ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Self Immunity Technoque to Improve Register File Integrity against Soft Errors

Bungai Karishma Kour Harjeet Singh
M.Tech (VLSI),
Avanthi Scientific Technological and Research Academy, Approved By AICTE, Registered By AP And,Affiliated To JNTU, Hyderabad.

Mr.N.Ashok Kumar, M.Tech
Associate Professor,
Dept of ECE,
Avanthi Scientific Technological and Research Academy, Approved By AICTE, Registered By AP And,Affiliated To JNTU, Hyderabad.

Abstract:
Ceaseless contracting in highlight size, increasing power thickness and so forth increment the powerlessness of chip against delicate mistakes even in earthbound applications. The register record is one of the key engineering segments where delicate mistakes can be extremely wicked on the grounds that blunders may quickly spread from that point all through the entire framework. Accordingly, enroll documents are perceived as one of the significant concerns with regards to dependability. This paper presents Self-Immunity, a method that enhances the honesty of the register document regarding delicate mistakes. Taking into account the perception that a specific number of register bits are not generally used to speak to a quality put away in a register. This paper manages the trouble to adventure this undeniable perception to improve the register record trustworthiness against delicate blunders. We demonstrate that our system can diminish the powerlessness of the register document extensively while displaying littler overhead as far as zone and power utilization contrasted with best in class in register record insurance.

I. INTRODUCTION:

Over the last decade, and in spite of the progressively mind boggling models, and the quick development of new innovations, the innovation scaling has raised delicate blunders to end up one of the significant hotspots for processor smashing in numerous frameworks in the Nano scale time.

Delicate mistakes brought on by charged particles are hazardous principally in high-air, where substantial alpha particles are accessible [1]. Nonetheless, slants in today's nanometer advancements, for example, forceful contracting have made low-vitality particles, which are more superabundant than high-vitality particles, cause proper charge to incite a delicate blunder. Moreover, there is an overarching expectation that delicate mistakes will turn into a reason for a forbidden blunder rate issue soon even in terrestrial applications [2]. Scientists have essentially and generally centered around alleviating delicate blunders in memory and reserve structures [4][5][13], because of their extensive sizes. Then again, generally little work had been directed for register records in spite of the fact that they are exceptionally helpless against delicate mistakes [8].

Regardless of the general rather little zone impression of the register record, it is gotten to more oftentimes than whatever other structural part [6][9]. Along these lines, debased information in any register, if not dealt with, may proliferate quickly all through alternate parts of processor, prompting exceptional framework unwavering quality issues [6]. Truth be told, delicate mistakes in register records can be the reason for a substantial number of framework disappointments [10]. As of late, Blome et al. [8] demonstrated that a lot of issues that influence a processor more often than not originate from the register document. Subsequently, a few processors ensure their registers with Error Correction Code (ECC) [11], yet such

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

arrangements might be restrictive in certain applications (like installed) because of the noteworthy effect as far as range and power [14]. Besides, control utilization was routinely a noteworthy worry in installed frameworks because of their extensive impacts on the framework. To overcome any issues, there is a bothered need of systems to build the register document respectability against delicate mistakes with a little impact on both territory and power overhead. This paper addresses this test by presenting a novel strategy, called Self-Immunity to enhance the strength of register records to delicate mistakes, particularly attractive for processors that interest high enrolls document trustworthiness under stringent requirements.

Our contributions within this paper are as follows:

(1) We introduce a procedure for enhancing the resistance of register records against delicate blunders by putting away the ECC in the unused bits of a register.

(2) We take care of the issue of the territory and power overhead that normally comes as a negative reaction in register file insurance by accomplishing high range and power sparing with a slight corrupting in the register document weakness decrease (7%) contrasted with a full security plan.

Whatever remains of this paper is sorted out as takes after. Area 2 condenses the past work while Section 3 shows our proposed strategy. Area 4 displays the usage points of interest and Section 5 assesses the register document powerlessness decrease and gives a correlation with the cutting edge. At last, Section 6 finishes up the paper.

II. RELATED WORK AND BACKGROUND

The most punctual plans of register record assurance, for example, Triple Modular Redundancy (TMR) and ECC can accomplish an abnormal state of adaptation to internal failure however they may not be reasonable arrangements in implanted frameworks because of their energy and zone overheads. As of late, Fazeli et al. [14] demonstrated that ensuring the entire register document with SEC-DED accompanies around 20% power overhead. The proposed approach in [15] uses the Cross-equality check as a strategy for redressing different mistakes in the register documents. Spica et al. [16] demonstrated that there is an almost no addition (only 2%) in adaptation to internal failure for reserves on the off chance that they increment the security to Double Error Correction while the overhead for that increase is impressive. Expanding on the idea of Architectural Vulnerability Factor (AVF), presented by Mukherjee [3], Yan et al. [19] proposed the Register Vulnerability Factor (RVF) to portray the probability that a delicate mistake in registers can be spread to other framework parts. When all is said in done, a quality is built into a register, then it is perused much of the time, and later another worth is composed once more. In this manner, any delicate blunder happening amid "compose" or "read-compose" interims will have no impact on the framework; since it will be revised consequently by the following compose operation. Then again, "compose read" and "read-read" interims are viewed as helpless interims as is portrayed in Fig. 1. The RVF of a register is characterized as the total of the lengths of all its helpless interims partitioned by the total of the lengths of every one of its lifetimes [19].

$$Vulnerability(reg) = \frac{\sum Vulnerable\ Interval\ Time}{\sum LifeTime}$$

At long last, the aggregate helplessness of the register document is expected as the whole of powerlessness of all registers. [21].

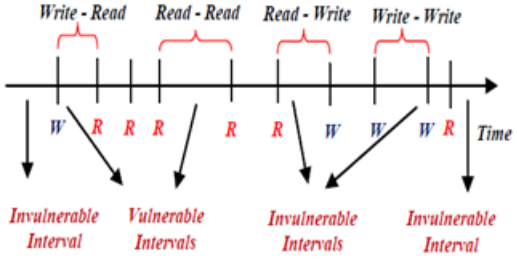

Fig. 1.Different Register Access Intervals [19].

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

The unadulterated programming approach at aggregate level presented by Yan et al. [19] re-plans the directions keeping in mind the end goal to diminish the RVF of a register document however the proposed system is not generally exceptionally powerful on the grounds that it might build the execution cycles and even the RVF in a few benchmarks [19]. In an offer to decrease the territory and force punishments, Yan et al. [19] proposed to secure a subset of the registers rather than full assurance plots and adjust the register distribution calculation to dole out the most delicate registers against delicate blunders to the ensured registers. The accomplished RVF diminished is 23%, 41%, 67% and 93% for securing 2, 4, 8 and 16 out of 32 registers individually. Montesano's et al. [9] settle on a choice of which register qualities ought to be ensured at runtime by equipment rationale however the run time expectation is immoderate as far as vitality [22]. Lee et al. [7] exhibited an assemble method to diminish RVF by securing a little piece of memory and compose the defenseless enlist values in this memory by embedding load/store guidelines yet it increments both run time and code size.

Another imperative methodology is In-Register Replication "IRR" [17], which misuses the way that a huge portion of register qualities are not exactly or equivalent to 16 bits wide for 32-bit structures. Such values can be imitated in the same register for expanding the invulnerability against delicate blunders. The principal struggle is that, while augmenting register record in susceptibility against delicate mistakes by diminishing the defenselessness of the register document, this lessening (either with full or halfway assurance plans) builds the range and power overheads.

III. PROPOSED SELF-IMMUNITY TECHNIQUE

We propose to misuse the register values that don't require the majority of the bits of a register to speak to certainvalue. At that point, the upper unused bits of a register can be misused to expand the register's resistance by putting away the relating SEC Hamming Code [11] without the requirement for additional bits. The Hamming Code is characterized by k, the quantity of bits in the first word and p, the requiredthe required number of equality bits (roughly log2K).Thus, the codewill be (K + log2K + 1) [23]. In our proposedtechnique, the ideal estimation of k is the quality which ensures that w, the bit - width of the register document, can cover both k, the required number of bits to speak to the worth, and the comparing ECC bits of that worth. At the end of the day, the worth and its ECC ought to be put away together inside the bit-width of a register. Therefore, the accompanying condition ought to be substantial ($K+ \log2K + 1 \leq W$). Along these lines, the ideal estimation of k is 26 in 32-bit designs and 57 in 64-bit architectures.And57 in 64-bit architectures.For occasion, when examining 32-bit structures, where every register can speak to a 32-bit esteem, we may misuse the register values, which require not exactly or equivalent to 26 bits by putting away the comparing ECC bits in the upper unused six bits of that register to improve the register record invulnerability against delicate errors1.

We call this procedure Self-Immunity and we call such values"26-bit"values. On theother hand, we call register values which require more than 26 bits to be spoken to "more than 26-bit" register values. Fig. 2 demonstrates the rate of register qualities utilization for various uses of the MiBench Benchmark [12] ordered for MIPS architectureAs it can be seen, in all benchmarks the vast majority of the register qualities are "26-bit" values. As such, the upper six bits of 88% of the put away information in the register record are really unused. Thusly, we can store the comparing ECC in these accessible bits and increment the register's resistance. In addition to the previous key observation, the contribution of “26-bit” register values in the total vulnerable intervals is much more than the contribution of“over-26-bit” register values. In Fig. 3, the fraction of vulnerable intervals of each benchmark is reported. As is demonstrated, the part of helpless interims of "26-bit" qualities is 93% by and large.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

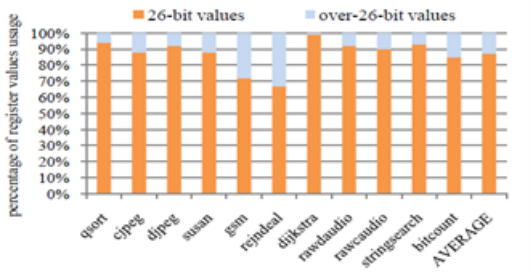

Fig. 2. "26-bit" register values and "over-2 bit" register values I different benchmarks.
Fig. 3. The fraction of vulnerable intervals of "26-bit" register values and "over-26-bit" register values in different benchmarks.

A. Problem Description:

1) Goal: The objective of our procedure is to diminish the register record weakness with least effect on both region and power overhead. Give N a chance to be the aggregate number of registers and V the powerlessness of a register, then the helplessness of the register document is ($\sum_{i=1}^{N}$ Vi). Since the force overhead2 for the most part originates from getting to the encoder and decoder, it can roughly be displayed through the quantity of gets to [22]. Let M is the quantity of secured register values and A the quantity of gets to, then the aggregate power overhead can be evaluated as ($\sum_{i=1}^{M}$ Ai). Thus the general objective can be figured as Minimize p= ($\sum_{i=1}^{N}$ Vi).,Minimize p=($\sum_{i=1}^{N}$ Vi).

2) Effectiveness of Our Technique:

In a full insurance plot, an ECC era is performed with each compose operation and comparably ECC checking is performed with every perused operation. Our system chooses to ensure the quality depending in the event that it is substantial for self-Immunity, then it actuates the ECC generator to figure the ECC bits. Something else, the ECC era is skipped. So also, on each register read operation, rather than continually checking ECC, our strategy checks whether the ECC is being inserted in the register esteem, and just in the event that it is, ECC checking is performed. As is shown in Fig. 2, all things considered 12% of the information will be put away in the register document without insurance. Accordingly, our strategy diminishes M and it might prompt decrease the devoured power. As is appeared in Fig. 3, when contemplating 32-bit models, 93% (by and large) of the aggregate vulnerable intervals are powerless interims of legitimate register values for our strategy. At the end of the day, around 93% of vulnerable intervals will conceivably be immune. In this way, our technique guarantees to lessen the helplessness of the register record considerably. Engineering for Our Proposed Technique. The key test in recognizing whether the ECC bits are inserted in the register esteem or not, is that the processor does not have adequate data to settle on this choice when perusing a worth from a register. Therefore, we have to recognize "26-bit" register values from "more than 26-bit" register values. To do that, a self-π bit is connected with every register and we at first clear all self-π bits to show the nonappearance of any Self - Immunity. For effortlessness, we clarify the proposed design with the required calculations in two unique strides.

Writing into a register:

Fig. 4 represents that whenever an guideline composes a quality into a register it checks the upper six bits of that worth on the off chance that they are "0" or not. In the event that they are (26 - bit register esteem case), the relating self-π bit is set to "1" showing the presence of Self-Immunity. The ECC worth is produced and put away in the upper unused bits of the register. Consequently, the information worth and its ECC are put away together in that register. In the second case (over - 26-bit register esteem), the relating self-π bit is set to "0" and the worth is built into the register without encoding.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

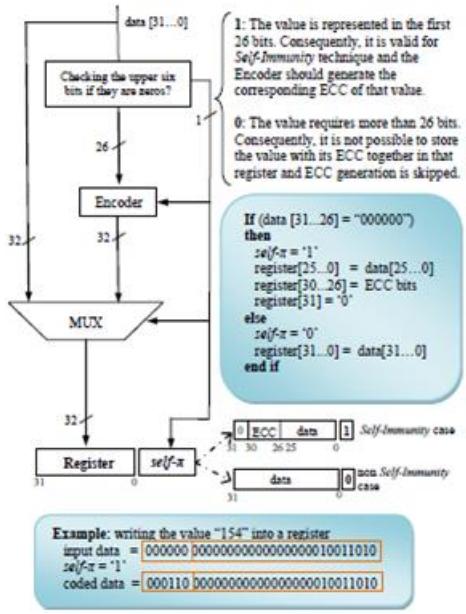

Fig. 4.Microarchitectural support for writing a register value.

Reading from a register:

In read operations, this lf-πbit is utilized to recognize a Self-Immunity case and a non self-Immunity case. In the principal case, the worth and the comparing ECC are put away together in that register and thus the read quality ought to be decoded. In the second case, the put away esteem is not encoded and thus there is no should be decoded as is shown in Fig.5.

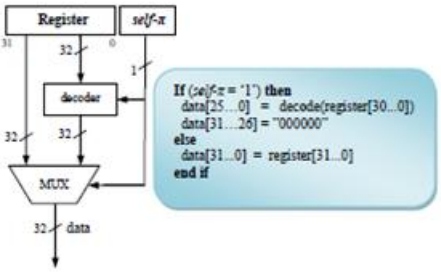

Fig. 5.Microarchitectural support for reading a register value.

C. Potential Power Saving:

In this area we clarify why our proposed design guarantees to expend less power. In our proposed engineering, "more than 26-bit" register qualities are neither encoded nor decoded and thus the encoding and translating operations are not performed with every perused and compose operation as it happens in a full insurance plan. This may lessen the force utilization of our proposed engineering in light of the fact that the encoding and deciphering operations are performed just on account of "26-bit" register values. Fig. 6 exhibits that by and large 12% and 13% of the aggregate number of read and compose operations, individually, are happened on account of "more than 26 - bit" register values. Therefore, our proposed design may expend less power in light of the fact that the encoder and decoder are lesser times got to.

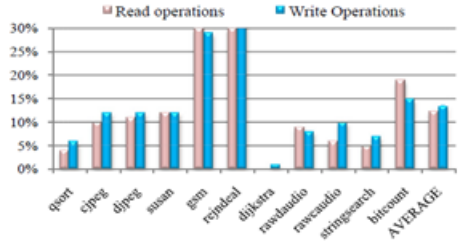

Fig. 6. The percentage of read and write operations in the case of "over-26-bit" register values.

Since the contribution of the sent encoder in our design is 26 bits rather than 32 bits, it produces 5 equality bits rather than 6 equality bits. In like manner, the utilized decoder as a part of our design takes 31 bits (26 bits for information + 5 bits for ECC) as a contribution rather than 38 bits. At the end of the day, our proposed design utilizes a less intricate encoder and decoder. This may likewise prompt a further sparing in the terms of the force utilization. At last, our proposed design lessens the aggregate number of bits of a shielded register from 38 bits to 33 bits and thus the devoured exchanging force is lower. To put it plainly, the force sparing is mostly because of the less ECC operations, the utilization of a less minds boggling ECC generator and checker, and the nonattendance of extra stockpiling for ECC.

IV.IMPLEMENTATION DETAILS:

Since the likelihood of numerous piece mistakes is to a great extent lower than the single piece - blunder [20], a solitary piece mistake model has been considered in this paper.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

In our issue infusion environment, flaws are infused on the fly while the processor executes an application. In every flaw infusion reproduction, one of the 32 registers is chosen arbitrarily and a bit in that register is picked haphazardly and after that flipped. Notice that a compose operation gets out the past infused blunder into that register. In like manner, by utilizing a uniform dispersion, an arbitrary cycle is picked as the time that delicate blunder happens. This ensures the issues will be infused just when the system is executed [20]. Since an infused shortcoming may deliver an interminable circle, a guard dog clock was actualized for the required number of execution cycles.

We stop the reproduction when the cycle check surpasses two times the quantity of cycles in the flaw free case. Towards assessing our proposed strategy, we utilize distinctive applications from MiBench Benchmark aggregated for MIPS engineering [12] to check diverse conceivable situations for register usage. Recreations were directed utilizing the MIPS model test system [18]. At the point when a recreation ends, the relating yield data (last results, substance of the register document, execution time and condition of the processor) are put away and used to order the reenactment. For the arrangement, we abuse the accompanying classes proposed in [10] [20]:

- Wrong Answer: The application ends normally but the outcomes created are not right.
- Latent: The application ends ordinarily, the results are amend yet toward the end of reenactment the substance of the register document are unique in relation to that of flaw free case.
- Effect-Less: The application ends typically, the outcomes are right, and the substance of the register document is like that of issue free case.
- Exception: The processor distinguished the infused blame and created a special case (e.g., invalid location exemption).
- Timed-Out: The application neglected to end and deliver results with a predefined time limit.
- Stalling: The processor figured the normal results in a period more prominent than the season of flaw free case.
- Crashing: The processor neglects to end typically.

Every benchmark was mimicked 10,000 times. Thus, 10,000 delicate mistakes were infused haphazardly in the register record. This number follows those utilized by other examination to keep the aggregate time of recreations sensible. For a reasonable correlation, we consider three models of the processor:

Base: a typical processor (without actualizing any security strategy).

IRR: a fault tolerant model, where an In-Register Replication strategy [17] is executed. This strategy has been picked here on the grounds that it tries to accomplish a comparative objective as our proposed system.

SI: a fault tolerant version, where our proposed technique, Self-Immunity, is implemented.

V. EXPERIMENTAL RESULTS AND EVALUATION

Of course, our proposed strategy keeps up elevated amounts of adaptation to internal failure contrasted with the "Base" case. As is delineated in Table 1, our proposed strategy enhances the register record respectability adequately by lessening to a great extent the quantity of blunders in every classification. Besides, the quantity of mistakes achieves zero in a few benchmarks.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

All things considered, our proposed method lessens the quantity of blunder by 100%, 87%, 93%, 93%, and 100% for the accompanying classes: Exception, Timed-Out, Crashing, Wrong Answer, and Stalling individually as is appeared in Fig. 7.Since dormant blunders have no impact on the yield of an application, and they are less destructive. This implies we can securely include the "Idle" classification to the "Impact less" class [20] since in both classes the last results are still totally right. For this situation, all things considered the framework deficiency scope in the wake of actualizing our strategy comes to all things considered 98% and up to 100% as is appeared in Fig. 8. Towards further assessment the impact of our method regarding Register Vulnerability Factor (RVF), which is a broadly utilized metric [7][9][19], Fig. 9 demonstrates that the potential RVF decrease is constantly high. It achieves 93% by and large and up to 100%. Moreover, we accomplish the best result contrasted with the IRR procedure [17]. As specified before in Section 2, Yan et al. [19] proposed a halfway ECC insurance strategy as opposed to securing the entire register record "Completely ECC" to accomplish.

Table. 1. Processor behavior for single error injection after implementing our proposed technique.

		cjpeg	djpeg	bitcount	qsort	rawdaudio	rawca-udio
Effect-Less	Base	5934	5801	5118	6415	5607	2655
	SI	9011	8961	8501	8751	7842	7768
Latent	Base	1113	1241	826	1164	896	1916
	SI	965	965	953	1249	2006	2062
Wrong Answer	Base	24	42	872	192	294	1434
	SI	2	13	0	0	0	2
Timed-Out	Base	180	736	801	22	1038	713
	SI	15	44	377	0	152	11
Stalling	Base	117	10	5	57	285	1917
	SI	1	0	0	0	0	0
Exception	Base	1646	1461	1567	1063	1464	335
	SI	0	0	0	0	0	0
Crashing	Base	926	621	811	1105	416	1030
	SI	6	17	169	0	0	157

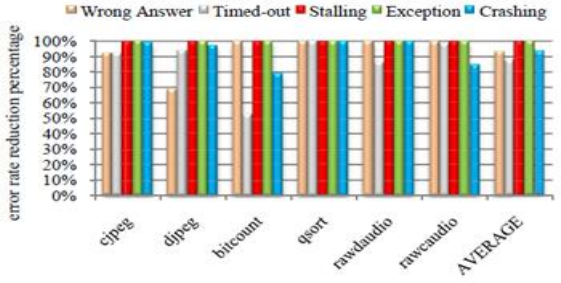

Fig. 7. Percentage of error rate reduction after implementing our proposed technique.

Since inactive mistakes have no impact on the yield of an application, they are less destructive. This implies we can securely include the "Inactive" classification to the "Impact Less" class [20] since in both classes the last results are still totally right. For this situation, all things considered, the framework issue scope in the wake of actualizing our procedure comes to by and large 98% and up to 100% as is appeared in Fig.8. Towards further assessment the impact of our system regarding Register Vulnerability Factor (RVF), which is a generally utilized metric [7][9][19], Fig. 9 demonstrates that the potential RVF lessening is constantly high. It achieves 93% overall and up to 100%. In addition, we accomplish the best result contrasted with the IRR strategy [17]. As said before in Section 2, Yan et al. [19] proposed a fractional ECC security method as opposed to ensuring the entire register document "Completely ECC" to achieve area and power sparing while expanding the register file vulnerability reduction.

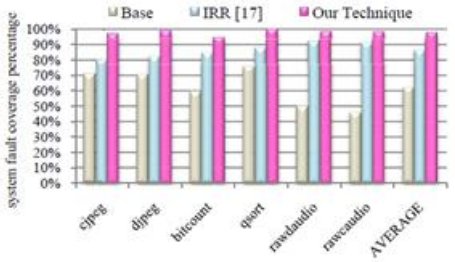

Fig. 8. System fault coverage comparison for different benchmarks.

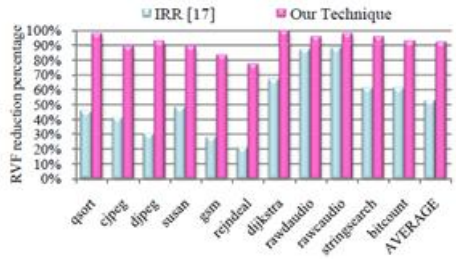

Fig. 9.Comparison of Register Vulnerability Factor Reduction.

To examine the benefits of utilizing our proposed strategy as a part of terms of territory overhead against "Completely ECC" and against the mostly assurance, we executed and integrated for a Xilinx XC2V600 distinctive variants of a 32-bit, 32-section, double read

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

ports, single compose port register record. Fig. 10 demonstrates the examination results as far as RVF lessening and region overhead. As is seen, our method accomplishes a decent territory sparing with slight corruption (7%) in the register record defenselessness lessening contrasted with "Completely ECC". Moreover, securing 16 out of 32 registers "16ECCs" can accomplish comparative RVF decrease to our outcome however our procedure involves 31% less zone. Then again, securing 4 registers "4ECCs" accompanies a range overhead comparative as our strategy yet our method accomplishes 1.3X change as far as RVF decrease.

Since the fundamental focus of this paper is 32-bit inserted processors, a synthesizable VHDL model of the DLX processor is utilized to research the execution and force punishments for every system. Additionally the Xpower device from Xilinx is utilized to appraise the aggregate force utilization in each of the distinctive processor variants for the adcpm decoder benchmark application. Since the utilized encoder and decoder are fewer minds boggling as clarified before, the basic way in our proposed design is shorter. Thus, our strategy enhances the execution contrasted with different contenders. As indicated Fig. 11, our strategy accompanies a base effect on both execution and force.

It accomplishes 54% postponement decrease and devours with 94% less power contrasted with "Completely ECC". Besides, securing 16 out of 32 registers "16 ECCs" accomplishes comparable RVF diminishment as specified some time recently; however our method accomplishes a 47% execution change and devours 87% less power. Then again, our system devours 75% less power and accomplishes 29% change as far as postponement overhead contrasted with "4ECCs" 4. It can be inferred that our system accomplishes the best general result contrasted with cutting edge in register document defenselessness diminishment.

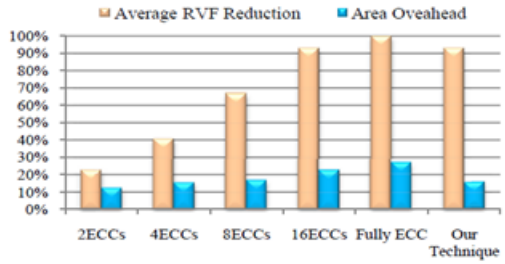

Fig. 10. Reduction of RVF (Register Vulnerability Factor) and area overhead.

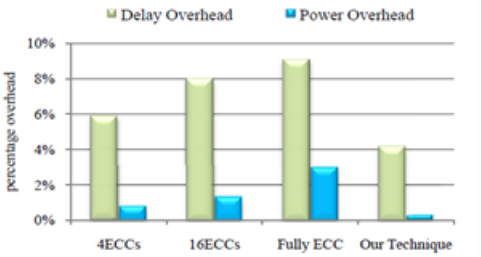

Fig. 11.The performance and power overhead comparison.

VI. CONCLUSION:

For implanted frameworks under stringent cost imperatives, where region, execution, force and unwavering quality can't be just traded off, we propose a delicate blunder moderation procedure for register documents. Our analyses on various implanted framework applications exhibit that our proposed Self-Immunity strategy diminishes the register document helplessness successfully and accomplishes high framework shortcoming scope. In addition, our system is non specific as it can be actualized into assorted models with least effect on the expense.

VII.REFERENCES:

[1] Greg Bronevetsky and Bronis R. de Supinski, ”Soft Error Vulnerability of Iterative Linear Algebra Methods,” in the 22nd annual international conference on Supercomputing, pp. 155-164, 2008.

[2] J.L. Autran, P. Roche, S. Sauze, G. Gasiot, D. Munteanu, P. Loaiza, M. Zampaolo and J. Borel, “Real-time neutron and alpha soft-error rate testing of CMOS 130nm SRAM: Altitude versus

underground measurements," in ICICDT'08, pp. 233–236, 2008.

[3] S.S. Mukherjee, C. Weaver, J. Emer, S.K. Reinhardt and T. Austin, "A Systematic Methodology to Compute the Architectural Vulnerability Factors for a High-Performance Microprocessor," in International Symposium on Microarchitecture (MICRO-36), pp. 29-40, 2003.

[4] T.J. Dell, "A whitepaper on the benefits of Chippkill-Correct ECC for PC server main memory," in IBM Microelectonics division Nov 1997.

[5] S. Kim and A.K. Somani, "An adaptive write error detection technique in on-chip caches of multi-level cache systems," in Journal of microprocessors and microsystems, pp. 561-570, March 1999.

[6] G. Memik, M.T. Kandemir and O. Ozturk, "Increasing register file immunity to transient errors," in Design, Automation and Test in Europe, pp. 586-591, 2005.

[7] Jongeun Lee and AviralShrivastava, "A Compiler Optimization to Reduce Soft Errors in Register Files," in LCTES 2009.

[8] Jason A. Blome, Shantanu Gupta, ShuguangFeng, and Scott Mahlke, "Cost-efficient soft error protection for embedded microprocessors," in CASES '06, pp. 421–431, 2006.

[9] P. Montesinos, W. Liu, and J. Torrellas, "Using register lifetime predictions to protect register files against soft errors," in Dependable Systems and Networks, pp. 286–296, 2007.

[10] M. Rebaudengo, M. S. Reorda, and M.Violante, "An Accurate Analysis of the Effects of Soft Errors in the Instruction and Data Caches of a Pipelined Microprocessor," in DATE'03, pp. 602-607, 2003.

[11] I. Koren and C. M. Krishna, Fault-Tolerant Systems. San Mateo, CA: Morgan Kaufmann, 2007.

[12] MiBench(http://www.eecs.umich.edu/mibench/).

[13] T. Slegel et al, "IBM's S/390 G5 microprocessor design," in IEEE Micro, 19, pp. 12-23, 1999.

[14] M. Fazeli, A. Namazi, and S.G. Miremadi "An energy efficient circuit level technique to protect register file from MBUs and SETs in embedded processors," in Dependable Systems & Networks 2009, pp. 195–204, DNS'09.

[15] K. Walther, C. Galke and H.T. VIERHAUS, "On-Line Techniques for Error Detection and Correction in Processor Registers with Cross-Parity Check," in Journal of Electronic Testing: Theory and Applications 19, pp.501-510, 2003.

[16] M. Spica and T.M. Mak, "Do we need anything more than single bit error correction (ECC)?," in Memory Technology, Design and Testing, Records of the International Workshop on 9-10, pp. 111– 116, 2004.

[17] M. Kandala, W. Zhang, and L. Yang, "An area-efficient approach to improving register file reliability against transient errors," in Advanced Information Networking and Applications Workshops, AINAW '07, pp. 798–803, 2007.

[18] http://archc.sourceforge.net/.

[19] Jun Yan and Wei Zhang, "Compiler-guided register reliability improvement against soft errors," in EMSOFT '05, pp. 203–209, 2005.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

[20] E. Touloupis, J.A. Flint, V.A. Chouliaras and D.D. Ward, "Efficient protection of the pipeline core for safety-critical processor-based systems," in IEEE workshop on Signal Processing Systems Design and Implementation, pp. 188-192, 2005.

[21] Jongeun Lee and A. Shrivastava, "A Compiler-Microarchitecture Hybrid Approach to Soft Error Reduction for Register Files," in Computer-Aided Design of Integrated Circuits and Systems, pp. 1018-1027, 2010.

[22] Jongeun Lee and A. Shrivastava, "Compiler-managed register file protection for energy-efficient soft error reduction," in ASP-DAC, pp. 618–623, 2009.

[23] RiazNaseer, RashedZafarBhatti, and Jeff Draper, "Analysis of Soft Error Mitigation Techniques for Register Files in IBM Cu-08 90nm Technology," in MWSCAS'06, pp. 515-519, 2006.