# Parallel Fast Fourier Transform with fault Tolerant realization on System on Chip

RAJU KATRU[1], DR.M.CHANDRASEKHER[2]
[1]Ph.D Student in Rayalaseema University, Kurnool, Andhrapradesh, India
[2] Manager(D&E),Bharath Dynamite Ltd, Khanchnbagh ,Ministry of Defense, Hyderabad, India

*Abstract*— Soft faults pretence a constancy threat to recent electronic circuits. This makes defense in opposition to soft faults a prerequisite for many applications. For some applications, an interesting option is to use algorithmic-based fault tolerance (ABFT) techniques that try to utilize the algorithmic properties to detect and correct faults. Signal processing and communication applications are well matched for ABFT. One illustration is fast Fourier transforms (FFTs) that are a key building block in many systems. Several defense systems have been planned to detect and correct faults in FFTs. Among those, probably the exploit of the Parseval or sum of squares check is the most broadly known. In modern communication systems, it is more and more common to locate several blocks operating in parallel. in recent times, a technique that exploits this fact to execute fault tolerance on parallel filters has been planned. In this brief, this method is first applied to protect FFTs. Then, two enhanced protection systems that combine the use of fault correction codes and Parseval checks are planned and evaluated. The results show that the planned systems can further decrease the execution cost of protection.

*Key words*— Fault correction codes (ECCs), Fast Fourier Transforms (FFTs), Soft faults.

## I. INTRODUCTION

The complication of communications and signal handing out circuits increases every year. This is made probable by the CMOS technology scaling that enables the incorporation of more and more transistors on a single chip. This increased complication makes the circuits more exposed to faults. At the similar time, the scaling means that transistors operate with lower voltages and are more vulnerable to faults caused by noise and developed variations [1]. The significance of radiation-induced soft faults also increases as technology scales [2]. Soft faults can modify the logical value of a circuit node creating a temporary fault that can affect the system operation. To make certain that soft faults do not have an effect on the operation of a given circuit, a wide variety of *techniques* can be used [3]. These include the use of extraordinary developed processes for the integrated circuits like, for example, the SOC. Another option is to design basic circuit blocks or absolute design libraries to reduce the probability of soft faults. Finally, it is also doable to add redundancy at the system level to sense and correct faults.

Improvement techniques are that they want a large overhead in condition of circuit operation. For example, One traditional replica is the use of triple modular redundancy (TMR) that triples a build block and votes among the three outputs to notice and accurate faults. The major complexity with those soft fault for TMR, the transparency is greater than 200%. This is because the undefended unit is simulated three times (it requires a 200% overhead versus the undefended unit), and moreover, voters are needed to accurate the faults building the overhead greater than 200%. This overhead is unwarranted for several applications. one additional approach is to attempt to apply the algorithmic property of the circuit to differentiate/correct faults. This is frequently referred to as algorithm-based fault tolerance [4]. This come near can reduce the overhead vital to remain a circuit.

Signal dispensation as well as communications circuits are fit suitable for ABFT when they have ordinary structure and a lot of algorithmic properties [4]. Over the duration, several ABFT techniques have been designed to save from harm the basic blocks that are frequently used in those circuits. Several works have consider the protection of digital filter [5], [6]. For example, the use of reproduction using reduced accuracy copies of the filter has been projected as an alternative to TMR but with a lower price [7]. The information of the allocation of the filter output has also been newly exploited to detect and correct faults with lower overheads [8]. The safety of fast Fourier transforms has moreover been broadly studied [9], [10].

As signal-processing circuits become additional complex, it is general to find some filters or FFTs operating in parallel. This occur for example in filter banks [11] or in multiple-input multiple-output (MIMO) communication systems [12]. In MIMO orthogonal frequency division modulation (MIMO-OFDM) systems use parallel iFFTs/FFTs for modulation/demodulation [13]. MIMO-OFDM is designed on durable evolution mobile systems [14] and also on WiMax [15]. The occurrence of parallel filters or FFTs creates an chance to execute ABFT techniques for the complete group of equivalent modules in its place of for each one alone. This has been studied for digital filters at first in [16] where two filters were considered. More newly, a general system based on the use of fault correction codes (ECCs) has been planned [17]. In this method, the system is that each filter can be the equivalent of a bit in an ECC and parity check bits can be computed using addition. This method can be used for operations, in which the output of the addition of a number of inputs is the addition of the individual outputs. This is correct for any linear

function as, for example, the discrete Fourier transforms (DFT).

In this brief, the security of parallel FFTs is studied. In particular, it is assumed that there can single be a only fault on the system at any given point in time. This is a general theory when considering the security against radiation-induced soft faults [3]. There are three major contributions in this brief

1) The assessment of the ECC method [17] for the security of parallel FFTs presentation its efficiency in terms of overhead and safety effectiveness.
2) The proposal of a new technique based on the use of Parseval or sum of squares (SOSs) checks [4] add with a parity FFT.
3) The proposal of a new method on which the ECC is used on the SOS checks in its place of on the FFTs.

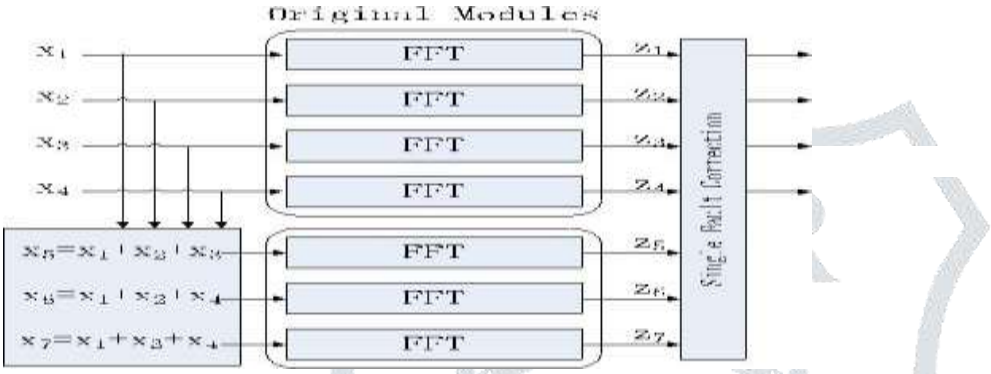



Fig. 1. Parallel FFT Protection using ECCs.

The two proposed techniques provide new alternatives to save from harm parallel FFTs that can be more capable than protecting all of the FFTs separately

The planned systems have been evaluated using FPGA exestuations to assess the safety overhead. The results show that by adding the use of ECCs and Parseval checks, the safety overhead can be reduced compared with the use of only ECCs as projected in [17]. Fault injection experiments have also been conducted to check the ability of the exestuations to detect and correct faults.

## II. PROPOSED PROTECTION SYSTEMS FOR PARALLEL FFTS

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

$$x_5 = x_1 + x_2 + x_3 \qquad (1)$$

and because the DFT is a linear operation, its output $z_5$ can be used to check that

$$z_5 = z_1 + z_2 + z_3. \qquad (2)$$

This will be denoted as $c_1$ check. The same reasoning apply to the other two redundant modules that will provide checks $c_2$ and $c_3$. Based on the differences observed on each of the checks, the module on which the fault has occurred can be determined. The diverse patterns and the corresponding faults are summarized in Table I. Once the module in fault is known, the fault can be corrected by reconstructing its output using the remaining modules. For example, for an fault affecting $z_1$, this can be done as follows:

$$z_{1c}[n] = z_5[n] - z_2[n] - z_3[n]. \qquad (3)$$

Similar correction equations can be used to correct faults on the other modules. More advanced ECCs can be used to correct faults on multiple modules if that is needed in a given application.

The overhead of the given technique, as discussed in [17] , is lower than TMR as the number of redundant FFTs is related to the logarithm of the number of original FFTs. For example, to defend four FFTs, three redundant FFTs are needed, but to protect eleven, the number of redundant FFTs in only four. This shows how the overhead decrease with the number of FFTs

Table I
Fault Location in the Hamming Code

| | |
|---|---|
| 000 | No Fault |
| 111 | Z1 |
| 110 | Z2 |
| 101 | Z3 |
| 011 | Z4 |
| 100 | Z5 |
| 010 | Z6 |
| 001 | Z7 |

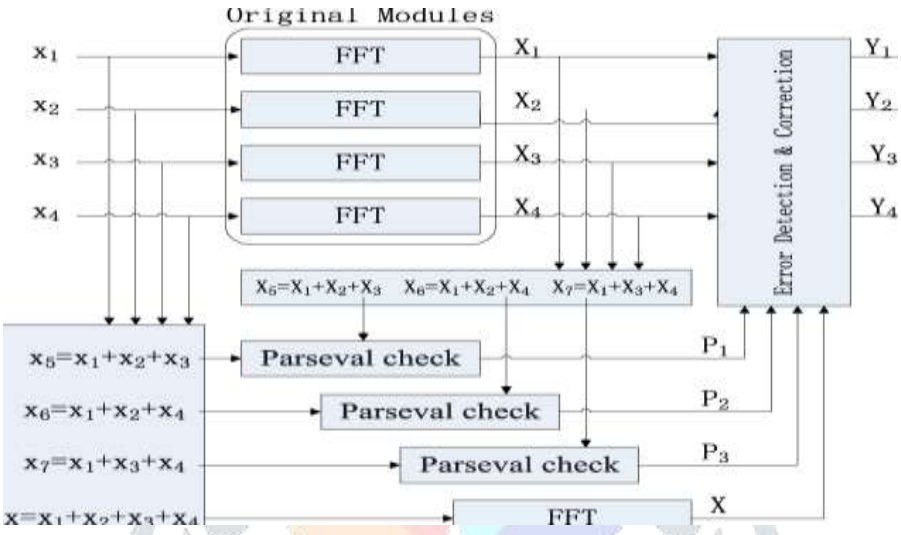



Fig. 2. Parity-SOS (first technique) fault-tolerant parallel FFTs

In Section I, it has been mention that over the years, many techniques have been proposed to protect the FFT. One of them is the Sum of Squares (SOSs) check [4] that can be used to notice faults. The SOS ensures is based on the Parsuval theorem that states that the SOSs of the inputs to the FFT are equal to the SOSs of the outputs of the FFT except for a scaling factor. This relationship can be used to detect faults with low overhead as one multiplication is needed for each input or output sample (the two multiplication and adders for SOS per sample).

For parallel FFTs, the SOS check can be shared with the ECC approach to decrease the shield overhead. Since the SOS check can only detect faults, the ECC part should be able to apply the correction. This can be done using the equaling of a simple parity bit for all the FFTs. In totaling, the SOS verification is used on each FFT to detect faults. When an fault is detected, the output of the parity FFT can be used to exactly the fault. This is better explained with an example. In Fig. 2, the first proposed valid system is to be illustrated for the case of four parallel FFTs. A out of work (the parity) FFT is added that has the sum of the inputs to the original FFTs as input. An SOS check is also added for each original FFT. In case an fault is detected (using $P_1$, $P_2$, $P_3$, $P_4$), the correction can be done by re compute the FFT in fault using the output of the parity FFT($X$) and rest of the FFT outputs. For example, if an fault occurs in the first FFT, $P_1$ will be set and the fault could be corrected by doing

$$X_{1c} = X - X_2 - X_3 - X_4. \qquad (4)$$

This combination of a parity FFT and the SOS check reduced the number of extra FFTs to just one and may, therefore, reduce the shield overhead. In the following, this system will be referred to as parity-SOS (or first proposed method).

Another possibility to be combine the SOS check and the ECC approach is done instead of using an SOS check per FFT, use an ECC for the SOS checks. Then as in the parity-SOS method, an extra parity FFT is used to correct the faults. This second method is shown in Fig. 3. The main benefit over the first parity-SOS system is to reduce the number of SOS check required. The fault location process is the same as for the ECC system in Fig. 1 and alteration is as in the parity-SOS method. In the next, this method will be referred to as parity-SOS-ECC (or second planned technique).

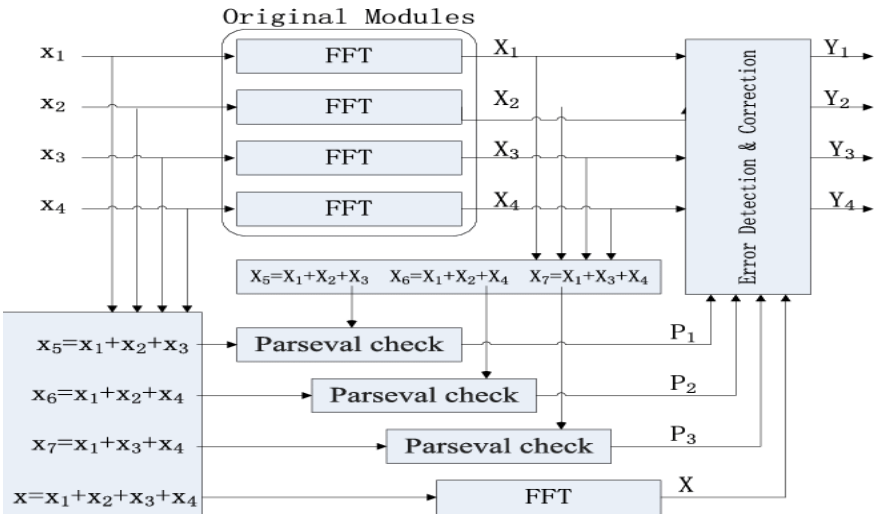



Fig. 3. Parity-SOS-ECC (second technique) fault-tolerant parallel FFTs.

TABLE II
OVERHEAD OF THE DIVERSE SYSTEMS TO PROTECT *k* FFTs

| | | SOS Checks |
|---|---|---|
| ECC | 1-Log(k) | 0 |
| Parity- SOS | 1 | k |
| Parity- SOS-ECC | 1 | 1-Log(k) |

The spending of the two planned systems can be initially estimate is done by using the number of extra FFTs and SOS check blocks needed. This information is shortened in Table II for a set of *k* original FFT modules assuming *k* is a power of two. It can be observed that the two proposed system reduce the number of additional FFTs to be just one. In addition to the second technique also reduces the number of SOS checks. In the given below Section III, a full evaluation for an FPGA functioning is to discuss to illustrate the relative overheads of the planned techniques.

In all the techniques discussed, soft fault can also affect the elements added for protection. For the ECC method, the protection of these elements was discussed in [17]. In the case of the out of the work or parity FFTs, an fault will have no effect as it will not broadcast to the data outputs and will not mainly trigger a alteration. In the case of SOS checks, an fault will triggers a alteration when actually there is no fault on the FFT. This will cause an unnecessary corrections but will also produces the corrects the result. Finally, faults on the discovery and alteration blocks in Figs. 2 and 3 can propagate faults to the outputs. In our executions, those blocks are protected with TMR. The same applies for the adders used to calculate the inputs to the redundant FFTs in Fig. 1 or to the SOS checks in the Fig. 3. The triplication of these blocks has some small impact on circuit complexity as they are much simpler than the FFT computations.

A final watching is that the ECC method can detect all faults that exceed a given threshold (given by the quantization used to executes the FFTs) [17]. On the other hand, the SOS check detect most faults but does not guarantee the detection of all faults [4]. thus, to evaluate the three technique for a given functioning, fault injection experiment should be done to determine the proportion of faults that are actually corrected. This means that an assessment has to be done both in terms of overhead and fault coverage.

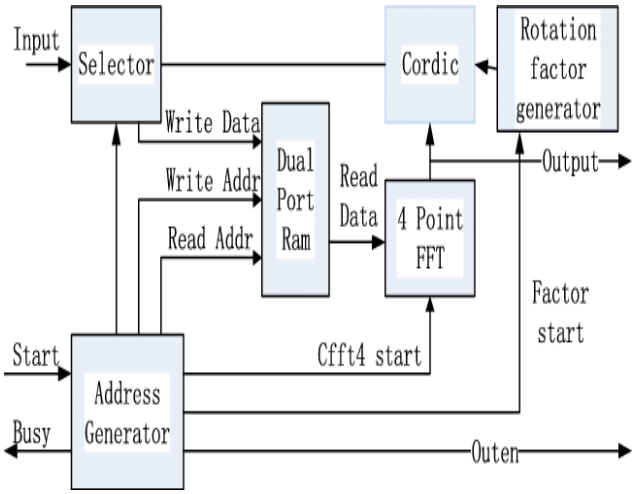



Fig. 4 . Architecture of FFT Execution

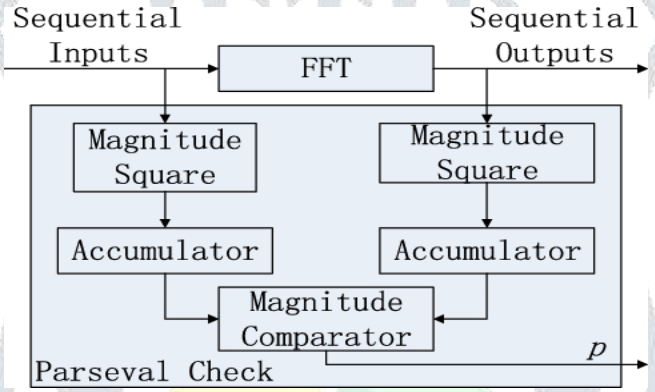



Fig. 5. Execution of the SOS check

## III. EVALUATION

The two proposed system and the ECC system presented in [17] have been executed on FPGA and evaluated both in terms of overhead and fault coverage. A four-point decimation-in-frequency FFT core is used to compute the FFT iteratively. This core has been developed to execute MIMO-OFDM for wireless systems. The execution of the four-point FFT core is shown in Fig. 4. The number of FFT points is programmable and the rotation coefficients are calculated on-line for each stage and stored in registers. For the evaluation of 1024 points FFT is configured with five stages calculation ($\log_4 1024 = 5$), so in total $5^* 1024 = 5120$ cycles are needed to calculate the FFT for 1024 input samples. The inputs are 12 - bit wide and the outputs are 14-bit wide. For the redundant FFT, the bit widths are complete to 14 and 16 bit, respectively, to cover the larger vigorous 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 in succession using accumulators that are to be compared at the end of the block. This is shown in Fig. 5. To lessen the impact of round offs on the fault coverage, the outputs of the collector are 39-bit wide. For the estimate, several values of the number of parallel FFTs are considered. This is done to evaluate the diverse techniques as a function of the number of parallel FFTs in the original system.

The fault detection and alteration blocks (Figs. 1–3) are executed as multiplexers that select the accurate output depending on the fault pattern detected. As mentioned before, these blocks are tripled to ensure that faults that affect them do not corrupt the final outputs.

The FFT and the diverse protection techniques have been executed using Verilog. Then, the design has been mapped to a Virtex-4 xc4vlx80 FPGA setting the maximum effort on minimizing the use of resources. The results obtained are summarized in Tables III–VII. The first table provides the resources needed to execute a single FFT and an SOS check. The results shows that the FFT is more complex than the SOS check as expected. The difference will be much larger when a fully parallel FFT execution is used

TABLE III
RESOURCES USAGE FOR SINGLE FFT AND SOS CHECK

| | FFT | SOS Checks |
|---|---|---|
| Slices | 1366 | 490 |
| Flip-Flops | 1036 | 140 |
| LUT-4 | 2529 | 970 |

TABLE IV
RESOURCES USAGE FOR FOUR PARALLEL FFTs

| | Un Protected FFT's | ECC Protected | Parity- SOS Protected | Parity SOS-ECC Protected |
|---|---|---|---|---|
| Slices | 5460 | 9890(1.81) | 90089(1.65) | 8552(1.56) |
| Flip-Flops | 4144 | 7780(1.87) | 6046(1.46) | 5988(1.44) |
| LUT-4 | 10114 | 18188(1.80) | 16966(1.68) | 16090(1.59) |

TABLE V
RESOURCES USAGE FOR SIX PARALLEL FFTs

| | Un Protected FFT's | ECC Protected | Parity- SOS Protected | Parity SOS-ECC Protected |
|---|---|---|---|---|
| Slices | 8334 | 14412(1.75) | 13023(1.58) | 12171(1.48) |
| Flip-Flops | 6684 | 11294(1.69) | 8592(1.29) | 8582(1.28) |
| LUT-4 | 15198 | 26571 (1.75) | 24539(1.61) | 23036(1.52) |

TABLE VI
RESOURCES USAGE FOR Eight PARALLEL FFTs

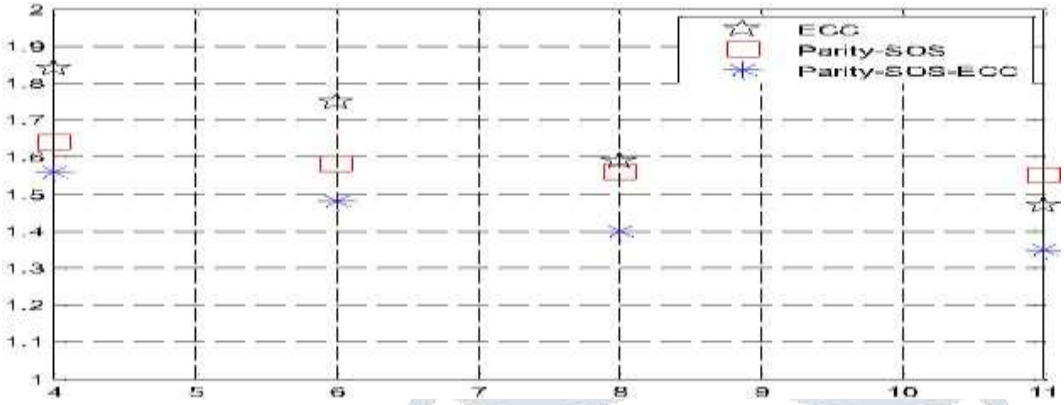
| | Un Protected FFT's | ECC Protected | Parity- SOS Protected | Parity SOS-ECC Protected |
|---|---|---|---|---|
| Slices | 10984 | 17439(1.59) | 17127(1.56) | 15380(1.40) |
| Flip-Flops | 8908 | 12580(1.52) | 11095(1.25) | 10858(1.22) |
| LUT-4 | 20260 | 32265(1.59) | 32316 (1.59) | 29160(1.44) |

TABLE VII
RESOURCES USAGE FOR ELEVEN PARALLEL FFTs

| | Un Protected FFT's | ECC Protected | Parity- SOS Protected | Parity SOS-ECC Protected |
|---|---|---|---|---|
| Slices | 15037 | 21810 (1.81) | 23370(1.55) | 20156(1.34) |
| Flip-Flops | 11407 | 16530 (1.87) | 14720(1.29) | 13648(1.19) |
| LUT-4 | 27825 | 40800 (1.80) | 44272(1.59) | 38520(1.38) |

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

normal from the discussion in Section II and the initial estimates presented in Table II. On the other hand, the parity-SOS system needs less resources than the ECC system when the number of FFTs is 4, 6, or 8 but more when the number of FFTs is 11. This can be explained as in the ECC system, the number of extra FFTs grows logarithmically with the number of FFTs, while in the parity-SOS method, the number of SOS checks grows linearly. This means that as the number of FFTs to protect increases, the ECC system becomes more competitive. For the parity-SOS-ECC system, the number of SOS checks also grows logarithmically and they are simpler to execute than FFTs. Therefore, it remains




X-Axis- No of Parallel FFT's Y-Axis - Overhead (protected /unprotected)

Fig. 6. Overhead comparison for the number of slices

more competitive than the ECC systems regardless of the number of FFTs protected. To better illustrate this phenomenon, the number of slices required for the diverse systems and number of FFTs is plotted in Fig. 6. It can be observed that eight is the value for which equality – SOS and ECC have almost the same cost. For larger values, the ECC system out performs the equality - SOS technique in our execution.

As a summary, the results show that the parity-SOS system outperforms only the ECC system for small number of parallel FFTs, and the parity - SOS- ECC system always provides the best results.

As mentioned before, a key aspect of any fault tolerant system is to validate that it can effectively correct faults. To that end, fault injection experiments have been done on the two proposed systems and the ECC only system. In each simulation run, one fault is inserted to mimic the behavior of soft faults that occur in separation. In particular, 20 000 faults have been at random injected on the registers for the Fourier coefficients and on the RAMs for the results of each stage of the FFT calculation, respectively. For ECC protected parallel FFTs, a acceptance level of 1 is used for the equation checks. For example, in (2), all faults that introduce faults out of range of [−1, 1] were detected and corrected, which is the same as that reported for parallel FIR filters in [17]. For the parity-SOS and parity- SOS -ECC systems, the fault coverage is determined by the tolerance level $\tau$ used in the Perceval check (the absolute difference between the input power and the output power) [4]. In the experiments, we have set $\tau$ = 1, and the fault coverage is ∼99.9%, which is similar to the results reported in [19]. This means that approximately 1 out of 1000 faults will not be correct. Since soft faults are rare events, the residual fault rate will be very low and, therefore, acceptable for many communication and signal processing applications.

## IV. CONCLUSION

In this brief, the protection of parallel FFTs execution against soft faults has been studied. Two techniques have been proposed and evaluated. The proposed techniques are based on combining ⁱan existing ECC approach with the usual SOS check. The SOS checks are used to detect and locate the faults and a simple parity FFT is used for correction. The detection and location of the faults can be done using an SOS check per FFT or instead using a set of SOS checks that form an ECC. FFT and a set of SOS checks that form an ECC, provides the best results in terms of execution complexity. In terms of fault protection, fault injection experiment show that the ECC system can recover all the fault that are out of the tolerance range. The fault coverage for the parity-SOS system and the parity-SOS-ECC system is ∼99.9% when the acceptance level for SOS check is 1

## REFERENCES

[1] N. Kanekawa, E. H. Ibe, T. Suga, and Y. Uematsu, *Dependability in Electronic System's: Mitigation of Hardware Failures, Soft Faults, and Electro-Magnetic Disturbances*. New York, NY, USA: Springer-Verlag, 2010.
[2] R. Bauman, "Soft faults, in advanced computer systems," *IEEE Des.Test Compute.*, vol. 22, no. 3, pp. 258–266, May/Jun.
[3] M. Nicolaids, "Design for soft fault mitigation," *IEEE Trans.Device Mater. Rel.*, vol. 5, no. 3, pp. 405–418, Sep. 2005
[4] A.L.N.Reddy and P.Benerjee"Algorithem- basedfault detection for signal processing applications," *IEEE Trans. Compute.*, vol. 39, no. 10,
a. 1304–1308, Oct. 1990.

[5] T. Hitana and A. Deb, "Bridging con-current and non-concurrent fault detection in FIR filters," in *Proc. Norchip Conf.*, Nov. 2004,
a. 75–78.
[6] S. Pontarelli, G.C.Cardarilli, M. Re, nd A. Salsano, "Totally fault tolerant, RNS FIR filters," in *Proc. 14th IEEE Int. On-Line Test Symp. (IOLTS)*, Jul. 2008, pp. 192–194
[7] B. Shim and N. R. Shanbag, "Energy- efficient, soft-fault-tolerant digi signal procesing," *IEEE Trans. Very Large Scale Integra. (VLSI) Syst.*, vol. 14, no. 4, pp. 336–348, Apr. 2006
[8] E. P. Kim and N. R. Shanbag, "Soft N-modular -redundancy," *IEEE Trans. Compute.*, vol. 61, no. 3, pp. 323–336, Mar. 2012.
[9] J. Y. Jou and J. A. Abraham, "Fault – tolerant- FFT networks," *IEEE -Trans. Compute.*, vol. 37, no. 5, pp. 548–561, May 1988
[10] S.-J. Wang and N. K. Jha,- "Algorithm ,based fault, tolerance for FFT network's," *IEEE -Trans. Compute.*, 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 Execution*. 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 , andThe UMTS, Long Term -Evolution, .From Theory to Practice*, 2nd . 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; fault -detection and correction, for parallel filters," *IET Electron. Let.*, vol. 48, no. 20, pp. 1258–1260, Sep. -2012
[17] Z. Gao *et al.*, "Fault tolerant parallel filter's based on fault correction codes,"*IEEE Trans. very Large Scale Integi. (VLSI) Syst.*, vol. 23, no. 2, pp. 384–387, Feb. 2015
[18] R. W. Hamming,"Fault- detecting and fault correcting codes," *Bell Syst. Tech. J.*, vol. 29, no. 2, pp. 147 –160, Apr. 1950
[19] P. Reviriego, C. J. -Bleakley, and J. A. Maestro, "A novel concurrent- fault-, detecting -technique, for the fast Fourier transform," in *Proc. ISSC*, Maynooth, Ireland,- Jun. 2012, pp. 1–5