Fault Tolerant Parallel FFTs Using Error Correction Codes and Parseval Checks

Shaik.Imran, Jabeena shaik,,M.Tech
Department of Electronics and communication Engineering
Quba College of Engineering & Technology,Venkatachalam
(JNTU Anantapur)

***Abstract*— Soft errors pose a reliability threat to modern electronic circuits. This makes protection against soft errors a requirement for many applications. Communications and signal processing systems are no exceptions to this trend. For some applications, an interesting option is to use algorithmic-based fault tolerance (ABFT) techniques that try to exploit the algorithmic properties to detect and correct errors. Signal processing and communication applications are well suited for ABFT. One example is fast Fourier transforms (FFTs) that are a key building block in many systems. Several protection schemes have been proposed to detect and correct errors in FFTs. Among those, probably the use of the Parseval or sum of squares check is the most widely known. In modern communication systems, it is increasingly common to find several blocks operating in parallel. Recently, a technique that exploits this fact to implement fault tolerance on parallel filters has been proposed. In this brief, this technique is first applied to protect FFTs. Then, two improved protection schemes that combine the use of error correction codes and Parseval checks are proposed and evaluated. The results show that the proposed schemes can further reduce the implementation cost of protection.**

***Index Terms*— Error correction codes (ECCs), fast Fourier transforms (FFTs), soft errors.**

I. Introduction

The complexity of communications and signal processing circuits increases every year. This is made possible by the CMOS technology scaling that enables the integration of more and more transistors on a single device. This increased complexity makes the circuits more vulnerable to errors. At the same time, the scaling means that transistors operate with lower voltages and are more susceptible to errors caused by noise and manufacturing variations [1]. The importance of radiation-induced soft errors also increases as technology scales [2]. Soft errors can change the logical value of a circuit node creating a temporary error that can affect the system operation. To ensure that soft errors do not affect the operation of a given circuit, a wide variety of techniques can be used [3]. These include the use of special manufacturing processes for the integrated circuits like, for example, the silicon on insulator. Another option is to design basic circuit blocks or complete design libraries to minimize the probability of soft errors. Finally, it is also possible to add redundancy at the system level to detect and correct errors.

Manuscript received September 10, 2014; revised November 24, 2014 and February 12, 2015; accepted February 26, 2015. Date of publication March 11, 2015; date of current version January 19, 2016. This work was supported in part by the China's 863 Plan Program under Grant 2012AA01A502, in part by the Beijing Natural Science Foundation under Grant 4110001, in part by the National Basic Research Program of China under Grant 2012CB316000, in part by the National Natural Science Foundation of China under Grant 61402044, and in part by the Spanish Ministry of Science and Education under Grant AYA2009-13300-C03.

Z. Gao is with the School of Electronic Information Engineering, Tianjin University, Tianjin 300072, China (e-mail: zgao@tju.edu.cn).

P. Reviriego and J. A. Maestro are with the Universidad Antonio de Nebrija, Madrid E-28040, Spain (e-mail: previrie@nebrija.es; jmaestro@nebrija.es).

Z. Xu is with the School of Information and Communication Engineering, Beijing Information Science and Technology University, Beijing 100085, China (e-mail: xuzhan@tsinghua.edu.cn).

X. Su, M. Zhao, and J. Wang are with the Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: suxin@tsinghua.edu.cn; zhaoming@tsinghua.edu.cn; wangj@tsinghua.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2015.2408621

One classical example is the use of triple modular redundancy (TMR) that triples a block and votes among the three outputs to detect and correct errors. The main issue with those soft errors mitigation techniques is that they require a large overhead in terms of circuit implementation. For example, for TMR, the overhead is >200%. This is because the unprotected module is replicated three times (which requires a 200% overhead versus the unprotected module), and additionally, voters are needed to correct the errors making the overhead >200%. This overhead is excessive for many applications. Another approach is to try to use the algorithmic properties of the circuit to detect/correct errors. This is commonly referred to as algorithm-based fault tolerance (ABFT) [4]. This strategy can reduce the overhead required to protect a circuit.

Signal processing and communications circuits are well suited for ABFT as they have regular structures and many algorithmic properties [4]. Over the years, many ABFT techniques have been proposed to protect the basic blocks that are commonly used in those circuits. Several works have 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 an alternative to TMR but with a lower cost [7]. The knowledge of the distribution of the filter output has also been recently exploited to detect and correct errors with lower overheads [8]. The protection of fast Fourier transforms (FFTs) has also been widely studied [9], [10].

As signal-processing circuits become more complex, it is common to find several filters or FFTs operating in parallel. This occurs for example in filter banks [11] or in multiple-input multiple-output (MIMO) communication systems [12]. In particular, MIMO orthogonal frequency division modulation (MIMO-OFDM) systems use parallel iFFTs/FFTs for modulation/demodulation [13]. MIMO-OFDM is implemented on long-term evolution mobile systems [14] and also on WiMax [15]. The presence of parallel filters or FFTs creates an opportunity to implement ABFT techniques for the entire group of parallel modules instead of for each one independently. This has been studied for digital filters initially in [16] where two filters were considered. More recently, a general scheme based on the use of error correction codes (ECCs) has been proposed [17]. In this technique, the idea is that each filter can be the equivalent of a bit in an ECC and parity check bits can be computed using addition. This technique can be used for operations, in which the output of the sum of several inputs is the sum of the individual outputs. This is true for any linear operation as, for example, the discrete Fourier transform (DFT).

In this brief, the protection of parallel FFTs is studied. In particular, it is assumed that there can only be a single error on the system at any given point in time. This is a common assumption when considering the protection against radiation-induced soft errors [3]. There are three main contributions in this brief.

1) The evaluation of the ECC technique [17] for the protection of parallel FFTs showing its effectiveness in terms of overhead and protection effectiveness.
2) The proposal of a new technique based on the use of Parseval or sum of squares (SOSs) checks [4] combined with a parity FFT.
3) The proposal of a new technique on which the ECC is used on the SOS checks instead of on the FFTs.

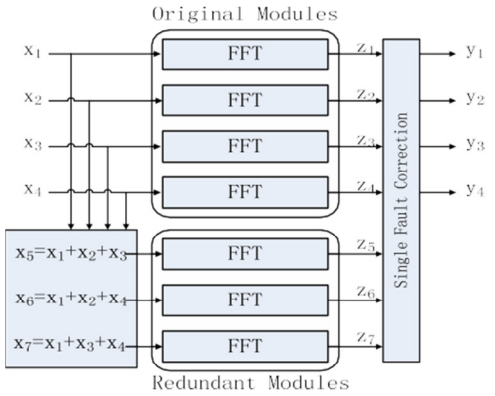

Fig. 1. Parallel FFT protection using ECCs.

The two proposed techniques provide new alternatives to protect parallel FFTs that can be more efficient than protecting each of the FFTs independently.

The proposed schemes have been evaluated using FPGA implementations to assess the protection overhead. The results show that by combining the use of ECCs and Parseval checks, the protection overhead can be reduced compared with the use of only ECCs as proposed in [17]. Fault injection experiments have also been conducted to verify the ability of the implementations to detect and correct errors.

The rest of this brief is organized as follows. Section II presents the two proposed schemes. In Section III, the implementation overheads and fault tolerance of the schemes are evaluated. Finally, the conclusions are drawn in Section IV.

II. Proposed Protection Schemes for Parallel FFTs

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

$$x_5 = x_1 + x_2 + x_3 \tag{1}$$

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

$$z_5 = z_1 + z_2 + z_3. \tag{2}$$

This will be denoted as c_1 check. The same reasoning applies 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 error has occurred can be determined. The different patterns and the corresponding errors are summarized in Table I. Once the module in error is known, the error can be corrected by reconstructing its output using the remaining modules. For example, for an error affecting z_1, this can be done as follows:

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

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

The overhead of this technique, as discussed in [17], is lower than TMR as the number of redundant FFTs is related to the logarithm

TABLE I
Error Location in the Hamming Code

$c_1\,c_2\,c_3$	Error Bit Position
0 0 0	No error
1 1 1	z_1
1 1 0	z_2
1 0 1	z_3
0 1 1	z_4
1 0 0	z_5
0 1 0	z_6
0 0 1	z_7

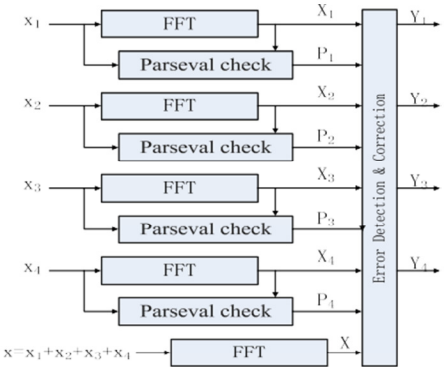

Fig. 2. Parity-SOS (first technique) fault-tolerant parallel FFTs.

of the number of original FFTs. For example, to protect four FFTs, three redudant FFTs are needed, but to protect eleven, the number of redundant FFTs in only four. This shows how the overhead decreases with the number of FFTs.

In Section I, it has been mentioned 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 detect errors. The SOS check is based on the Parseval 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 errors with low overhead as one multiplication is needed for each input or output sample (two multiplications and adders for SOS per sample).

For parallel FFTs, the SOS check can be combined with the ECC approach to reduce the protection overhead. Since the SOS check can only detect errors, the ECC part should be able to implement the correction. This can be done using the equivalent of a simple parity bit for all the FFTs. In addition, the SOS check is used on each FFT to detect errors. When an error is detected, the output of the parity FFT can be used to correct the error. This is better explained with an example. In Fig. 2, the first proposed scheme is illustrated for the case of four parallel FFTs. A redundant (the parity) FFT is added that has the sum of the inputs to the original FFTs as input. An SOS check is also added to each original FFT. In case an error is detected (using P_1, P_2, P_3, P_4), the correction can be done by recomputing the FFT in error using the output of the parity FFT (X) and the rest of the FFT outputs. For example, if an error occurs in the first FFT, P_1 will be set and the error can be corrected by doing

$$X_{1c} = X - X_2 - X_3 - X_4. \tag{4}$$

This combination of a parity FFT and the SOS check reduces the number of additional FFTs to just one and may, therefore, reduce the protection overhead. In the following, this scheme will be referred to as parity-SOS (or first proposed technique).

Another possibility to combine the SOS check and the ECC approach is instead of using an SOS check per FFT, use an

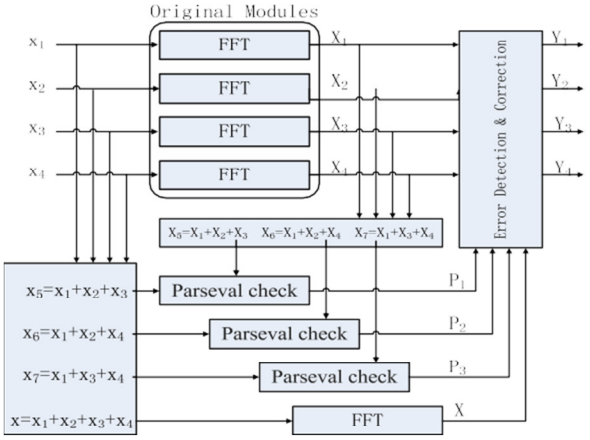

Fig. 3. Parity-SOS-ECC (second technique) fault-tolerant parallel FFTs.

TABLE II
OVERHEAD OF THE DIFFERENT SCHEMES TO PROTECT k FFTS

	FFTs	SOS checks
ECC	$1+\log_2(k)$	0
Parity-SOS	1	k
Parity-SOS-ECC	1	$1+\log_2(k)$

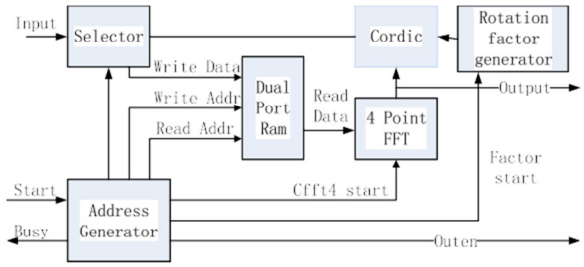

Fig. 4. Architecture of the FFT implementation.

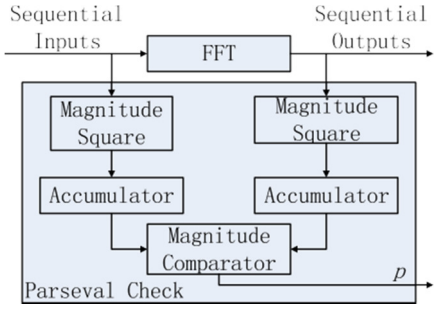

Fig. 5. Implementation of the SOS check.

ECC for the SOS checks. Then as in the parity-SOS scheme, an additional parity FFT is used to correct the errors. This second technique is shown in Fig. 3. The main benefit over the first parity-SOS scheme is to reduce the number of SOS checks needed. The error location process is the same as for the ECC scheme in Fig. 1 and correction is as in the parity-SOS scheme. In the following, this scheme will be referred to as parity-SOS-ECC (or second proposed technique).

The overheads of the two proposed schemes can be initially estimated using the number of additional FFTs and SOS check blocks needed. This information is summarized 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 schemes reduce the number of additional FFTs to just one. In addition, the second technique also reduces the number of SOS checks. In Section III, 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 added for protection. For the ECC technique, the protection of these 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 Figs. 2 and 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.

III. EVALUATION

The two proposed schemes and the ECC scheme presented in [17] have been implemented on an FPGA and evaluated 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 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, a 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 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. This is shown in Fig. 5. To minimize the impact of roundoffs on the fault coverage, the outputs of the accumulator are 39-bit wide. For the evaluation, several values of the number of parallel FFTs are considered. This is done to compare the different techniques as a function of the number of parallel FFTs in the original system.

The error detection and correction blocks (Figs. 1–3) are implemented as multiplexers that select the correct output depending on the error pattern detected. As mentioned before, these blocks are tripled to ensure that errors that affect them do not corrupt the final outputs.

The FFT and the different protection techniques have been implemented 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 implement a single FFT and an SOS check. The results show that the FFT is more complex

TABLE III
RESOURCES USAGE FOR A SINGLE FFT AND SOS CHECK

	FFT	SOS Check
Slices	1367	494
Flip-Flop	1037	141
LUT-4	2530	974

TABLE IV
RESOURCES USAGE FOR FOUR PARALLEL FFTS

	Unprotected FFTs	ECC protected	Parity-SOS protected	Parity-SOS-ECC protected
Slices	5468	9890 (1.81)	9009 (1.65)	8552 (1.56)
Flip-Flop	4148	7780 (1.87)	6047 (1.46)	5988 (1.44)
LUT-4	10120	18188 (1.80)	16968 (1.68)	16092 (1.59)

TABLE V
RESOURCES USAGE FOR SIX PARALLEL FFTS

	Unprotected FFTs	ECC protected	Parity-SOS protected	Parity-SOS-ECC protected
Slices	8238	14412 (1.75)	13024 (1.58)	12171 (1.48)
Flip-Flop	6684	11294 (1.69)	8593 (1.29)	8584 (1.28)
LUT-4	15198	26571 (1.75)	24539 (1.61)	23036 (1.52)

TABLE VI
RESOURCES USAGE FOR EIGHT PARALLEL FFTS

	Unprotected FFTs	ECC protected	Parity-SOS protected	Parity-SOS-ECC protected
Slices	10984	17439 (1.59)	17127 (1.56)	15382 (1.40)
Flip-Flop	8912	13580 (1.52)	11095 (1.25)	10858 (1.22)
LUT-4	20264	32265 (1.59)	32319 (1.59)	29164 (1.44)

TABLE VII
RESOURCES USAGE FOR 11 PARALLEL FFTS

	Unprotected FFTs	ECC protected	Parity-SOS protected	Parity-SOS-ECC protected
Slices	15037	21811 (1.45)	23378 (1.55)	20156 (1.34)
Flip-Flop	11407	16533 (1.45)	14727 (1.29)	13648 (1.19)
LUT-4	27830	40805 (1.47)	44273 (1.59)	38528 (1.38)

than the SOS check as expected. The difference will be much larger when a fully parallel FFT implementation is used.

Tables IV–VII show 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. This is expected from the discussion in Section II and the initial estimates presented in Table II. On the other hand, the parity-SOS scheme needs less resources than the ECC scheme 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 scheme, the number of additional FFTs grows logarithmically with the number of FFTs, while in the parity-SOS technique, the number of SOS checks grows linearly. This means that as the number of FFTs to protect increases, the ECC scheme becomes more competitive. For the parity-SOS-ECC scheme, the number of SOS checks also grows logarithmically and they are simpler to implement than FFTs. Therefore, it remains

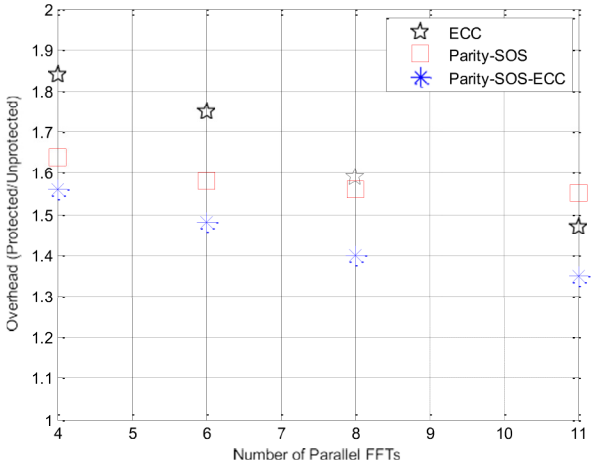

Fig. 6. Overhead comparison for the number of slices.

more competitive than the ECC scheme regardless of the number of FFTs protected. To better illustrate this phenomenon, the number of slices required for the different schemes and number of FFTs is plotted in Fig. 6. It can be observed that eight is the value for which parity-SOS and ECC have almost the same cost. For larger values, the ECC scheme outperforms the parity-SOS technique in our implementation.

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

As mentioned before, a key aspect of any fault tolerant scheme is to validate that it can effectively correct errors. To that end, fault injection experiments have been done on the two proposed schemes and the ECC only scheme. In each simulation run, one error is inserted to mimic the behavior of soft errors that occur in isolation. In particular, 20 000 errors have been randomly 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 tolerance level of 1 is used for the equation checks. For example, in (2), all faults that introduce errors 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 schemes, the fault coverage is determined by the tolerance level τ used in the Parseval 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 errors will not be corrected. Since soft errors are rare events, the residual error 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 implementation against soft errors has been studied. Two techniques have been proposed and evaluated. The proposed techniques are based on combining an existing ECC approach with the traditional SOS check. The SOS checks are used to detect and locate the errors and a simple parity FFT is used for correction. The detection and location of the errors can be done using an SOS check per FFT or alternatively using a set of SOS checks that form an ECC.

The proposed techniques have been evaluated both in terms of implementation complexity and error detection capabilities. The results show that the second technique, which uses a parity

FFT and a set of SOS checks that form an ECC, provides the best results in terms of implementation complexity. In terms of error protection, fault injection experiments show that the ECC scheme can recover all 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 check is 1.

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 digital 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.

[17] Z. Gao *et al.*, "Fault tolerant parallel filters based on error correction codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 2, pp. 384–387, Feb. 2015.

[18] R. W. Hamming, "Error detecting and error 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 error detection technique for the fast Fourier transform," in *Proc. ISSC*, Maynooth, Ireland, Jun. 2012, pp. 1–5.