Error Correction Codes and Parseval Check For Robust Parallel FFTs

[1]Aruna A, [2]Dr. V Sumathy

[1]PG Scholar, [2]Professor
Department of ECE
[1]Government College of Technology, [2]Government College of Engineering
[1]Coimbatore, India
[2]Bodi, India

***Abstract*— Communication and signal processing systems experience the soft error which makes threat to modern electronics circuits. This makes protection against soft errors requirement for many communication and signal processing application. One of the basic building blocks used in many communication applications are FFT. To detect and correct the soft errors occurring in FFT blocks due to environmental changes, Error correction codes (ECC), parity sum of square (SOS), and parity - error correction codes (SOS-ECC) methods are adopted. Redundant module performs error correction using hamming codes. Parity SOS compares the magnitude of input square and output square and produces the result 0 as indication of no error. If there any error present, it produce the result as 1. Parity SOS-ECC combines the operation of ECC and parity SOS. This detects and correct the soft errors present in the parallel FFT'S blocks. The simulation was carried out using Xilinx ISE. The combined parity SOS-ECC produces the best results in terms of implementation complexity and recovers the errors that are out of the tolerance range. The fault coverage for the parity SOS scheme and the parity SOS-ECC scheme is 99.9 % when the tolerance level for SOS checks.**

Keywords—*Error correction code (ECC), Fast Fourier transform (FFTs), Sum of Square SOS), Soft error.*

I. Introduction

The complexity of digital signal processing and communication circuits increases every year. Communication circuits in different signals are often processed simultaneously by parallel copies of the same filters operating [5]. In modern signal processing circuits, it is common to find several filters are operating in parallel. This is the case, for example, the filter bank to apply the input signal is processed simultaneously in different type of filters. Signal processing applications may include communication system, image processing, video processing, data storage and audio processing [4]. This complexity makes the circuits more vulnerable errors. As the scale for designing system that are tolerant to soft errors is becoming circuits increased due to reductions in circuit voltage level and feature size of the circuit. At the same time, the scaling means that transistors operate with lower voltages are more susceptible to errors caused by noise and manufacturing variations. Soft errors are transient error in circuit nodes that can affected in combinational circuits and sequential circuits. Many techniques has been used to product the circuits against soft errors. Including specialized manufacturing processes, system level redundancy and circuit level design techniques [2].

The soft error rate (SER) produced by these effects in may exceed the failure in time (FIT) specifications in various application domains. In many applications for soft error mitigation Schemes should be employed for memories and eventually for logic. Soft error mitigation techniques include approaches using process modifications for instance, the removal of BPSG for eliminating soft errors produced by thermal neutrons [1], [2], approaches adding extra process steps for instance, the addition of DRAM-type capacitance to the Nodes of memory or latch cells, and design-based approaches. These solutions include electrical-level, gate-level and architectural-level approaches. The importance of radiation-induced soft errors also increases as technology scales. Soft errors can change the logical value of a node in a circuit creating a temporary error that can affect the system operation. To ensure that soft errors do not affect the operation of a given circuit, various techniques can be used. These include the use of special manufacturing processes for the integrated circuits like, for example, the silicon on insulator. Alternative option is to design basic circuit blocks or complete design libraries in order to minimize the probability of soft errors. Finally, it is also possible to add redundancy at the system level to detect and correct errors. In this report soft error correct in fast Fourier transform in going to be analysis which is established in any communication system and signal processing system.

Soft errors in semiconductor memories have been a known concern for many years. Error-correcting codes (ECCs) have been employed to combat soft errors in memories. Soft errors occurs in logic and data-path circuits were not a problem in the past because particle hit-induced voltage glitches were masked [14] through the following three mechanisms:

1. Attenuation of noise pulses occurs as a result of propagating through a logic chain (electrical masking);

2. Overriding of the noisy input by logical values (logical masking);
3. Noise pulse missing the latch setup and hold timing window (latching-window masking).

The masking mechanisms are become less effective with, increase in clock frequency, reduction in pipelining depth of modern architectures and reduction in feature size. Soft-error tolerance is an important problem that needs to be addressed at the circuit, architectural, and algorithmic levels. In all cases, enhancing the robustness of systems and circuits to soft errors introduces redundancy. Doing so results in an area and power overhead. Therefore, there is a fundamental tradeoff between energy efficiency and robustness [8]. In this method for extend our past work on algorithmic noise tolerance (ANT) [23]–[26] for combating DSM noise. An ANT based system is composed of an error-control (**EC**) block that detects and corrects errors in the main (**M**) block. the fact that **EC** blocks are much smaller than the main block and, hence, can be designed to be robust to DSM noise source as voltage/frequency scaling, process variation, ground bounce, etc. The primary contributions of this paper are twofold.

Redundant Residue Number System (RRNS) representation method in which some additional module are used to detect and correct errors in one of the element of RNS processor. The techniques proposed to correct errors in the RRNS representation do not guarantees that a fault inside the reverse conversion block or inside the error detection and correction block will be detected and corrected. In this paper, to mask the effect of a fault inside these blocks a method based on a particular voter is introduced. This voter performs both the correction of an error inside a module of the RRNS and of an error inside the reverse converter. This method of voter allows the implementation of a totally fault tolerant RNS based in FIR filter saving more than 33% of resource with then TMR implementation of the blocks performing RNS based error correction. Triple modular redundancy (TMR), three copies of the basic circuit is operating in parallel given to the same inputs. The TMR, which triplicates the design and adds voting logic to correct errors, is commonly used. However, it more than triples the area and power of the circuit. Dual modular redundancy (DMR) only uses one redundant module and so, conventionally, DMR can detect errors but cannot correct them. Alternative method in the algorithmic or structural properties of certain circuits, such as signal processing circuits can be used to detect and correct errors.

Environmental interference and physical defects in the communication system can causes the random bit errors during data transmission. Error coding method of detecting and correcting the errors to ensure information is transferred intact from source to destination. Error coding is used for fault tolerant computing in computer memory, satellite and space communications, magnetic and optical data storage media, cellular telephone networks, network communications, and almost any other form of digital data communication.

Signal processing and communications circuits are well Suite for ABFT. They have regular structures and many algorithmic properties [4]. Many ABFT techniques have been proposed to protect the basic blocks are commonly used in the circuits. Several works have been considered the protection of digital filters [5], [6]. For example, the use of replication using reduced precision copies of the filter has been proposed as alternative to TMR with an lower cost [7]. The knowledge of the distribution in the filter output has been recently exploited to detect and correct errors with lower overheads [8]. The protection of fast Fourier transforms (FFTs) has been widely studied [9], [10].

It is common to determine the several filters or FFTs operating in parallel as signal-processing circuits are more complex. This is a mainly in filter banks in multiple-input multiple-output (MIMO) communication systems [2]. MIMO orthogonal frequency division modulation (MIMO-OFDM) systems use parallel IFFTs and FFTs for modulation and demodulation. On long-term evolution mobile systems and on WiMax [8] MIMO-OFDM is implemented [12]. It is a possible to advantage of the parallel filters or FFTS to implement using ABFT techniques [6]. It can be implement ABFT techniques for the entire group of parallel modules instead of each one independently. Initially this is studied for digital filter.

II. PARALLEL FFT PROTECTION USING ECCS

FFT algorithm computes the DFT of a sequence or its inverse. Fourier analysis converts a signal from its original domain to a representation in the frequency domain. It has high operating speed and less computation time. Fast Fourier transforms are widely used for many and mathematics. Hardware cost less than other architecture level.

The starting point for our work is the protection based on the use of ECCs that was presented in [17] digital filters. This scheme is shown in Fig.1. In this example, a simple single error correction using Hamming code [18]. The original system consists of four FFT modules and three redundant modules is added to detect and correct errors. The inputs to three redundant modules are linear combinations of the inputs. They are used to check the linear combinations of the outputs. For example, the input to the first redundant module is

$$X_5 = X_1 + X_2 + X_3 \quad \text{... (1)}$$

And since the DFT is a linear operation, its output of $z5$ can be used to check below equation is

$$Z_5 = Z_1 + Z_2 + Z_3 \quad \text{… (2)}$$

This will be denoted by $c1$ check error. The same method applies to the other two redundant modules that will be provide checks $c2$ and $c3$. Based on the differences observed on each of the checks, the module on which the error has been occur can be determined. The different pattern and the corresponding errors are summarized in hamming codes Table.1. It is an one of the error correction code. Hamming codes are single bit error correction code. Hamming codes widely used in computing, telecommunication and data compression. This technique will work for any single error correcting code.

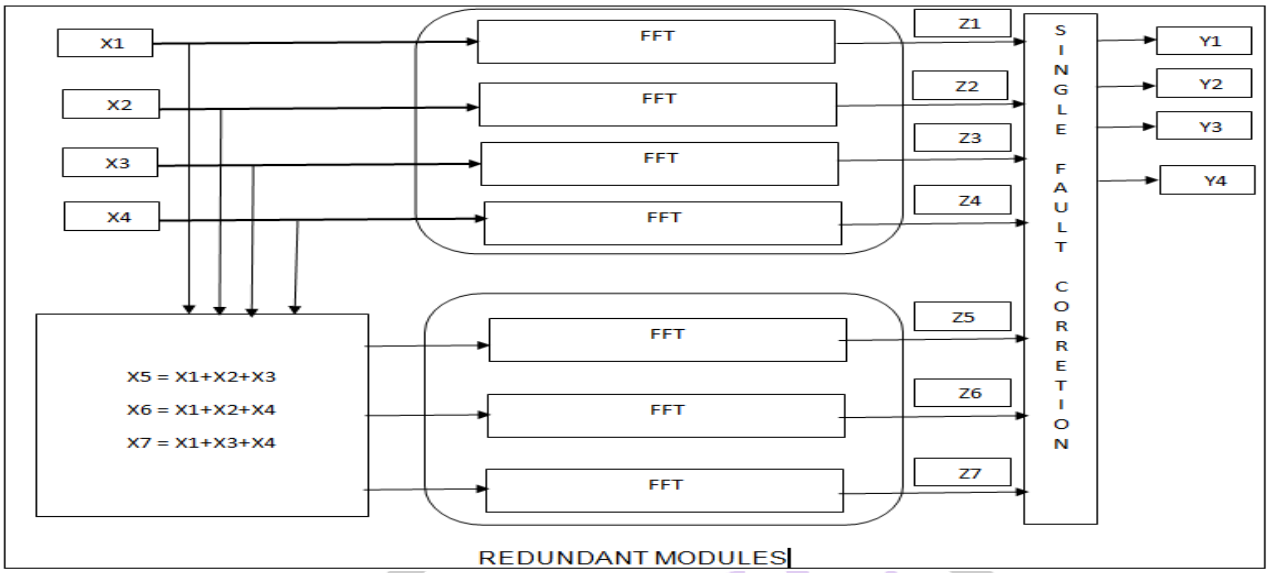

Fig .1 parallel FFT production using ECCs

Once the module an error is known, the error can be corrected by reconstructing the output using remaining modules. For example, error affected $z1$, this can be done as follows:

$$Z_1c[n] = Z_5[n] - Z_2[n] - Z_3[n] \quad \text{... (3)}$$

Similar correction equations can be used to correct the errors on other modules. More advanced ECCs can be used to correct errors of multiple modules if that is need to give application. The overhead of this technique, discussed in [17], is lower than TMR as the number of redundant FFTs is related to the logarithm of original FFTs. For example of protect four FFTs, three redundant FFTs are needed, but to protect eleven FFTs using the number of redundant FFTs only for four. This shows how the overhead decreases with the number of FFTs.

Table I. Error Location in the Hamming Codes

BINARY BITS			ERROR BIT POSITION
C1	C2	C3	
0	0	0	No error
1	1	1	Z1
1	1	0	Z2
1	0	1	Z3
0	1	1	Z4

III . Proposed Method

A.Parity-Sos Fault-Tolerant Parallel FFTs

It has been many techniques have proposed to protect the FFTs. One of them is the Sum of Squares (SOSs) check [4] that can be used to detect errors. A simple ECC technique takes a block of *k* bits and produces a block of *n* bits by adding *n−k* parity check bits. XOR combinations of the *k* data bits are used as the parity check bits. It is possible to detect and correct errors by properly designing those combinations. The SOS check is based on the Parseval theorem states that the SOSs of the inputs to the FFTs are equal to the SOSs of the outputs to the FFTs are except in a scaling factor. This relationship can be detect errors with low overhead as one multiplication is needed for each input and output samples are used (two multiplications and adders for SOS per sample).

For parallel FFTs are the SOS check can be combined with the ECC approach to reduce the protection overhead. Since the SOS check can only detect errors, and the ECC part should be able to implement correct the error. This can be done using the equivalent of a simple parity bit for all the FFTs. In addition, of the SOS check can be used to each FFT to detect errors. When a error is detected and the output of the parity FFT can be used to correct the error. This is better explained with an example in Fig.2. In this method for illustrated the case of four parallel FFTs. The redundant (the parity) FFT is added that the sum of the inputs to the original FFTs as input. An SOS check is added to each original FFT. In case of error is detected (using *P*1, *P*2, *P*3, *P*4*)*, the correction can be done by recomputing the FFT in error occur the output of the parity FFT (*X*) and the rest of the FFT outputs. For example, if an error occurs in the first FFT, to *P*1 will be set and the error is corrected by using below equation

$$X_1c = X - X_2 - X_3 - X_4 \quad \text{... (4)}$$

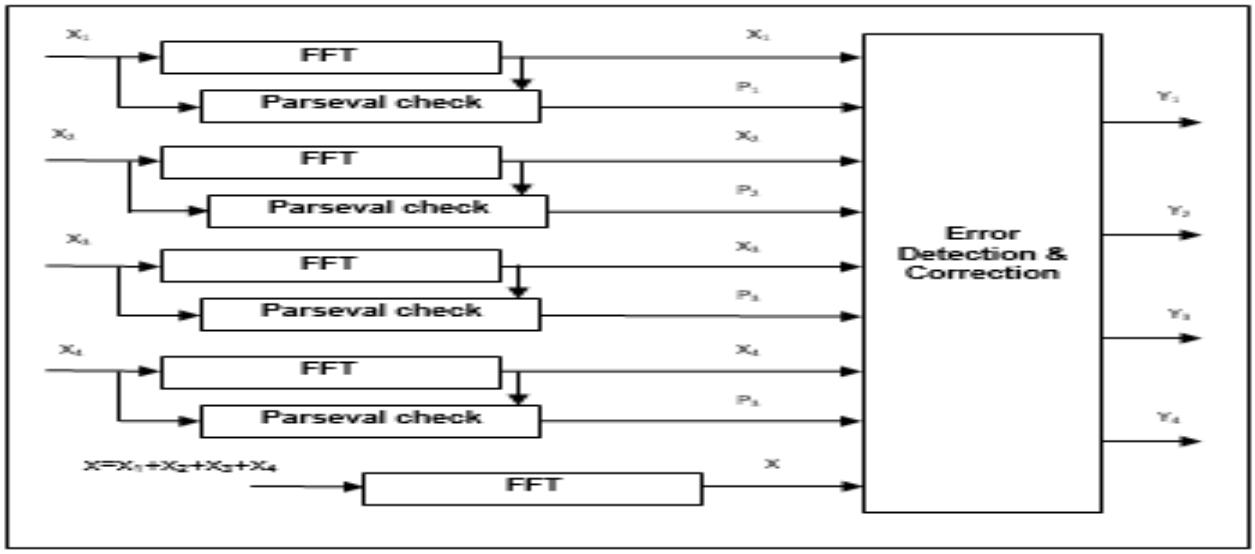

Fig.2 Parity-SOS fault-tolerant parallel FFTs

B.Parity-Sos-Ecc Fault-Tolerant Parallel FFTs

This combination of a parity FFT and the SOS check reduces the number of FFTs to just one and may therefore, reduce the protection overhead. In the following, this method will be referred to as parity-SOS. Another possibility to combine the SOS check and the ECC approach is instead of using an SOS check FFT, use an ECC for the SOS checks. Then the parity-SOS method is an additionally using parity FFT is correct the errors. This second technique is shown in Fig. 3. The main benefit over the first parity- SOS method is to reduce the number FFTs and SOS checks needed. The error location process is the same as for the ECC method in Fig.1 and correction in the parity-SOS method. In the following, this method will be referred to as parity-SOS-ECC (or second proposed technique). The overheads of the two proposed methods can be initially estimated using the number of additional FFTs and SOS check blocks are needed. It can be observed that the two proposed methods are reduce the number of additional FFTs to just one. In addition, the second technique also reduces the number of SOS checks. A detailed evaluation for an FPGA implementation is discussed to illustrate the relative overheads of the proposed techniques. In all the techniques discussed, soft errors can also affect the elements are added for protection. For the ECC technique, the protection of the elements was discussed in [17].

In the case of the redundant or parity FFTs, an error will have no effect as it will not propagate to the data outputs and will not trigger a correction. In the case of SOS checks, an error will trigger a correction when actually there is no error on the FFT. This will cause an unnecessary correction but will also produce the correct result. Finally, errors on the detection and

correction blocks in Fig.2 and fig.3 can propagate errors to the outputs. In our implementations, those blocks are protected with TMR. The same applies for the adders used to compute the inputs to the redundant FFTs in Fig.1 or to the SOS checks in Fig.3. The triplication of these blocks has a small impact on circuit complexity as they are much simpler than the FFT computations. A final observation is that the ECC scheme can detect all errors that exceed a given threshold (given by the quantization used to implement the FFTs) [17]. On the other hand, the SOS check detects most errors but does not guarantee the detection of all errors [4]. Therefore, to compare the three techniques for a given implementation, fault injection experiments should be done to determine the percentage of errors that are actually corrected. This means that an evaluation has to be done both in terms of overhead and error coverage.

In the case of parity-SOS and parity-SOS-ECC, SOS checks are used to detect and locate the errors and a simple parity FFT is used for correction. For the detection and correction of the errors, we can use SOS check per FFT or alternatively using a set of SOS checks. Another technique used for triple modular redundancy to the circuit. Both methods are algorithm based fault tolerant technique [1]. TMR has the ability to triplicates the design voting logic to correct errors. It is commonly used. TMR is an very effective at protecting FPGA circuits from the soft errors. But it is costly in terms of the circuit area, power, and circuit timing.

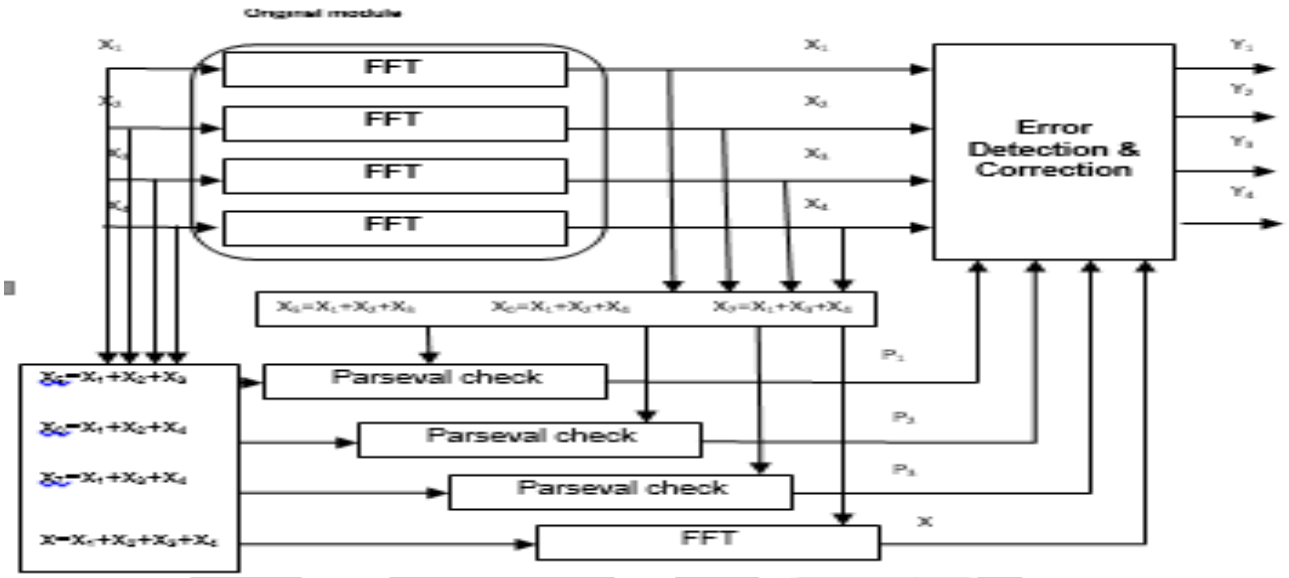

Fig.3 Parity SOS ECC fault tolerant parallel FFTs

FFT's protection method is studied and different techniques are implemented [15]. The latest technique are parity-SOS and parity-SOS-ECC used to reduce soft errors in FFTs. These techniques are formed by combining an existing method is known as ECC approach with the traditional SOS check. To detect and locate the errors used in SOS checks and a simple parity FFT is used for correction. Error detection can be done using an SOS check FFT or alternatively using a set of SOS checks form an ECC. It protects against large magnitude errors in arithmetic circuits. Its area savings make it an alternative for protecting FPGA signal processing circuits, transient and soft data errors. All these are single error detection and correction methods are only used.

iv. Results And Discussions

Parity-Sos Fault-Tolerant Parallel FFTs

Parallel FFT protection using ECC with fault was simulated using Xilinx software and gives the results as shown in below. An error will trigger a correction when actually there is no error on the FFT. This will cause an unnecessary correction, but will produce the correct result. If there is no error occurred the hamming codes return the value "000". If the error is occurred the following equation are used to detect the location of the error bit position using table 1.

The inputs of the three redundant modules are linear combinations of the inputs and they are used to check linear combinations' of the outputs. The given inputs are given below

$$X_5 = X_1 + X_2 + X_3$$
$$X6 = X_1 + X_2 + X_4$$
$$X_7 = X_1 + X_3 + X_4$$

Since the DFT is a linear operation, its output of the following equations are used to detect the error bit location.

$$Z_5 = Z_1+Z_2+Z_3$$
$$Z_6 = Z_1+Z_2+Z_4$$
$$Z7 = Z_1+Z_3+Z_4$$

The 4-point FFT operation which works for faulty condition is given below. Here the error is injected in the input. For example,

Input	→	Error Inject Input
X(0) = 0001		x(0) = 0000
X(1) = 0010		x(1) = 0010
X(2) = 0011		x(2) = 0011
X(3) = 0100		x(3) = 0100

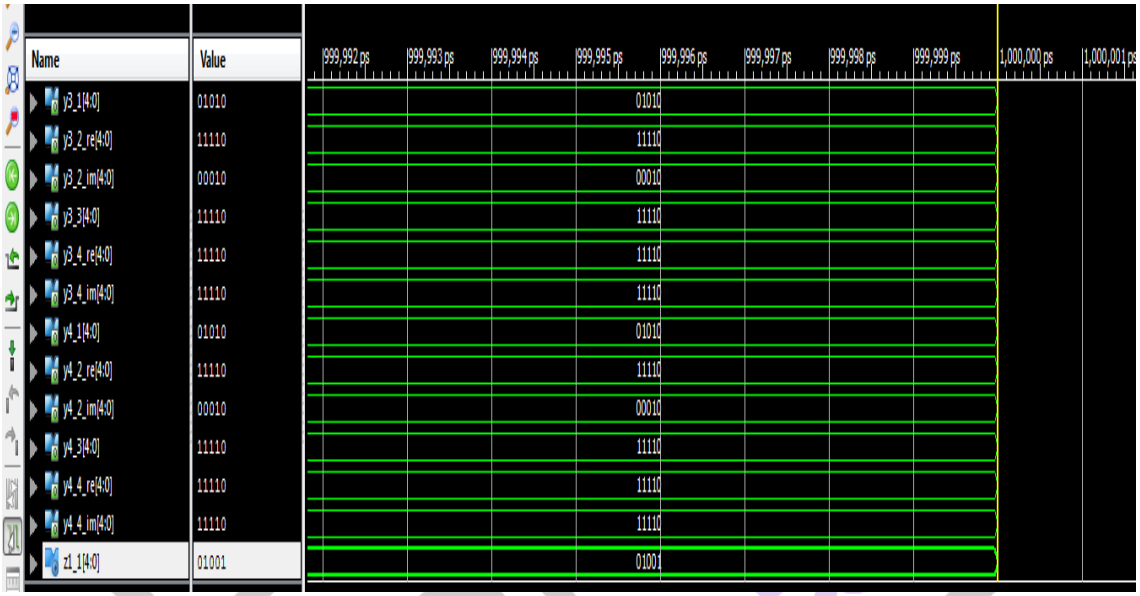

Fig.4.a. simulation output

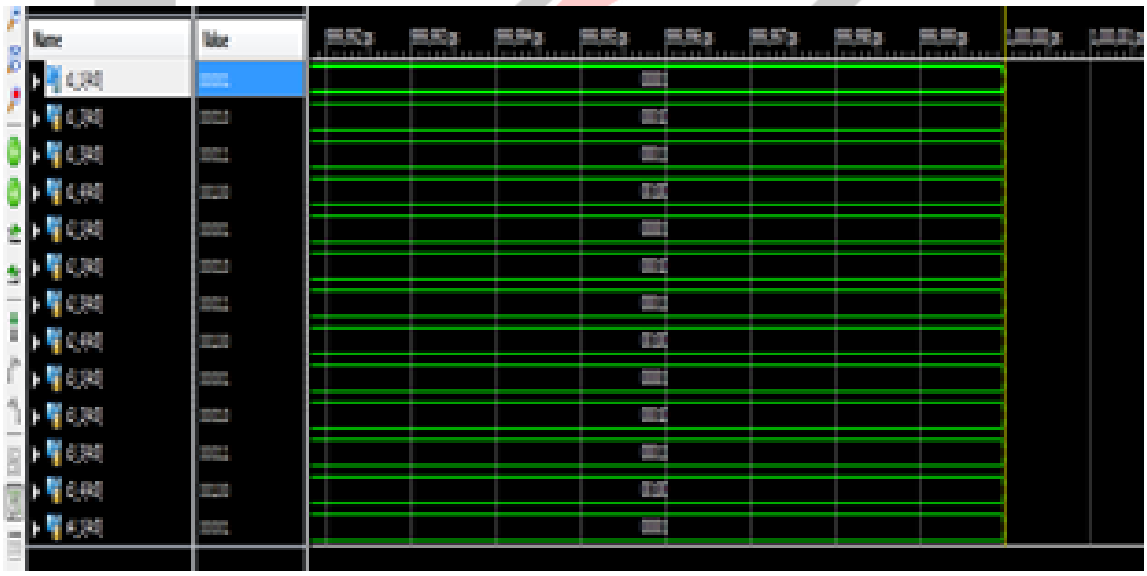

Fig.4.b. simulation output

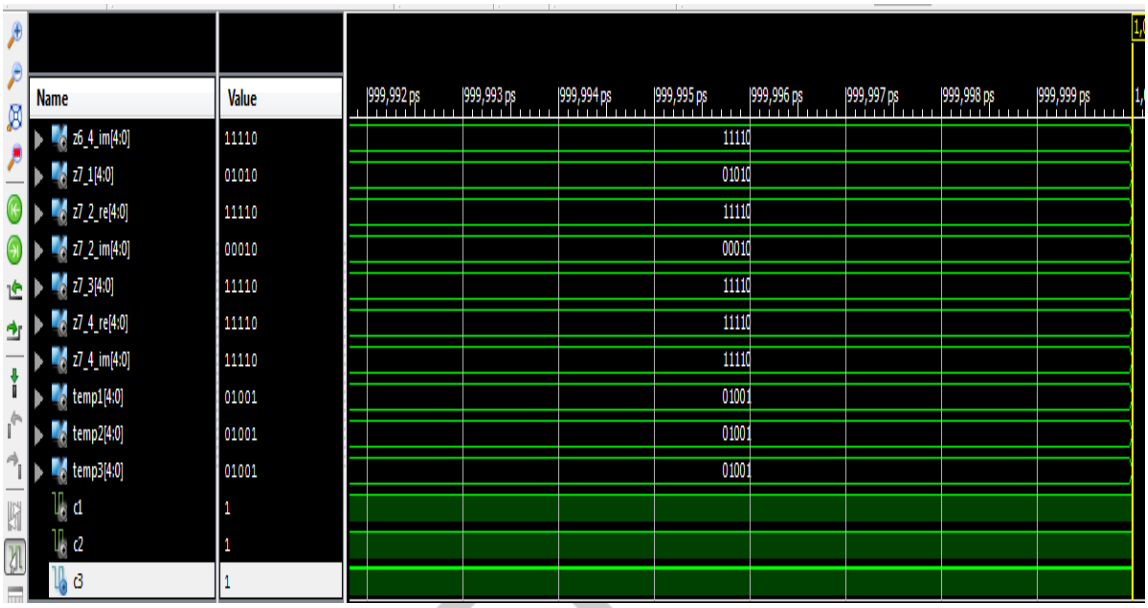

Fig.4.c. simulation output

The outputs from the FFT block is as follows,

Z(0) = 1011
Z(1) = 1101
Z(2) = 1111
Z(4) = 1101

The simulation operation will be performed on other three FFT blocks. Once the error is detected in any one of the FFT blocks, this will be done by using hamming codes (table .1).Then the error will be corrected by ECC block.

V. EVALUATION

The two proposed schemes and the ECC scheme presented in [17] have been implemented on FPGA and evaluate both in terms of overhead and error coverage. A four-point decimation-in-frequency FFT core is used to compute the FFT iteratively. This core has been developed to implement MIMO-OFDM for wireless systems. The implementation of the four-point FFT. The number of FFT points is programmable and the rotation coefficients are calculate for each stage and store in registers. For the evaluation in a 1024 points FFT is configured with five stages of calculation is (log41024 = 5), so in total for 5∗1024 = 5120 cycles are need to calculate the FFT for 1024 input samples. The inputs are 12-bits wide and the outputs are 14-bits wide. For the redundant FFT, the bit widths are extended to 14 and 16 bit, respectively, to cover the larger dynamic range (as the inputs are the sum of several signals). Since both the inputs and outputs to the FFT are sequential, the SOS check is also done sequentially using accumulators that are compared at the end of the block.

The FFT and the different protection techniques have been implemented using VHDL Then, the design has been mapped to a Virtex-4 in FPGA setting the maximum effort on minimize the use of resources. The results obtained are summarized in Tables II and III. The first table provides the resources needed to implement a single FFT and an SOS check. The results show that the FFT is more than complex of the SOS check. The difference will be much larger when a fully parallel FFT implementation is used. Tables II–III show in the results when different number of parallel FFTs are protected.

The objective is to illustrate how the relative overheads of the different techniques vary with the number of parallel FFTs. In parentheses, the cost relative to an unprotected implementation is also provided. The results show that all techniques have a cost factor of <2.This demonstrates that the ECC-based technique proposed in [17] is also competitive to protect FFTs and requires a much lower cost than TMR. The parity-SOS-ECC technique has the lowest resource use in all cases and, therefore, is the best option to minimize the implementation cost.

TABLE II Resources Usage for a Single FFTs and SOS Check

	FFT	SOS Check
Slices	1377	494
Flip-flops	1037	141
LUT – 4	2530	979

TABLE III Resources Usage for Four Parallel FFTS

	Unprotected FFTs	ECC protected	Parity SOS protected	Parity SOS ECC protected
Slices	5468	9890	9009	8552
Flip-flops	4148	7780	6047	5988
LUT – 4	10120	18188	16968	16092

TABLE IV Resources Usage for Six Parallel FFTS

	Unprotected FFTs	ECC protected	Parity SOS protected	Parity SOS ECC protected
Slices	8238	14412	13024	12171
Flip-flops	6684	11294	8593	8584
LUT – 4	15198	26571	24539	23036

V. CONCLUSION

Two techniques have been proposed for the production against FFTs implementation soft errors. The proposed techniques are based on combining an existing ECC approach with the additional SOS checks. The SOS checks are used to detect and locate the errors. A simple parity FFT is used for correction. The detection and location of the errors can be done using an SOS checks per FFT (or) alternatively using a set of SOS checks that form an ECC. The technique have been evaluated in terms of complexity and error detection capabilities. The result show that the method which uses a parity FFT set of SOS checks that form an ECC provides the best result in terms of implementation complexity.

In term of error protection, fault injection method show that the ECC Scheme can recover all the error that are out of the tolerance range. In future the error detection and error correction will be implemented using Reed-Solomon (RS) codes which are best suited for error detection and error correction, which provides high complexity and efficiency. In this paper a single bit error correction for FFT was discussed. For future studies, the multi-bit error correction and detection was introduced by replacing two full precision FFTs by two partial precision FFTs which limits error, area and power consumption. The multi-bit error correction are considered as a single bit error for this approach.

REFERENCES

[1] N. Kanekawa, E. H. Ibe, T. Suga, and Y. Uematsu, *Dependability in Electronic Systems: Mitigation of Hardware Failures, Soft Errors, and Electro-Magnetic Disturbances*. New York, NY, USA:Springer- Verlag,2010.

[2] R. Baumann, "Soft errors in advanced computer systems," *IEEE Des. Test Comput.*, vol. 22, no. 3, pp. 258–266, May/Jun. 2005.

[3] M. Nicolaidis, "Design for soft error mitigation," *IEEE Trans. Device Mater. Rel.*, vol. 5, no. 3, pp. 405–418, Sep. 2005.

[4] A. L. N. Reddy and P. Banerjee, "Algorithm-based fault detection for signal processing applications," *IEEE Trans. Comput.*, vol. 39, no. 10, pp. 1304–1308, Oct. 1990.

[5] T. Hitana and A. K. Deb, "Bridging concurrent and non-concurrent error detection in FIR filters," in *Proc. Norchip Conf.*, Nov. 2004, pp. 75–78.

[6] S. Pontarelli, G. C. Cardarilli, M. Re, and A. Salsano, "Totally fault tolerant RNS based FIR filters," in *Proc. 14th IEEE Int. On-Line Test Symp. (IOLTS)*, Jul. 2008,pp. 192–194.

[7] B. Shim and N. R. Shanbhag, "Energy-efficient soft error-tolerant signal processing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* vol. 14, no. 4, pp. 336–348, Apr. 2006.

[8] E. P. Kim and N. R. Shanbhag, "Soft N-modular redundancy," *IEEE Trans. Comput.*, vol. 61, no. 3, pp. 323–336, Mar. 2012.

[9] J. Y. Jou and J. A. Abraham, "Fault-tolerant FFT networks," *IEEE Trans. Comput.*, vol. 37, no. 5, pp. 548–561, May 1988.

[10] S.-J. Wang and N. K. Jha, "Algorithm-based fault tolerance for FFT networks," *IEEE Trans. Comput.*, vol. 43, no. 7, pp. 849–854, Jul. 1994.
[11] P. P. Vaidyanathanm, *Multirate Systems and Filter Banks*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1993.
[12] A. Sibille, C. Oestges, and A. Zanella, *MIMO: From Theory to Implementation*. San Francisco, CA, USA: Academic, 2010.
[13] G. L. Stüber, J. R. Barry, S. W. McLaughlin, Y. Li, M. A. Ingram, and T. G. Pratt, "Broadband MIMO-OFDM wireless communications," *Proc. IEEE*, vol. 92, no. 2, pp. 271–294, Feb. 2004.
[14] S. Sesia, I. Toufik, and M. Baker, *LTE—The UMTS Long Term Evolution: From Theory to Practice*, 2nd ed. New York, NY, USA: Wiley, Jul. 2011.
[15] M. Ergen, *Mobile Broadband—Including WiMAX and LTE*. New York, NY, USA: Springer-Verlag, 2009.
[16] P. Reviriego, S. Pontarelli, C. J. Bleakley, and J. A. Maestro, "Area efficient concurrent error detection and correction for parallel filters," *IET Electron. Lett.*, vol. 48, no. 20, pp. 1258–1260, Sep. 2012.