent point of a schedule and on detecting a malicious environment heals the scenario and takes precautions to prevent similar malfunctions in future.In the proposed system SBI(Spare blocks interfacing) is utilized for scheduling the process more accurately and timely. The advantage of spare block is multiple choice multi tasking spare support during scheduling process. The spare blocks are interfaced in four directions so if there is any Trojan malfunction in process, the spare blocks will provide scheduling help by generating temporary storage space, spare data, or delay clock etc which obviously improve the performance of the FPGA configurations.

1. INTRODUCTION

The property of quick reconfiguration of Field Programmable Gate Array(FPGA) at runtime is being utilized in arenas ranging from avionics and automotive systems to nuclear reactors. A Field-Programmable Gate Array(FPGA)is an integrated circuit designed to be configured by a customer or a designer after manufacturing. The FPGA configuration is generally specified using a Hardware-Description Language(HDL),similar to that used for an application-specific integrated circuit(ASIC). Correctness of the time system depends both on the logical results and the time of the result generation, as output generation after deadline may be catastrophic.

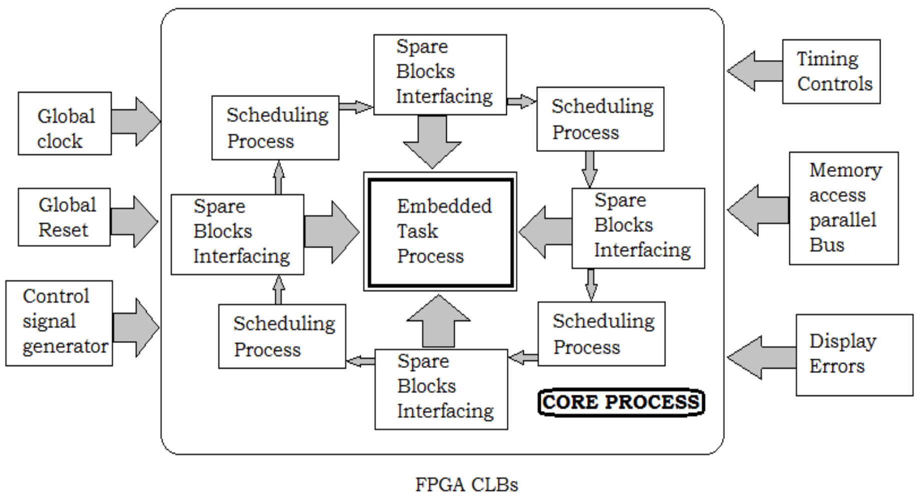


Fig 1. Block diagram of FPGA CLB's

2. EXISTING SYSTEM:

In the existing system, aims at tackling such a problem with a self aware approach. The security module checks the course of the proceedings at each intermittent point of a schedule and on detecting a malicious environment heals the scenario and takes precautions to prevent similar malfunctions in future 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.

DISADVANTAGES OF EXISTING SYSTEM:

- Process is slow in this method.
- Existing process utilized only one Self healing block which can be affected sometimes and can interrupt the scheduling process.
- Data cannot be retrieved back in this method.

3. PROPOSED SYSTEM

In the proposed system SBI (Spare blocks interfacing) is utilized for scheduling the process more accurately and timely. The advantage of spare block is multiple choice multi tasking spare support during scheduling process. The spare blocks are interfaced in four directions so if there is any Trojan malfunction in process, the spare blocks will provide scheduling help by generating temporary storage space, spare data, or delay clock etc which obviously improve the performance of the FPGA configurations 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
- SBI (Spare blocks interfacing) is utilized for scheduling the process more accurately and timely
- Multiple choice multi tasking spare support during scheduling process

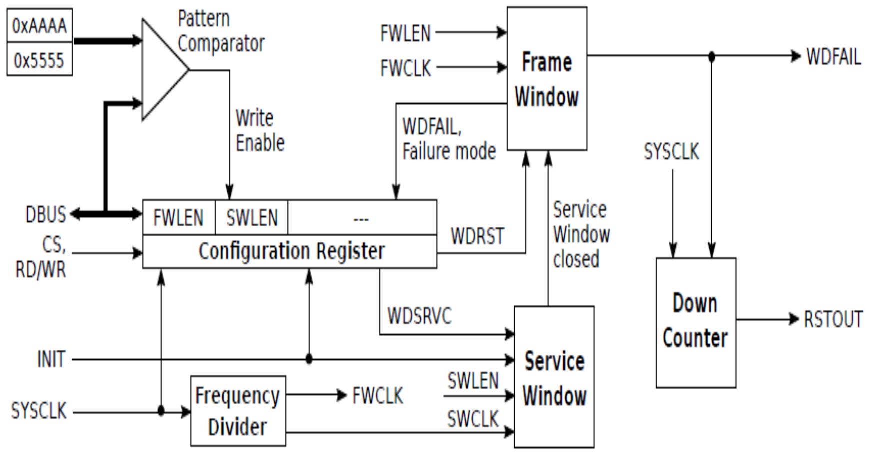


Fig. 6. Functional block diagram of the proposed watchdog timer

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:

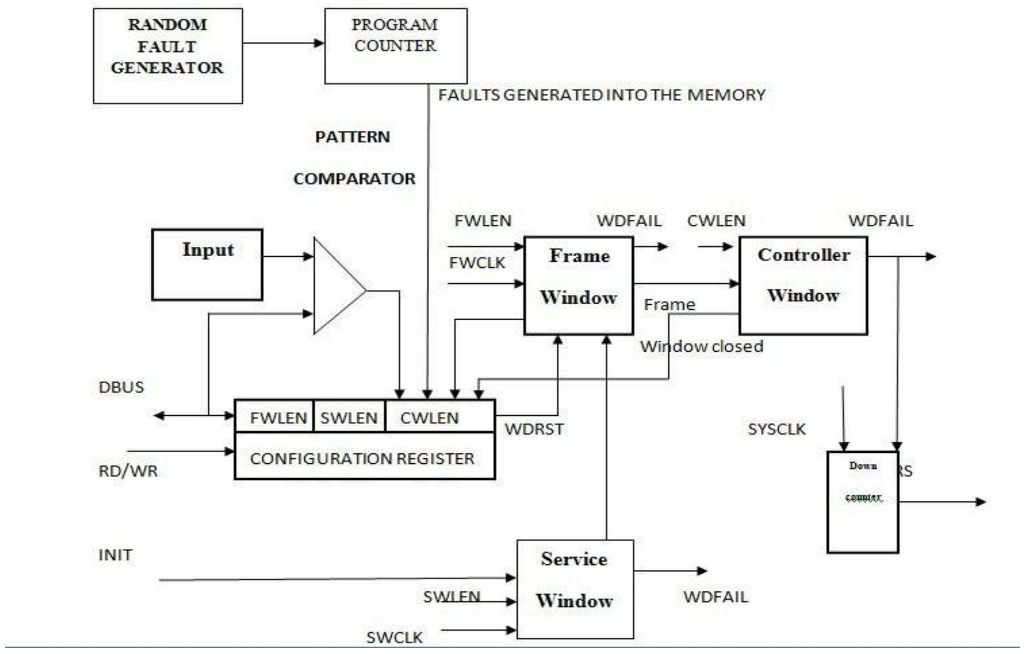


Fig2. Fault injection block diagram

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.

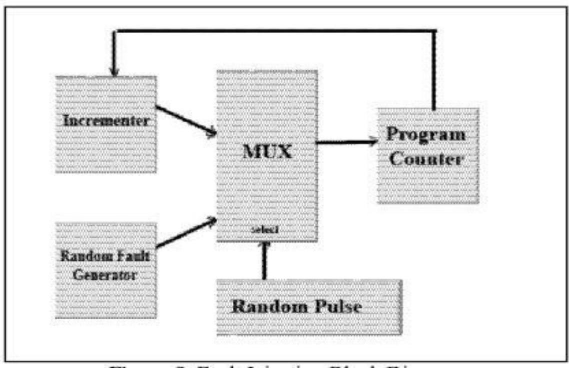


Fig3. Random fault generator block diagram

5. IMPLEMENTATION OF FPGA IN SCHEDULING PROCESS

A behavioral description specifies the sequence of operations to be performed by the synthesized hardware. This description is compiled into an internal data representation such as the control/data flow graph (CDFG). Scheduling algorithms then partition the CDFG into sub graphs so that each sub graph is executed in one control step. Each control step corresponds to one state of the controlling finite-state machine. Within a control step, a separate functional unit is required to execute each operation assigned to that step. Thus, the total number of functional units required in a control step corresponds to the number of operations scheduled in it. 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.

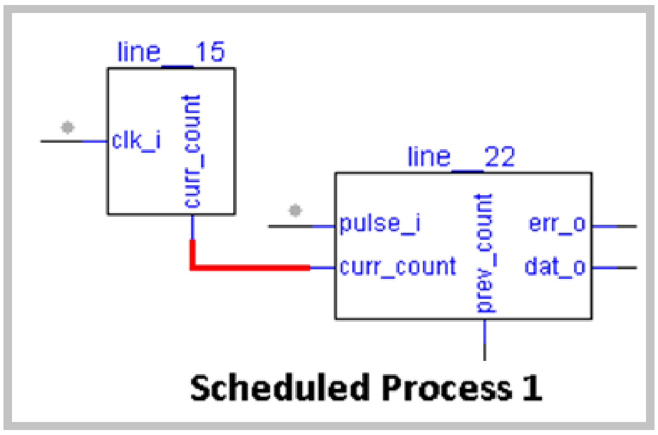
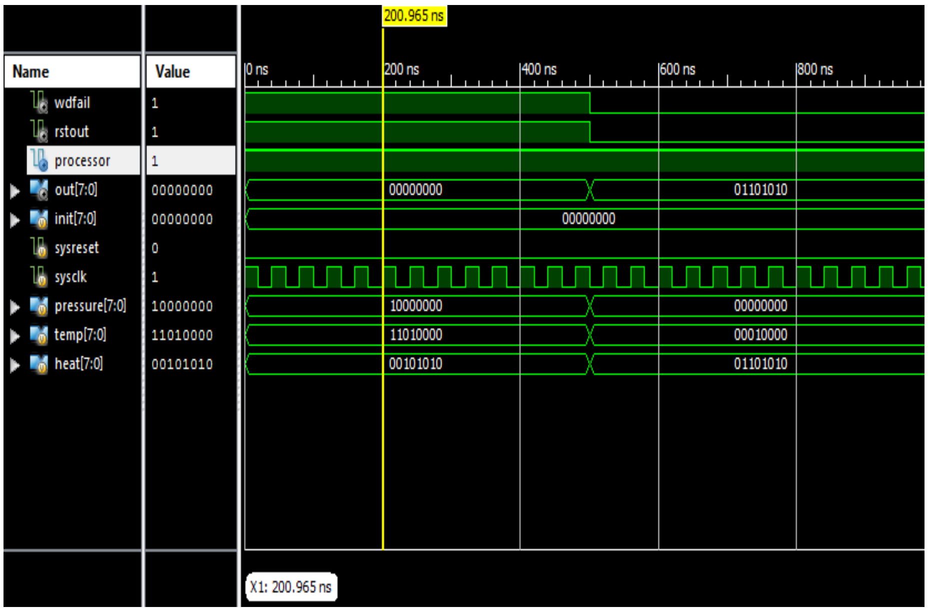


Fig4. SCHEDULING

6. RESULTS AND SIMULATION

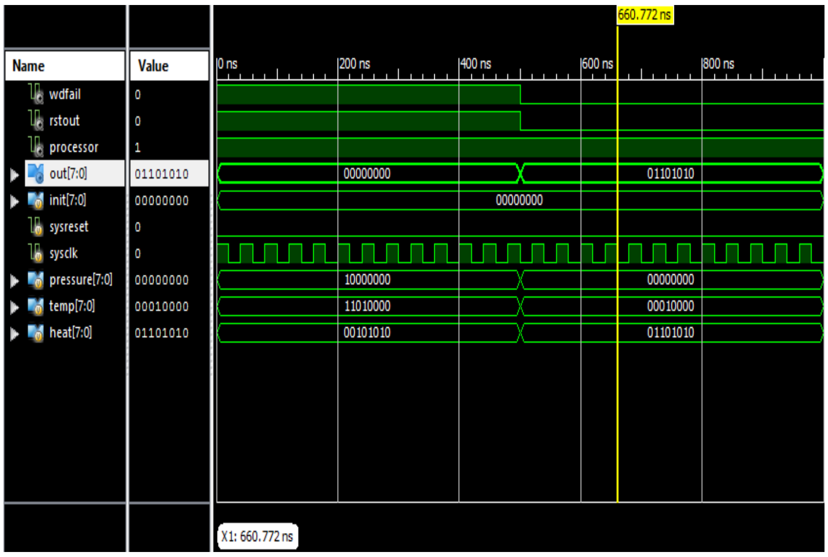
IF pressure, temp and heat values obtained from the sensor is subjected to test and maintain by watchdog, thus the terminal values of all three are maintained and processed, in windowless technique, if there is a parameter meeting its extreme me limit, wdfail = 0, thus watchdog making an interrupt request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



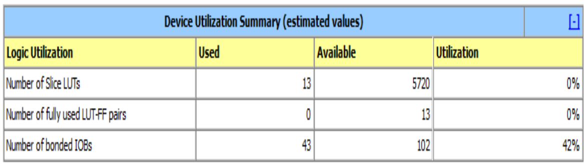
Name	Value
wdfail	1
rstout	1
processor	1
out[7:0]	00000000
init[7:0]	00000000
sysreset	0
sysclk	1
pressure[7:0]	10000000
temp[7:0]	11010000
heat[7:0]	00101010

Fig7. Simulation of existing application of space launch vehicle

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

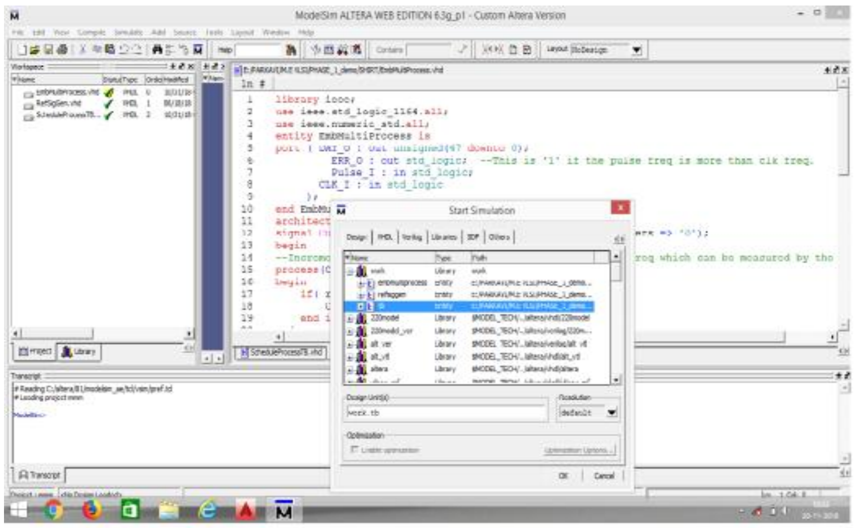


DEVICE SUMMARY:



Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	13	5720	0%
Number of fully used LUT-FF pairs	0	13	0%
Number of bonded IOBs	43	102	42%

CODE SIMULATION:

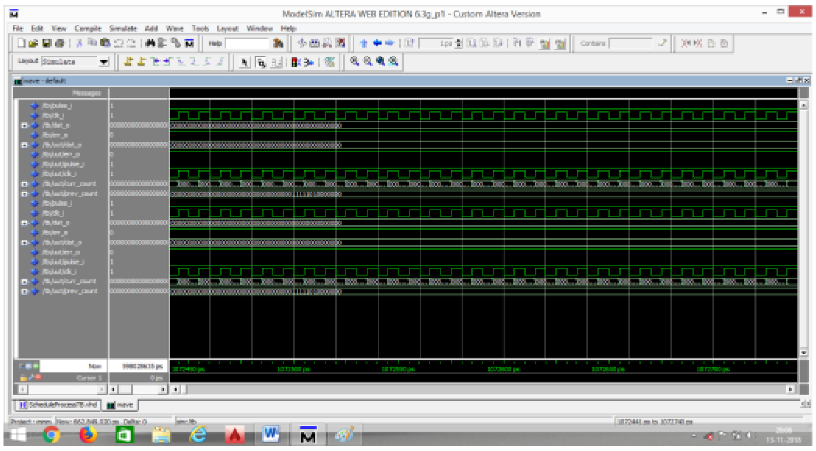


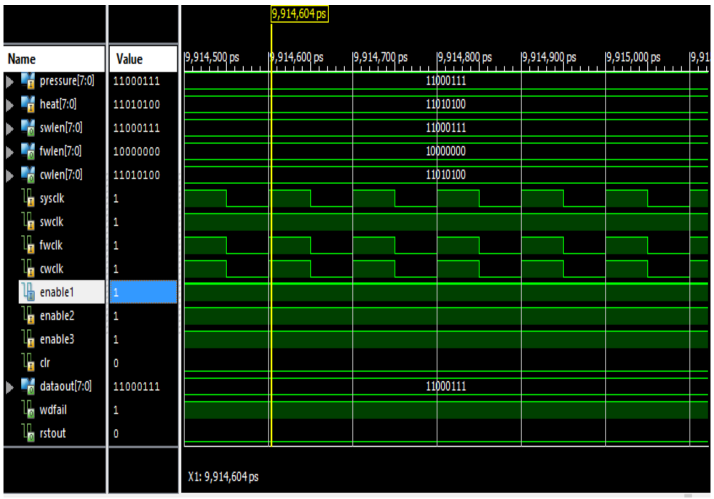
ModelSim ALTERA WEB EDITION 6.3g_p1 - Custom Altera Version

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity EmbMultiProcess is
port ( DAT_O : out unsigned(47 downto 0);
        ERR_O : out std_logic;  --This is '1' if the pulse freq is more than clk freq.
        Pulse_I : in std_logic;
      CLK_I : in std_logic
    );
end EmbMu
architect
signal
begin
--Increme
process(C
begin
    if(
       
    end i
```

FIG5. CODE FORMAT

PROPOSED SYSTEM:

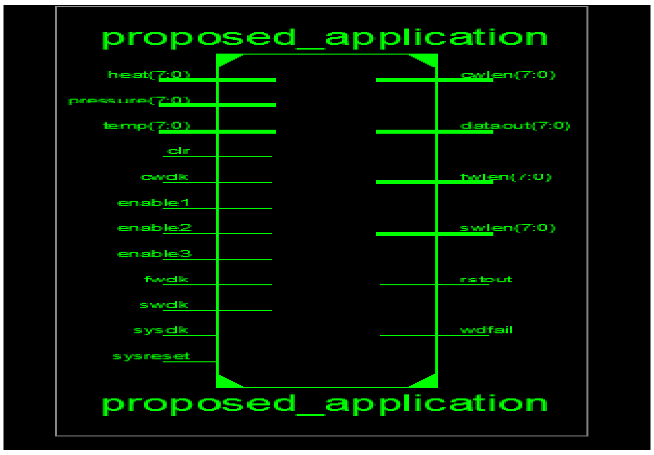




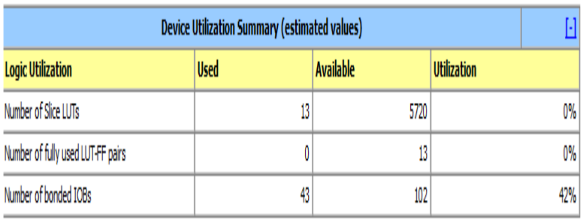
Name	Value
pressure[7:0]	11000111
heat[7:0]	11010100
swlen[7:0]	11000111
fwlen[7:0]	10000000
cwlen[7:0]	11010100
sysclk	1
swclk	1
fwclk	1
cwclk	1
enable1	1
enable2	1
enable3	1
clr	0
dataout[7:0]	11000111
wdfail	1
rstout	0

FIG 6. TEST BENCH WAVEFORM

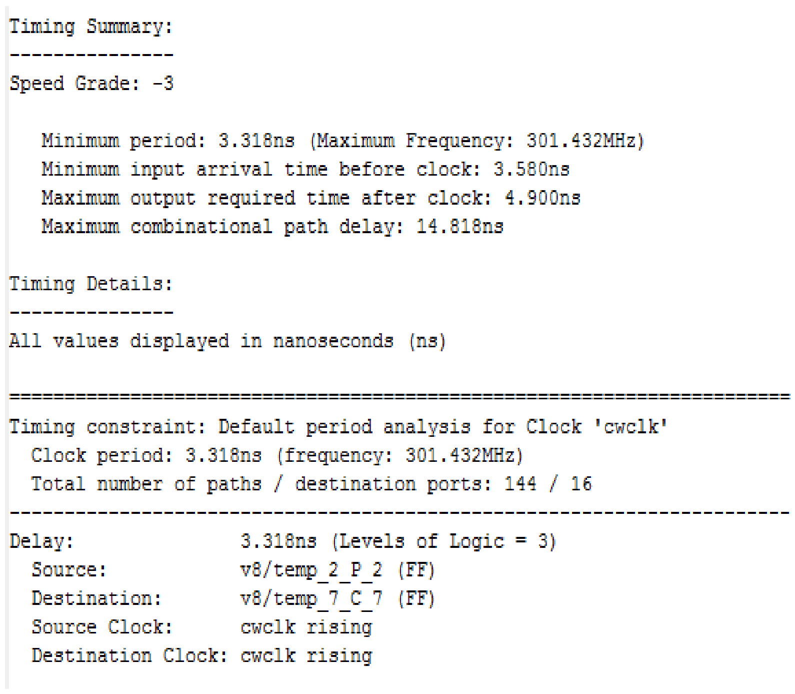
RTL SCHEMATIC:



DEVICE SUMMARY:



Device Utilization Summary (estimated values)			[-]
Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	13	5720	0%
Number of fully used LUT-FF pairs	0	13	0%
Number of bonded IOBs	43	102	42%



```
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:            v8/temp_2_P_2 (FF)
  Destination:       v8/temp_7_C_7 (FF)
  Source Clock:      cwclk rising
  Destination Clock: cwclk rising
```

6. CONCLUSION

The Scheduling in FPGAs is increasingly being employed in modern real-time embedded systems, which often impose strict timeliness constraints. The existing system aims at tackling such a problem at runtime with a self-aware approach. The security module checks the course of the proceedings at each intermittent point of a schedule and on detecting a malicious environment heals the scenario and takes precautions to prevent similar malfunctions in future. In the proposed system SBI(Spare blocks interfacing) is utilized for scheduling the process more accurately and timely. The advantage of spare block is multiple choice multitasking spare support during scheduling process. The spare blocks are interfaced in four directions so if there is any Trojan malfunction in process.

REFERENCES

1. "**Task Mapping Techniques For Embedded Many-core SOCs.**"Junya Kaida,Takuji Hieda, Ittestu Tangiguchi, Hiroyuki Tomiyama, Yuko Hara-Azumi, Koji Inoue **2012.**
2. **"Phase Cyclical Process Requirements For The Development Of Embedded System"** Magda A.Silverio Miya Shiro,Maurico G.V.Ferreira **2014.**
3. **"Task Scheduling of Improved Time Shifting Based on Genetic Algorithm for Phased Array Radar"** Lu Hua,Xiaopengyang,Shanlaun Hu **2016.**
4. **"Design and Implementation of a General Purpose Power Saving Scheduling Algorithm for Embedded System".** Kengo-Maochol,Chu-Hung Liang 1, Jun-Ying Huang 1, Chu-Shing Yng 1, 2 **2011.**
5. "**Shielding Heterogeneous mpsocs from untrustworthy 3PIPs through security driven task scheduling"** C. Liu, J.Rajendran, C. yang, R. Karri.

6. **“Self-Aware SoC Security to Counteract Delay Inducing Hardware Trojans at Runtime”** K. Guha, D. Saha, and A. Chakrabarti.
7. **“Scheduling Dynamic hard real-time task sets on fully and partially reconfigurable platforms”** S. Saha, A. Sarkar, A. Chakrabarti.
8. “**Hardware Trojan Attacks: Threat analysis and countermeasures**” S. Bhunia, M.S. Hsiao, M. Banga, S. Narasimhan.