**Philemon Daniel[1], Rajeevan Chandel[2]**

National Institute of Technology Hamirpur, India (1, 2)

# Dynamic Ruleset Based Online Concurrent Testing of Functional Faults for Embedded Controllers

***Abstract***. *Concurrent fault testing is mandatory in critical embedded systems like automobile applications. Architectures of concurrent testing are proposed in the present work, where testing is non-intrusive. The concept that embedded program has several tasks is exploited here. The hardware overhead of the test architecture implemented on the controller of OC8051 is minimal. It is easily scalable. Error detection latency is kept at a few cycles and cumulative functional error coverage is 100%. The architectures are compared with the best of the existing methods.*

*Streszczenie. Współbieżne testowanie jest często obowiązkowe w systemach wbudowanych. W artykule zaprezentowano architektury tego typu systemów gdzie wbudowany program wykonuje wiele zadań. Analizowano opóźnienie w wykrywaniu błędów oraz uwarunkowania sprzętowe.* ***Online współbieżne testowanie błędów w kontrolerach wbudowanych***

**Keywords:** Design for testability, software based self-test, concurrent error detection, online test, embedded controllers
**Słowa kluczowe**: testowanie współbieżne, wykrywanie błędów.

doi:10.12915/pe.2014.01.26

## Introduction

As designs are pressed hard towards CMOS nanotechnologies, on-field testing has become a major area of research. For safety critical real time systems concurrent testing is mandatory especially for embedded hardware controllers. Transient errors are frequently appearing whose causes cannot be well established. Errors due to alpha particles which might result in a momentary fault, noise of various kinds that might creep into the circuit during runtime also add to the complexity. Process defects, extreme operating conditions, design compromises play a major role in contributing to on-field permanent faults. Online concurrent error detection becomes important to ensure the safety of the embedded systems. The problem is explored in depth for decades now and there are several methods suggested to tackle the problem [2]. The easiest approach is duplication, wherein two copies of the design under test (DUT) are put in the design to avoid one of the DUTs becoming faulty [3] including data repetition methods as in. The difficulty is the overhead which is more than one hundred percent. There has been another method suggested to avoid errors almost completely, where three copies of the DUT are used and the majority output is taken as the final. Here the overhead is even greater at 300%. These both are impractical because of the overhead. A better duplication method focusing on the critical portions of a circuit has been presented in reference [4]. Another common technique has been to use different codes, particularly within the framework of finite state machine (FSM) controllers. Structural changes have been exploited for micro circuits for concurrent self-checking whose overhead is again huge [5, 6]. When using FSMs different parity checks are implemented along the circuit which do detect the faults, but hardware overhead and performance compromises prohibit these methods from being practical [7]. Non-intrusive self-checking synthesis methods have been explored in references [8, 9]. Parity based schemes are given in [10, 11]. Algorithmic error detection methods for digital signal processing systems are also examined [12]. Several external monitoring schemes which could be implemented on FPGAs are discussed in [13, 14]. Instruction level impact of low level faults of control logic is also explored [15]. Authors in [16] have explored online test based on input monitoring schemes where the input set of test vectors are monitored for fault detection, however again the overhead is excessive. Functional test methods are explored in [17]. Reconfigurable architectures [18] and reprogrammable structures [19] include a lot of flexibility in online testing. Consequently, an attempt has been made in the present work to propose new test architectures for embedded controllers with minimal area overhead.

## Online concurrent testing assumptions

The principle behind the proposed online concurrent test architecture is presented here. In any embedded application, there is a main program cycle and several supplementary tasks that are executed to attend to the services in the application. The program is not often changed. The proposed method exhaustively checks for errors in the controllers for the program that is being executed. The method does not check for faults in the controller that are not activated by the program executed. This reasoning is justified by the fact that it is sufficient if the current embedded program runs error free for any on-field device. Subsequently it is essential to provide a non-intrusive test architecture that does not degrade the performance, maintains minimum hardware overhead even while scaling and has minimum error detection latency. The checking conditions are termed as the rule set. Once the program is loaded, for each instruction executed the control signal conditions of the controller can be relied upon to be true during any future execution of the program.

## Online Concurrent Test for Embedded Controllers (OCT-EC)

In many of the embedded applications each of the tasks does a small amount of work. Only some of the output states of the controller are used. Three schemes of dynamic ruleset based online concurrent testing of functional faults for embedded controllers namely OCT-EC1, OCT-EC2 and OCT-EC3 are presented in this section.

**OCT-EC1: (**One ruleset per task)

An embedded program consists of a main loop and various service routines (tasks). Since each task is usually a specific service there are a set of invariants. A ruleset is extracted for each task based on the invariants at the output of the controller. This extraction is explained with an assembly program to find the factorial of a number read from Port0. In Table 1, the first column lists the instructions in the routine. The first row lists the outputs of the decoder. All the remaining rows list the valid output values for the corresponding instruction. The last row shows the cumulative valid values at each output. A "-" indicates don't care at that bit position. Any violation of the ruleset by the output states are checked every cycle for the respective task. If there is a violation, depending on the severity a repair service routine can be initiated, the processor can be stalled.

Table 1. Example routine to find factorial to show OCT-EC1

| Instruction | ram_r (3) | ram_w (3) | wr | src_sel1 (3) | src_sel2 (2) | src_sel3 | alu_op (4) | Comp_sel (2) | Bit_addr | Pc_wr | Pc_sel (3) | Rmw | Istb | wr_sfr (2) | psw_sel (2) | cy_sel (2) | Rd |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Default | 000 | 000 | 0 | 000 | 00 | 0 | 0000 | 00 | 0 | 0 | 000 | 0 | 1 | 00 | 00 | 00 | 0 |
| MOV R1,80H | 000 | 000 | 1 | 000 | 00 | 0 | 0000 | 00 | 0 | 0 | 000 | 0 | 1 | 00 | 00 | 00 | 0 |
| MOV R0,80H | 000 | 000 | 0 | 000 | 00 | 0 | 0000 | 00 | 0 | 0 | 000 | 0 | 1 | 00 | 00 | 00 | 0 |
| MOV A,R0 | 000 | 000 | 0 | 000 | 00 | 0 | 0000 | 00 | 0 | 0 | 000 | 0 | 1 | 01 | 00 | 00 | 0 |
| CALL:DEC R0 | 000 | 000 | 1 | 000 | 10 | 0 | 1110 | 00 | 0 | 0 | 000 | 1 | 1 | 00 | 00 | 11 | 0 |
| MOV B,R0 | 000 | 000 | 0 | 000 | 00 | 0 | 0000 | 00 | 0 | 0 | 000 | 0 | 1 | 01 | 00 | 00 | 0 |
| MUL AB | 100 | 000 | 1 | 011 | 00 | 0 | 0011 | 00 | 0 | 0 | 000 | 0 | 0 | 10 | 10 | 00 | 0 |
| DJNZ R1,CALL | 000 | 000 | 0 | 000 | 00 | 0 | 0000 /1110 | 01 | 0 | 0/1 | 010 | 1 | 0 | 00 | 00 | 00/11 | 0 |
| RET | 011 | 000 | 0 | 000 | 00 | 0 | 0000 | 00 | 0 | 1 | 001 | 0 | 0 | 00 | 00 | 00 | 0 |
| Ruleset | --- | 000 | - | 0-- | -0 | 0 | ---- | 0- | 0 | - | 0-- | - | - | -- | -0 | -- | 0 |

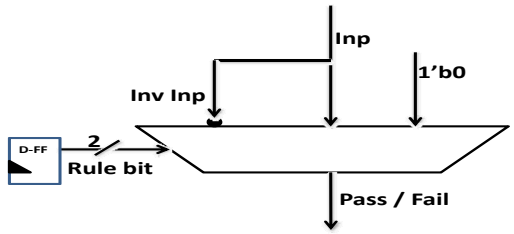


Fig. 2. Rule bit selects the mux input

The test architecture to monitor one bit is shown in Fig 2. D flip-flops store the rule bits that choose one of the three options through a multiplexer. The complete setup is shown in Fig. 3. The simplicity of the structure, concurrent fault detection, low error detection latency and the minimal hardware overhead are the key highlights of the architecture. The ruleset for each task is loaded at the beginning of the task from a dedicated local memory. The address of the first instruction of the task is used as the task identifier. The exit from the task is deducted either by RET or RETI instruction. The ruleset remains active through the entire task. The error coverage is not high enough for individual tasks though the cumulative coverage of several tasks is higher. This is because of the fact that the invariant for one instruction is not necessarily the same invariant for the next instruction within the task. For larger controllers with error coverage of individual tasks should be higher to guarantee higher overall error coverage.

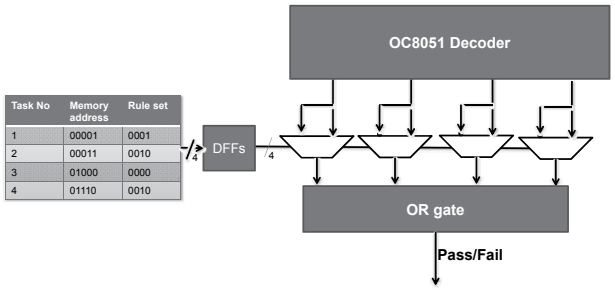


Fig.3. Architecture for OCT-EC1

**OCT-EC2: (**One ruleset for mutually inclusive and exclusive outputs for all tasks)

The output bits of the controller are grouped into categories according to their purposes. Their purposes have a direct correlation with the instructions. Only some of the instructions use each of them. These are 16 sets of outputs from the controller as shown in Table 2. Many of them are mutually exclusive and some of them are mutually inclusive. For example ram_write_source is mutually exclusive with ram_read_source. If the controller asserts ram_read_source it is actually selecting a source for read address. When the ram is being read, it would not be written to at the same time. Similarly ram_wr and rom_rd cannot be true at the same time.

Since 8051 is Harvard architecture, it uses a single bus to read data and code. An example for mutually inclusivity is with ram_wr and ram_write_source. When a ram is to be written to ram_write_source provides the source of the write_address and ram_wr signal is asserted. A combinational logic check is built to check any violations of both mutual exclusivity and mutual inclusivity. This method when implemented along with OCT-EC1 gives good error coverage at the controller output. The structure for the ram_write and ram_read mutual exclusivity is shown in Fig 4.

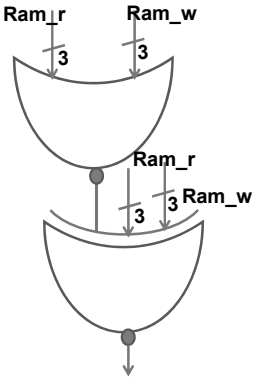


Fig.4. A sample mutually exclusive rule

Table 2. Decoder's output grouping

| | |
|---|---|
| 1. ALU source 1 | 9. Compare source |
| 2. ALU source 2 | 10. PSW set |
| 3. ALU source 3 | 11. ROM address source |
| 4. Write to SFR | 12. ROM read |
| 5. RAM read | 13. Write Accumulator |
| 6. RAM write | 14. Read modify write |
| 7. PC read | 15. PSW write |
| 8. PC write | 16. Carry input select |

**OCT-EC3:** (Group parity fed into CRC for each cycle of a task)

Each of the output groups shown in Table 2 is grouped and their group parity is extracted each cycle, passed on to a Cyclic Redundancy Check (CRC). 16bit Multiple Input Signature Register (MISR) is used for this purpose. The CRC signature is cumulatively computed for each instruction in the task. The signature at the end of the task is compared with a local look up table. Entry and exit points for each task are determined as in OCT-EC1.

The architecture is shown in Fig 5. At the output of the controller, each group is passed through a xor gate. These 16-bits are then fed to a 16-bit MISR. For complex entry exit routines, a counter is used to match with the task cycle count. The error detection latency is just a few cycles in the first two schemes. In this method it is a maximum of one task cycle. This structure can detect all single errors.

Double line errors can be a result of multiple internal faults or a single fault propagating through multiple lines. The probability of detecting double errors occurring in different groups is still high and it depends on the CRC polynomial chosen. All stuck at faults can be detected. The error coverage from this method is very good. The cumulative error coverage is usually 100%. Aliasing is negligible.

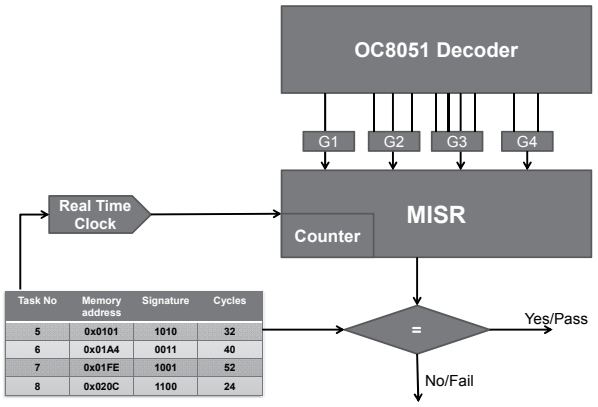


Fig.5. OCT-EC3 architecture with grouped outputs through MISR

**Experimental Results and Discussion**

For validation of the architectures these are implemented for the controller of the OC8051 microcontroller. Performance metrics for the three schemes proposed for online concurrent testing are presented in Table 3. Eight common tasks are chosen which include bulk data transfer from external memory and servicing the five interrupts. It is shown that using these architectures effective online concurrent testing can be performed. Methods 1 and 2 are important for safety critical systems where a single error could result in a catastrophe. The hardware overhead is well within accepted limits and these do not scale proportional to the size of the DUT. If a higher latency is tolerable method 3 is seen to give the best results. Any error on the output lines are captured within the execution of the same task. The cumulative error coverage reaches 100% within 3 tasks. But since there can be other sources of error, the architecture has to be run for all tasks and concurrent check is done at the end of each task. The hardware gate count is computed as shown in the table 3 with the parameters viz. i) Controller outputs to watch (N), ii) Tasks to test (T), iii) Bits of program counter are watched (M), iv) Groups of invariant outputs(G) and v) Bit counter to keep track of the per task execution cycles (C). A D-flip-flop (qDff) is taken to be equivalent to four gates.

The controller in 8051 which is the DUT has 3754 gates. For the cases investigated in the present work the hardware is computed in terms of number of gates as in Table 3. The overhead is nominal for all the three methods. However, method OCT-EC3 leads to minimal hardware overhead which is 19.21%. The hardware overhead in all the three proposed testing methods scales up only with increase in the number of tasks (T). Furthermore, if the number of tasks is very high, tasks with similar coverage can be omitted for concurrent test.

Table 3. Performance metrics for the schemes proposed

| Property | Proposed Methods | | |
|---|---|---|---|
| | OCT-EC1 | OCT-EC2 | OCT-EC3 |
| Hardware calculation | N Mux + N*T Dff + log T Mux + M Dff + 36gates (comparators) | Method1 + 2G gates | G gates + G Dff + G*T Dff + C Dff + M Dff + 72 gates (comparators) |
| Hardware overhead | 31.74% (N=32; T=8; M=8) | 32.59% (N=32;T=8; M=8;G=16) | 19.21% (N=32; T=8; M=8; G=16; C=8) |
| Error coverage | 53.42% (cumulative) | 71.31% (cumulative) | 100% (in 3 tasks) |
| Error detection latency | Same cycle | Same cycle | Task cycles after activation |
| Scalability | Scales with T | Scales with T | Scales with T |
| Performance overhead | Nil | Nil | Nil |
| Practical applicability | Yes | Yes | Yes |

Table 4. Comparison of the proposed schemes with those reported in literature

| Concurrent test methods | Hardware overhead (%) | Performance overhead | Error detection coverage (%) | Error detection latency | Design/ Code modification required | Scalability | Applicability in modern CPUs | Main highlights |
|---|---|---|---|---|---|---|---|---|
| DWC[20] | >200% | Yes | ~100% | Same cycle | Yes | Prohibitive | Nil | Generic but area overhead is too expensive |
| CIC [21] | Medium | Yes (>200%) | ~99% | ~80cycles | Yes | Medium | Yes | Excessive execution time overhead |
| SRS[22] | ~13% | Yes (~10%) | 100% | max 8ms | Yes | Prohibitive | Nil | Hardware/ software coupled architecture |
| IBCT[23] | ~41%(+control logic) | Yes | >90% | Same cycle | Yes | Prohibitive | Nil | Impractical for larger circuits |
| WC-CED[24] | 57% | Yes | ~81% | max 4 cycles | Yes | Medium | Yes | For superscalar schedulers; coverage non deterministic |
| CASP[25] | Medium | Yes | ~100% | Several seconds | Yes | Prohibitive | Yes | Involves hardware/ software/external storage |
| CED-TPOC[26] | ~200% | Yes | ~80% | Same cycle | Yes | Prohibitive | Nil | Dissimilar duplicity |
| OCT-EC1 (Proposed) | 31.74% | Nil | 53.42% | Same cycle | Nil | Very good | Yes | Generic structure |
| OCT-EC2 (Proposed) | 32.59% | Nil | 71.31% | Same cycle | Nil | Very good | Yes | Generic structure |
| OCT-EC3 (Proposed) | 19.21% (low) | Nil | 100% (error coverage) | No. of cycles in a task (max) | Nil | Very good | Yes | Generic structure |

A comparison of the proposed methods with the best of the existing methods reported in literature in this category is presented in Table 4. It is shown that only a few methods have applicability in present day architectures. There is no performance overhead or design/code changes required for the proposed methods. They are also generic in nature i.e. they can be implemented for any controller. Consequently these are flexible and scalable without a proportional scaling of area overhead. OCT-EC3 is the best amongst the methods considered in the present analysis. This is because the hardware overhead is low, there is no performance overhead and it gives 100% line error coverage. Moreover, design or code modification is not required, scalability is very good and it is applicable for modern CPUs. Its fault detection latency is a few cycles. The advantage is significant because of the division of the code into smaller tasks at the hardware level and each task is verified individually for the controller.

**Conclusions**

Three test architectures are proposed in this work for control logic of embedded processors. They apply dynamic on-field testing and are concurrent in nature where the normal execution of the processor is not affected. Since the embedded program is divided into tasks and each task is verified individually, the error detection latency is kept high and the hardware overhead is minimal. Two of them exhibit the best latency possible. One of them gives 100% line error coverage with a maximum latency of the number cycles in the current task. The results are encouraging and the work is further being explored to test other controllers.

REFERENCES


[1] M. Goessel, and S. Graf, Error Detection Circuits, McGraw-Hill, 1993.

[2] S. Mitra, and E. J. McCluskey, “Which Concurrent Error Detection Scheme to Choose?,” in Proc. of the International Test Conference, 2000, pp. 985–994.

[3] A. Avizienis, and J. P. J. Kelly, “Fault Tolerance by Design Diversity: Concepts and Experiments,” IEEE Transactions on Computers, vol. 17, no. 8, pp. 67-80, 1984.

[4] K. Mohanram, and N. A. Touba, “Cost-Effective Approach for Reducing Soft Error Rate in Logic Circuits,” in Proc. of the International Test Conference, 2003, pp. 893–901.

[5] G. Aksenova, and E. Sogomonyan, “Design of Self-Checking Built-in Check Circuits for Automata with Memory,” Automation and Remote Control, vol. 36, no. 7, pp. 1169–1177, 1975.

[6] S. Dhawan, and R. C. D. Vries, “Design of Self-Checking Sequential Machines,” IEEE Transactions on Computers, vol. 37, no. 10, 1988, pp. 1280–1284.

[7] C. Zeng, N. Saxena, and E.J. McCluskey, “Finite State Machine Synthesis with Concurrent Error Detection,” in Proc. of the International Test Conference, 1998, pp. 672–679.

[8] D. Das, and N. A. Touba, “Synthesis of Circuits with Low-Cost Concurrent Error Detection Based on Bose-Lin Codes,” Journal of Electronic Testing: Theory and Applications, vol. 15, no. 2, 1999, pp. 145–155.

[9] N. K. Jha and S.-J. Wang, “Design and Synthesis of Self-Checking VLSI Circuits,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 12, no. 6, 1993, pp. 878–887.

[10] R. A. Parekhji, G. Venkatesh, and S. D. Sherlekar, “Concurrent Error Detection Using Monitoring Machines,” IEEE Design and Test of Computers, vol. 12, no. 3, 1995, pp. 24–32.

[11] S. Almukhaizim, P. Drineas, and Y. Makris, “Entropy-Driven Parity-Tree Selection for Low-Overhead Concurrent Error Detection in Finite State Machines,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 25, no. 8, 2006, pp. 1547-1554.

[12] Costas-Perez, L.; Rodriguez-Andina, J.J., "Algorithmic Concurrent Error Detection in Complex Digital-Processing Systems," *Design & Test of Computers, IEEE* , vol.26, no.1, pp.60,67, Jan.-Feb. 2009

[13] Mandjavidze, I.; Romanteau, T., "Embedding online test and monitoring features in real time hardware systems," *Real Time Conference (RT), 2010 17th IEEE-NPSS* , vol., no., pp.1,8, 24-28 May 2010

[14] Abdelfattah, M.S.; Bauer, L.; Braun, C.; Imhof, M.E.; Kochte, M.A.; Hongyan Zhang; Henkel, J.; Wunderlich, H., "Transparent structural online test for reconfigurable systems," *On-Line Testing Symposium (IOLTS), 2012 IEEE 18th International* , vol., no., pp.37,42, 27-29 June 2012

[15] M. Maniatakos, N. Karimi, C. Tirumurti, A. Jas, and Y. Makris, “Instruction-Level Impact Analysis of Low-Level Faults in a Modern Microprocessor Controller,” IEEE Trans. Computers, vol. 60, no. 9, pp. 1260-1273, 2011.

[16] I. Voyiatzis, A. Paschalis, D. Gizopoulos, C. Halatsis, F.S. Makri, and M. Hatzimihail, "An Input Vector Monitoring Concurrent BIST Architecture Based on a Precomputed Test Set," IEEE Trans. on Computers, vol. 57, no. 8, pp. 1012-1022, Aug. 2008

[17] J. Shen and J. Abraham, “Native mode functional test generation for processors with applications to self test and design validation,” in *Proc. IEEE Int. Test Conf.*, Oct. 1998, pp. 990–999.

[18] Sandeep Sharma, Ashutosh Gupta, ManojDuhan and Solomon Raju Kota, "Design of Partially Reconfigurable Computing System and Implementation on Virtex-4 FPGA," The IUP Journal of Science & Technology, Vol. 6, Sept., 2010

[19] P. Philemon Daniel and RajeevanChandel, “A Flexible *Programmable*. Memory BIST Architecture,” IETE Journal of Education, vol. 51, pp. 67-74, Dec 2010.

[20] Johnson, B.W.; Aylor, J.H.; Hana, H.H., "Efficient use of time and hardware redundancy for concurrent error detection in a 32-bit VLSI adder," *Solid-State Circuits, IEEE Journal of* , vol.23, no.1, pp.208,215, Feb. 1988

[21] Rajabzadeh, A.; Mohandespour, M.; Miremadi, G., "Error detection enhancement in COTS superscalar processors with event monitoring features," *Dependable Computing, 2004. Proceedings. 10th IEEE Pacific Rim International Symposium on* , vol., no., pp.49-54, 3-5 March 2004

[22] Khan, O.; Kundu, S., "Hardware/Software Codesign Architecture for Online Testing in Chip Multiprocessors," *Dependable and Secure Computing, IEEE Transactions on* , vol.8, no.5, pp.714-727, Sept.-Oct. 2011

[23] Makris, Y; Bayraktaroglu, I.; Orailoglu, A., "Enhancing reliability of RTL controller-datapath circuits via Invariant-based concurrent test," *Reliability, IEEE Transactions on* , vol.53, no.2, pp.269- 278, June 2004

[24] Karimi, N.; Maniatakos, M.; Jas, A.; Tirumurti, C.; Makris, Y, "Workload-Cognizant Concurrent Error Detection in the Scheduler of a Modern Microprocessor," *Computers, IEEE Transactions on*, vol.60, no.9, pp.1274,1287, Sept. 2011

[25] Li, Y; Makar, S.; Mitra, S, "CASP: Concurrent Autonomous Chip Self-Test Using Stored Test Patterns," *Design, Automation and Test in Europe, 2008. DATE '08* , pp.885,890, 10-14 March 2008

[26] Khedhiri, C.; Karmani, M.; Hamdi, B.; KaLok Man, "Concurrent Error Detection Adder Based on Two Paths Output Computation," *Parallel and Distributed Processing with Applications Workshops (ISPAW), 2011 Ninth IEEE International Symposium on* , pp.27,32, May 2011



***Authors**: Contact author: Mr. Philemon Daniel, NIT Hamirpur, Hamirpur – 177005, Himachal Pradesh, India. Email: phildani7@gmail.com*

*Dr. Rajeevan Chandel, NIT Hamirpur, Hamirpur – 177005, Himachal Pradesh, India. Email: rchandel@nith.ac.in*