RESEARCH ARTICLE OPEN ACCESS

Detection of Errors on Serial Communication Lines in RTOS Based-Embedded Systems

Saurabh Kumar*, Narayanaraju Samunuri**
*(Department of ECE, Sreenidhi Institute of Science and Technology, JNTU-Hyderabad, India)

ABSTRACT
Embedded Real Time application uses multi threading, a key concept of any conventional OS. The advantage of multi-threading include greater throughput, more efficient use of CPU such that it cannot remain idle for long time, better system reliability, improved performance on multiprocessor computer. The use of Real Time Operating Systems (RTOSs) became an attractive solution to simplify the design of safety-critical real-time embedded systems. Due to their demanding strict attention to rules and procedures constraints such as high-speed, low voltage operation and battery-power dependence, because of sudden up and down of voltages, transients in flow of current can cause error in serial communication lines. External conditions, such as Electromagnetic Interference (EMI), Heavy-Ion Radiation (HIR) as well as Power Supply Disturbances (PSD) can also cause transient faults. As the major consequence, the system's reliability degrades.
In this paper, the main focus will be on the detection of the errors on the serial communication lines. This is achieved by using a hardware approach with the combination of Cyclic Redundancy Check and RTOS. During execution of these programs, the proposed system exposed to EMI according to the international standard for voltage transients, voltage dips and short interruptions on the serial communication lines of electronic systems. The obtained result shows that the proposed approach is able to provide higher fault coverage.
Keywords**—** Hardware-Based Approach, Real-Time Operating System, Reliable Embedded System, Electromagnetic Interference (EMI), Serial Communication lines, Cyclic Check Redundancy (CRC).

I. INTRODUCTION

One of the key characteristic of an operating system (OS) is its ability to handle to multiple tasks at a time on a time sharing basis commonly referred to as Multi-tasking. It is also responsible for managing the hardware resources of a computer and hosting applications that execute on the computer. A real-time operating system is a specialized type of operating system where execution of tasks has to be done precisely without exceeding the deadlines and is intended to use for Real-time systems.

The OS meant for RTS is referred as RTOS where the time at which results are produced is of major concern. Basically, real-time systems are classified in to two types: Hard and Soft real-time systems. Now days, all real time embedded systems for safety measures uses timing constraints. In general terms, real-time systems have to provide not only logically correct results [1] but also in how much time they are producing it. The necessity to adopt Real-Time Operating Systems (RTOSs) is increased as the real time systems are getting complex day by day in order to simplify the design. Though, embedded systems based on RTOSs exploit some important facilities associated to RTOSs' native intrinsic mechanisms to manage tasks, concurrency, memory as well as interrupts. In other words, RTOSs serve as an interface between software and hardware.

At the same time, the environment's always increasing hostility caused substantially by the ubiquitous adoption of wireless technologies, such as mobile telephones, represents a huge challenge for the reliability [2] of real-time embedded systems as these equipments causes' radiations and affects the behavior of any embedded system which can leads to many fatal results. In detail, external conditions, such as Electromagnetic Interference (EMI) [3], Heavy-Ion Radiation (HIR) as well as Power Supply Disturbances (PSD) [4] may cause transient errors. Currently, the consequences of transient errors represent a well-known concern in microelectronic systems. Though, embedded systems based on RTOS are subject to Single Event Upsets (SEUs) causing transient errors, which can affect the application running on embedded systems.

The suggested approach in this paper is to use the Plasma MIPS architecture with the combination of RTOS μC/OS-II (Platform) with Cyclic Check Redundancy (CRC) [5] to detect the errors on serial Communication lines. To test and achieve the required performance it is run on FPGA kit and the output is to be observed on the HyperTerminal via UART

II. IMPLEMENTATION

The implementation of this concept consists of Software Approach (which consists of combination of Real Time Operating System (μC/OS-II) with the application of detecting error i.e. Cyclic Redundancy Check) on Plasma MIPS architecture (Processor Used).

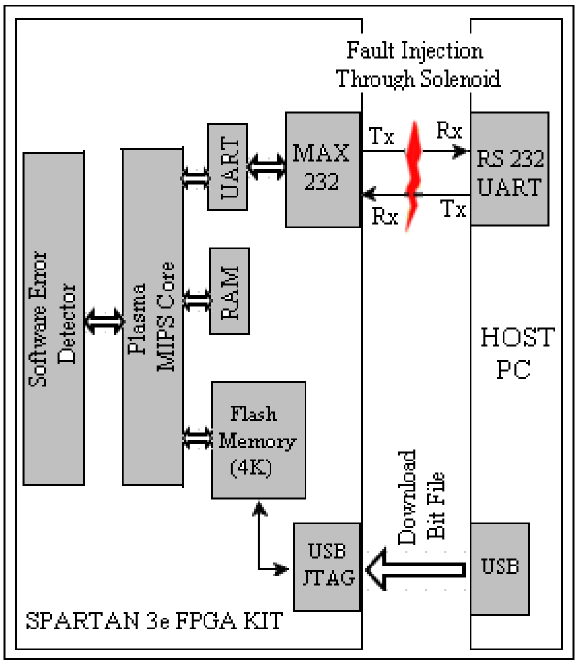


Fig.1: Block Diagram of Proposed Approach

1.1 Processor Used (Plasma MIPS Architecture)

To evaluate the hardware based approach it consists of RTOS running on Plasma MIPS architecture [10]. The Plasma MIPS is implemented in VHDL with exception of the load/store instruction, an instruction set compatible to the MIPS architecture. The Plasma RTOS uses the Pre-emptive scheduling algorithm with priority support compose of the following three states: *blocked*, *ready* and *executing*.

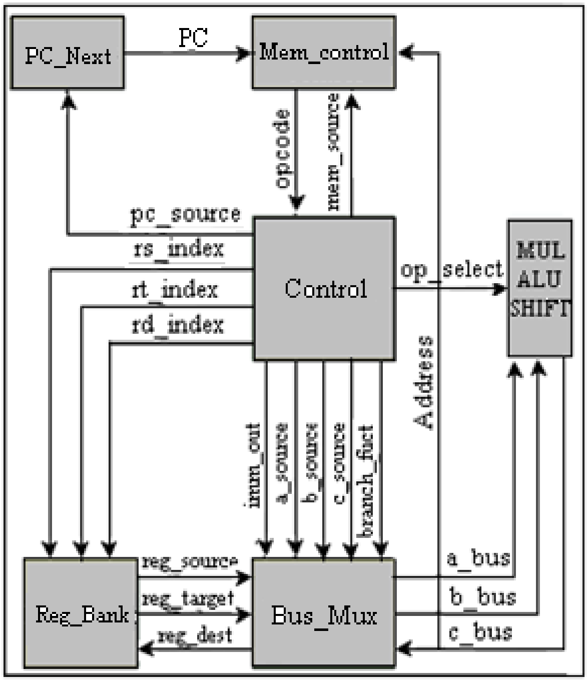


Fig. 2 Plasma MIPS Processor Architecture

The Plasma's RTOS provides a basic mechanism able to detect the errors on serial communication lines. This used processor i.e. Plasma MIPS architecture is linked with the software approach to produce desire results.

1.2 Software Approach

The software approach consists of µC/OS-II [6] platform with the application of detecting the errors which causes deviation in the outputs or generates dysfunctions that could lead to incorrect system behaviour using cyclic redundancy check. The main operation of software approach is to detect the fault on serial communication lines which causes the data corruption.

To carry out this approach GNU tool chain [7] is used to compile the µC/OS-II files with the application to detect the faults i.e. Cyclic Redundancy Check. These files will get compiled by using MIPS GNU Compiler to generate the individual object files ***(.obj files***). Now MIPS GCC Linker links all the ***.obj files*** and gives the output as ***.elf32 file*** (Executable and linkable format) which is saved with the extension of ***.axf file*** (Arm executable format; which is an object file format generated by GCC Compiler and linker and contains both object code and debugging information). Now with the help of ***convert_bin.exe*** command ***.axf file*** has to be converted into ***.txt file***. Now we have to generate ***ram_image.vhd*** file with the help of ***toimage*** command.

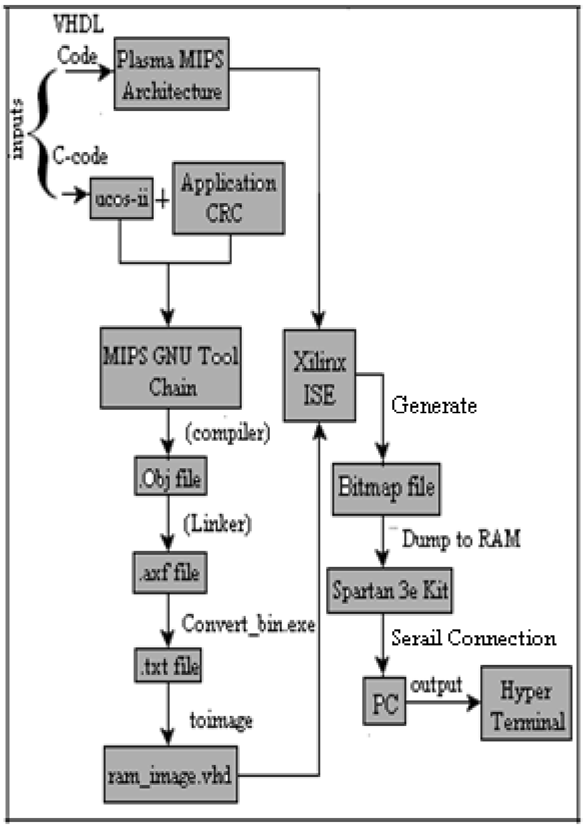


Fig.3 Project Process flow

Now, the generated ***ram_image.vhd*** which contains the complete information of µC/OS-II files and the application build to detect the error which is written in C-code is converted into ***.vhd format.*** Now the generated ***ram_image.vhd*** file has to be linked with the used architecture i.e. Plasma MIPS architecture written in VHDL code with the help of Xilinx ISE software to generate the ***Bitmap file*** and then this generated bit map file is to be dump on SPARTAN 3E 1600 FPGA kit [9] to perform the required operation using iMPACT Programming.

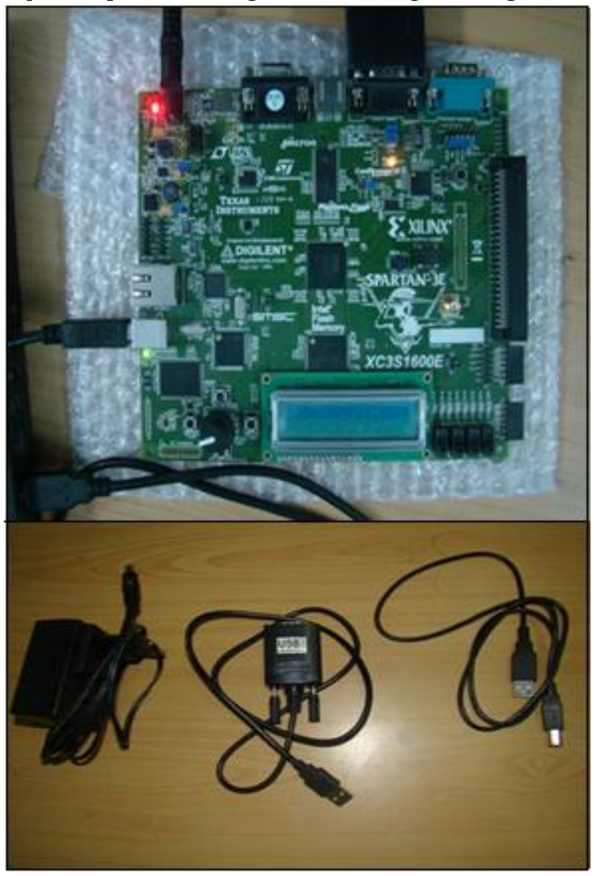


Fig.4 Hardware Setup

III. EXPERIMENTATION

After successfully compiling an FPGA design and generating the ***bit map file*** using the Xilinx development software, now the design (bit map file) has to be downloaded on the SPARTAN 3E FPGA kit using the iMPACT programming software and the USB cable.

To Configure the FPGA kit the following steps have to be followed:

1. The first step is to connect the SPARTAN board to PC using standard USB Type A/Type B cable.

2. Click on configure device (iMPACT), it opens iMPACT window then select boundary scan and click on initialize JTAG chain. If board is connected properly the software automatically recognizes the devices in the JTAG programming file.

3. Right click on FPGA device i.e. XC3S1600E and select assign configuration file, then browse for the bit file and click ok and bypass remaining options as shown in Fig.5.

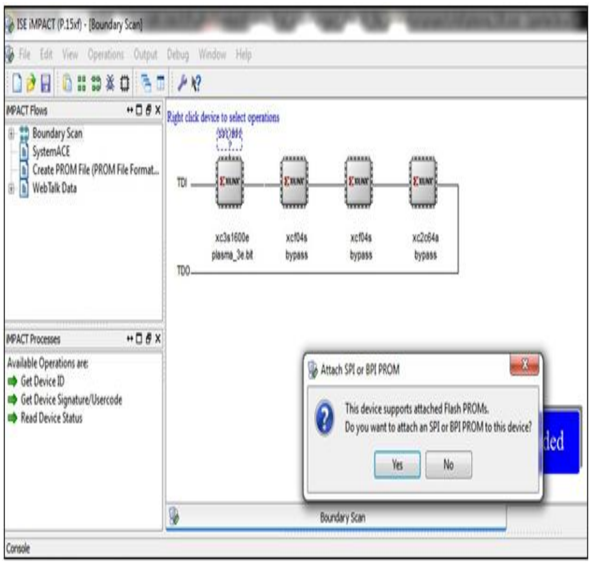
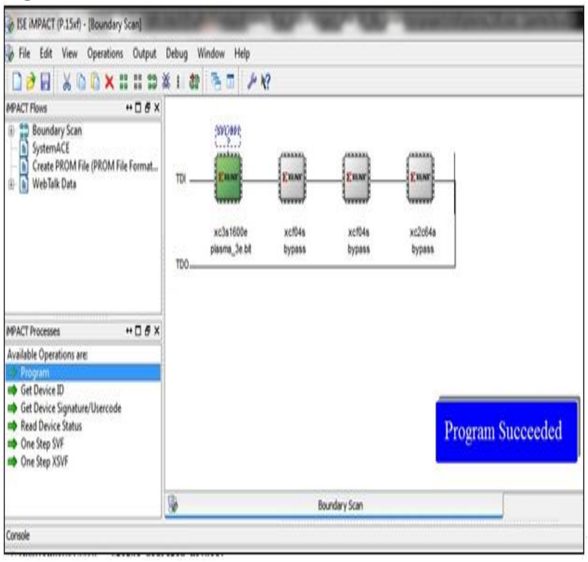

ISE iMPACT (P.15xf) - [Boundary Scan]
File Edit View Operations Output Debug Window Help
iMPACT Flows
Boundary Scan
SystemACE
Create PROM File (PROM File Format...
WebTalk Data
Right click device to select operations
TDI
TDO
xc3s1600e
plasma_3e.bit
xcf04s
bypass
xcf04s
bypass
xc2c64a
bypass
iMPACT Processes
Available Operations are:
Get Device ID
Get Device Signature/Usercode
Read Device Status
Attach SPI or BPI PROM
Yes
No
Boundary Scan
Console
This device supports attached Flash PROMs.
Do you want to attach an SPI or BPI PROM to this device?

Fig.5 Assigning Configuration File

4. Right on the device and select program, which configures the FPGA device, and it displays Program Succeeded once it is successfully finished as shown in Fig.6.



Program Succeeded

Fig.6 FPGA Configured

After successfully configured the FPGA now SPARTAN 3E FPGA development kit is ready to run the architecture with µC/OS-II files and the application to detect the errors (i.e. CRC). . Before dumping the architecture in to the FPGA the ***.bin*** file corresponding to the architecture is being generated and is dumped in to the board using the iMPACT software which is part of XILINX ISE. GCC tool is used for compiling the µC/OS-II with application and at the same time generating the *hex file* corresponding to the required application. The operating system and application executable were loaded into the FPGA's block RAM and executed from there. And then an application is to be run on architecture and the corresponding output is to be observed in HyperTerminal by means of UART.

IV. RESULTS

As mentioned earlier, the application (CRC) has to be developed with the combination of RTOS (µC/OS-II) with MIPS architecture and has to be dumped on FPGA kit [9]. The application tests the successful operation of the proposed approach and produces the required results on the HyperTerminal. As is Fig.7, there is no corruption of data on serial communication lines. Our proposed approach Cyclic Redundancy Check i.e. CRC [5] on µC/OS-II platform has produced the result as *No Error Detected* on the HyperTerminal.

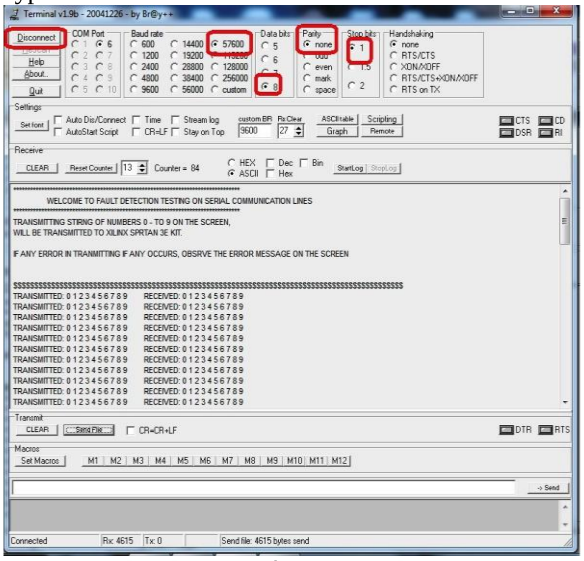

```
Terminal v1.9b - 20041226 - by Br@y++

COM Port: 6   Baud rate: 57600   Data bits: 8   Parity: none   Stop bits: 1   Handshaking: none
Settings: custom BR 9600   Rx Clear 27
Receive: 13   Counter = 84   ASCII

*******************************************************************************
          WELCOME TO FAULT DETECTION TESTING ON SERIAL COMMUNICATION LINES
*******************************************************************************
TRANSMITTING STIRNG OF NUMBERS 0 - TO 9 ON THE SCREEN,
WILL BE TRANSMITTED TO XILINX SPRTAN 3E KIT.

IF ANY ERROR IN TRANMITTING IF ANY OCCURS, OBSRVE THE ERROR MESSAGE ON THE SCREEN


$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
TRANSMITTED: 0 1 2 3 4 5 6 7 8 9     RECEIVED: 0 1 2 3 4 5 6 7 8 9
TRANSMITTED: 0 1 2 3 4 5 6 7 8 9     RECEIVED: 0 1 2 3 4 5 6 7 8 9
TRANSMITTED: 0 1 2 3 4 5 6 7 8 9     RECEIVED: 0 1 2 3 4 5 6 7 8 9
TRANSMITTED: 0 1 2 3 4 5 6 7 8 9     RECEIVED: 0 1 2 3 4 5 6 7 8 9
TRANSMITTED: 0 1 2 3 4 5 6 7 8 9     RECEIVED: 0 1 2 3 4 5 6 7 8 9
TRANSMITTED: 0 1 2 3 4 5 6 7 8 9     RECEIVED: 0 1 2 3 4 5 6 7 8 9
TRANSMITTED: 0 1 2 3 4 5 6 7 8 9     RECEIVED: 0 1 2 3 4 5 6 7 8 9
TRANSMITTED: 0 1 2 3 4 5 6 7 8 9     RECEIVED: 0 1 2 3 4 5 6 7 8 9
TRANSMITTED: 0 1 2 3 4 5 6 7 8 9     RECEIVED: 0 1 2 3 4 5 6 7 8 9
TRANSMITTED: 0 1 2 3 4 5 6 7 8 9     RECEIVED: 0 1 2 3 4 5 6 7 8 9
TRANSMITTED: 0 1 2 3 4 5 6 7 8 9     RECEIVED: 0 1 2 3 4 5 6 7 8 9

Connected   Rx: 4615   Tx: 0   Send file: 4615 bytes send
```

Fig.7 Serial Debug Log for Test application i.e.CRC (NO ERROR DETECTED)

As in Fig.8, we have introduced the faults using solenoid [11] by generating Electromagnetic Interference (EMI) and Power supply Disturbances on serial communication lines. Our proposed approach Cyclic Redundancy Check i.e. CRC [5] on µC/OS-II platform has produced the result as *CRC Detected the Error on the Received Byte* on HyperTerminal successfully.

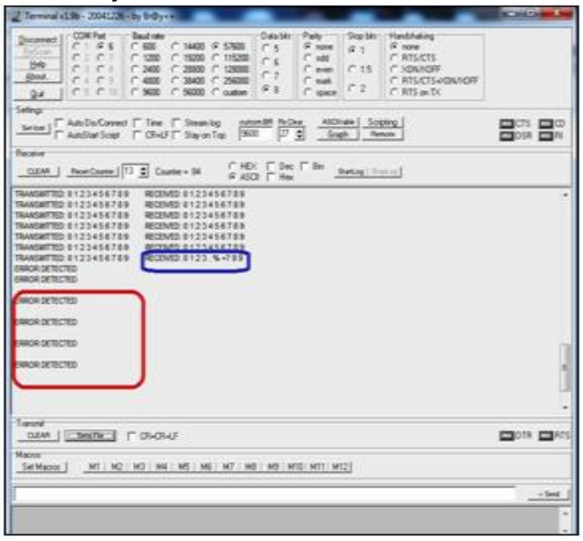

```
TRANSMITTED: 0 1 2 3 4 5 6 7 8 9     RECEIVED: 0 1 2 3 4 5 6 7 8 9
TRANSMITTED: 0 1 2 3 4 5 6 7 8 9     RECEIVED: 0 1 2 3 4 5 6 7 8 9
TRANSMITTED: 0 1 2 3 4 5 6 7 8 9     RECEIVED: 0 1 2 3 4 5 6 7 8 9
TRANSMITTED: 0 1 2 3 4 5 6 7 8 9     RECEIVED: 0 1 2 3 4 5 6 7 8 9
TRANSMITTED: 0 1 2 3 4 5 6 7 8 9     RECEIVED: 0 1 2 3 4 5 6 7 8 9
TRANSMITTED: 0 1 2 3 4 5 6 7 8 9     RECEIVED: 0 1 2 3 4 5 6 7 8 9
TRANSMITTED: 0 1 2 3 4 5 6 7 8 9     RECEIVED: 0 1 2 3 . % + 7 8 9
ERROR DETECTED
ERROR DETECTED

ERROR DETECTED

ERROR DETECTED

ERROR DETECTED

ERROR DETECTED
```

Fig.8 Serial Debug Log for Test application i.e.CRC (CRC DETECTED THE ERROR)

V. CONCLUSION AND FUTURE SCOPE

The main contribution of this paper consists of providing significantly more robust way of detecting the errors on serial communication lines on MIPS architecture with µC/OS-II RTOS, implemented and tested on Xilinx FPGA kit [9]. In this paper we proposed a Cyclic Redundancy Check (CRC) approach able to detect error occurrences on serial communication lines causing the data corruption. In general terms, the proposed approach targets transient errors affecting the data on serial bus, where it can be used for the scheduling process of the RTOS. It has been developed using cyclic check redundancy (CRC) to perform on-line detection of such type of faults. The main contribution of this paper consists of drastically improving the robustness of MIPS architecture based processor and µC/OS-II RTOS embedded systems operating in harsh environments like those where the electronics is exposed to conducted EMI noise. The proposed approach provides nearly 100% of fault coverage.

To conclude, we are convinced that the CRC-based approach proposed herein represents an important improvement to the state-of the- art of designing hardened embedded systems running on RTOSs. Future work includes correction of errors apart from the error detection, using error correction techniques such as hamming code etc. And also this can be extended to other serial communication buses like SPI, USB etc.

REFERENCES

[1] N. Ignat, B. Nicolescu, Y. Savari, G. Nicolescu, “Soft-Error Classification and Impact Analysis on Real-Time Operating Systems”, IEEE Design, Automation and Test in Europe, 2006.

[2] E. Touloupis, J. A. Flint, V. A. Chouliaras, D. D. Ward, “Study of the Effects of SEU Induced Faults on a Pipeline Protected Microprocessor”, IEEE TC, 2007.

[3] J. Freijedo, L. Costas, J. Semião, J. J. Rodríguez-Andina, M. J. Moure, F. Vargas, I. C. Teixeira, and J. P. Teixeira, “Impact of power supply voltage variations on FPGA-based digital systems performance”, Journal of Low Power Electronics, vol. 6, pp. 339-349, Aug. 2010.

[4] J. Arlat, Y. Crouzet, JU. Karlsson, P. Folkesson, E. Fuchs, G. H. Leber, “Comparison of Physical and Software-implemented Fault Injection Techniques”, IEEE Trans. on Computer, Vol. 52, N. 9, Sept., 2003.

[5] Peterson, W. W. and Brown, D. T. (January 1961). "Cyclic Codes for Error Detection". *Proceedings of the IRE* **49** (1):228235. doi:10.1109/JRPROC.1961.287814.

[6] Micrium Expands RTOS Family with µC/OS-II" (Press release). Micriµm, Inc. 2009-03-24. Retrieved 2010-02-14.
[7] Programming Languages Supported by GCC". GNU Project. Retrieved 2011-11-25.
[8] Denton J. Dailey (2004), *Programming Logic Fundamentals Using Xilinx ISE and CPLDs*, Pretince Hall, 203 pages. Introduction to PLDs using Xilinx ISE.
[9] Embedded Technology Journal, "Introducing the Xilinx Targeted Design Platform: Fulfilling the Programmable Imperative." Retrieved June 10, 2010.
[10] MIPS32 Architecture". MIPS Technologies. Retrieved 27 May 2009.
[11] D. Howard Dellinger, L. E. Whittmore, and R. S. Ould (1924). "Radio Instruments and Measurements". *NBS Circular* (National Bureau of Standards) **C74**. Retrieved 2009-09-07.

Author's biography

Saurabh Kumar, pursuing M.Tech in Digital systems and Computer Electronics (DSCE), Department of ECE, from Sreenidhi Institute of Science and Technology, JNTU-Hyderabad (A.P).

Narayanaraju Samunuri, M.Tech in Embedded Systems, with an experience of 11 years in Core Embedded System domain, published International paper on Flex ray Communication and Context Switching.