International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 19 (2017) pp. 8316-8322

Analysis and Implementation of Fault Tolerant Parallel FFTs & IFFTs Using Parity SOS and ECC

Afroz Banu Syed [1] and Rani R Kodali [2]

[1]Post Graduation Student, [2]Associate Professor

[1,2] Department of Electronics & Communication Engineering, Jawaharlal Nehru Technological University, Kakinada, Lakireddy Bali Reddy College of Engineering (Autonomous), L.B.Reddy Nagar, Mylavaram-521230, Andhra Pradesh, India.

[1]Orcid Id: 0000-0002-4831-803X

Abstract

Soft error is an error in which, data is corrupted in communication systems and mismatch between the system's description vis-à-vis the actual description. To detect and correct these errors, fault tolerant parallel FFTs & IFFTs are used. FFTs is a fast and efficient algorithm used in- speech processing, communications, signal processing, spectrum analysis, modulation scheme OFDM. 'Error Correction Code(ECC)' and 'Parseval's check(s)' or 'Parseval's Theorem' or 'Parity Sum of squares(SOS)' check(s) are the transformation techniques used to detect and correct errors and improve the accuracy of the system. As a part of this work/paper-(a) 4 point, 8 point and 16 point FFTs & IFFTs were simulated using MATLAB (b) Xilinx was used to implement and synthesize the "modified architecture of scalable FFTs & IFFTs"- thus derive the process delay and area (c) EDA tool, 'Cadence' - was used to evaluate- area, power consumption and speed. For instance, in comparison to the existing published work [10] where- the number of slices is 9,890; and with proposed architecture is 286. Thus an effective reduction of 97% is observed in the case of ECC in 4-point FFTs.

Keywords: Soft Error, Fourier Transforms, Hamming Code, Error Correction Code (ECC), Parity-Sum Of Squares (SOS), and CORDIC.

INTRODUCTION

Signal is a data or message that is to be transferred between transmitter and receiver. Selecting the analog signal values in discrete time instants is sampling; the reciprocal of the time interval between the samples is sampling rate. In sampling rate conversion, if the base signal is analog it passes through an Analog to Digital (A/D) convertor and then through a filter, which filters the output (if necessary) then it proceeds for resample at the required sampling rate. If base signal is available in digital domain then sampling rate conversion [1] is directly applied. For the sake of simplification, in this paper, signal in its digital domain is considered. This also helps in avoiding the signal distortions while A/D conversion.

An unwarranted change in an instruction of a data i.e. an error, it can occur in a computer's memory. It can be termed a soft error [2]. It will not harm a system's hardware; but can mash-up the data while under execution. This is different from a failure- in a memory chip of a computer which is termed as hard error. However, the difference between them is- the latter is not rectified while rebooting the computer. The remedy in case of hard error is to replace the memory chip or entire module.

'System-level soft error', these errors are due to the fusion of data and noise phenomenon when the data is being processed. The computer tries to read the noise as a 'data bit', which can be a reason for errors in processing program code. The error data bit can even be saved in memory and effect data, at a later time.

In every message which is being transmitted an error may occur- "Soft error" is one such error. Error correction is an automatic correction of errors, while the digital data is in transmission. FFTs rapidly compute unwarranted changes by factorizing the DFTs (Discrete Fourier Transforms) into a product of sparse (mostly zero) factors. FFTs are highly accurate and consume minimum area.

TECHNIQUES TO DETECT AND CORRECT

1) TMR (Triple Modular Redundancy) [3] triples a block and votes among the three outputs to detect and correct errors. But its major drawback is large overhead in terms of circuit.
2) ABFT (Algorithm Based Fault tolerance) [4] protects the basic blocks that are commonly used in circuits. Its advantage is to keep the process in progress without interruption and flag the error at the end of cycle.
3) Error Correction Code (ECC) is a code that corrects an error in a process of addition of parity data (redundant data), to a communication (or message)- such that it can be recovered by receiver, even when a number of

errors (the capability in code being used) get introduced either in the processing of transmission of data or on storage.

4) Parity SOS (Sum Of Squares) checks for equality between (a) the product of square of number of points (N) with the sum of square of inputs and (b) the sum of square of outputs with rotation factor generator of FFTs & IFFTs.

Error Correction & Detection Code

A brief about algorithm for Hamming code technique [5] - is as follows:

- To form a code word of 'n+k' bits, an 'n'-bit data word is appended to the 'k' parity bits;
- The number sequence is from '1' to 'n+k' for the 'bit positions';
- These bit positions are numbered through the "powers of two"; as '2^0' to '2^{n+k-1}'; residual are 'parity bits' and 'data bits';
- 'XOR operation' is done on a set of 'data bits' to calculate the respective set of 'parity bits';
- Say, combination of data bits are :

  P1 = 1 ⊕ 3 ⊕ 5 ⊕ 7 ⊕ 9 ⊕ 11 ⊕ 13...

  P2 = 2 ⊕ 3 ⊕ 6 ⊕ 7 ⊕ 10 ⊕ 11…

  P4 = 4 ⊕ 5 ⊕ 6 ⊕ 7 ⊕ 12 ⊕ 13....

  P8 = 8 ⊕ 9 ⊕ 10 ⊕ 11 ⊕ 12 ⊕ 13 ⊕ 14 ⊕ 15..

  P16 = 16 ⊕ 17 ⊕ 18 ⊕ 19 ⊕ 20 ⊕ 21 ⊕ 22….
- In order to check for error, all parity bits are run through a checker bit. Say,

  C1 = 1⊕ 3⊕ 5⊕ 7⊕ 9⊕ 11⊕ 13...

  C2 = 2⊕ 3⊕ 6⊕ 7⊕ 10⊕ 11...

  C4 = 4⊕ 5⊕ 6⊕ 7 ⊕12....

  C8 = 8⊕ 9 ⊕10 ⊕ 11⊕ 12⊕ 13⊕ 14 ⊕ 15..

  C16 = 16 ⊕ 17 ⊕ 18 ⊕ 19 ⊕ 20 ⊕ 21 ⊕ 22….
- In case of no error, C = C16 C8 C4 C2 C1 =00000; else, if C ≠ 00000, then a 4 bit binary sequence is generated, and it indicates the location of the error bit among checker bit set.

Fast Fourier Transforms (FFTs)

In digital signal processing- to compute the DFTs; FFTs algorithms are used, with a given criteria that the size N (number of points of DFTs) is either 'the power of two' or 'the power is four' [1].

$$X(k) = \sum_{n=0}^{N-1} x(n) * W_N^{nk} \; ; k = 0, 1, 2, \ldots . N-1$$

Where $W_N^{nk} = e^{-(j*2*\pi*n*k)/N}$

As from the above expression FFTs [8] are defined as- the product of 'the sum of the sequences of 'n' from '0' to 'N-1' of the input signal 'x (n)' with 'the twiddle factor of the number of points in FFTs'. There are 'N' complex multiplications and 'N-1' complex additions. Therefore, to compute all the 'N' values of DFTs, 'N^2' complex multiplications and 'N*(N-1)' complex additions are necessary. Number of complex multiplications is reduced from 'N^2' to '$N\log_2 (N-1)$' [6].

In FFTs, the computation process runs in two stages of Decimations [7]. They are-

(a) Decimation-in-time

$$W_N^{nk} = e^{-(j*2*\Pi*n*k)/N}$$

(b) Decimation-in-frequency

$$W_N^{-nk} = e^{(j*2*\Pi*n*k)/N}$$

And they have 'N/2' butterfly structure. IFFTs constitute Decimation-in-frequency. As the name itself suggests- the inverse operation of FFTs is IFFTs as shown below:

$$x(n) = (1/N) \sum_{k=0}^{N-1} X(k) * W_N^{-nk} \; ; n = 0, 1, 2, \ldots . N-1$$

Where $W_N^{-k} = e^{\frac{j*2*\pi*n*k}{N}}$

Design Flow of Proposed Technique

In Figure 1, the flowchart represents the chain of the scalable parallel FFTs and IFFTs using Error Correction Code (ECC) and Parity Sum Of Square (SOS), which is the design of research.

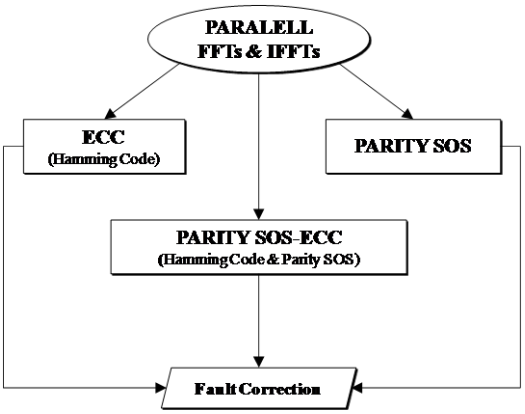

Figure 1: Flowchart of the design

The design explanation is as follows:

Modified Architecture of the FFTs Implementation

As explained in the Figure 2, input sequence (d_{in}) is given to

the selector (s), where it selects the input and forwards the output (d_{out}) to the CORDIC. In the discipline of digital signal processing, COordinate Rotational DIgital Computer (CORDIC) algorithm initially propounded by Volder-has two functional modes; (a) the vectoring mode (classical) and (b) the rotation mode, which are computationally different [9]. The algorithms of the two modes can be implemented only in shift operations (which are rotations by a fixed rotational angles / micro rotations but with variable rotational direction) and subtractions / additions. It is very simple and also an efficient algorithm. Because of its simple operations, CORDIC Algorithms are suitable in VLSI implementation.

CORDIC algorithms are not exact rotations; they also involve the combinations of multiplications with sine and cosine. If necessary the rotated vectors are scaled assembling a scale factor correction. The applications are - realizations of rotations, calculations of trigonometric functions and inverse trigonometric functions (based on vectoring mode); divisions and multiplications, logarithms, exponentials, digital filters, linear transforms, square rooting, linear functions and extended hyperbolic functions. The fields influenced by CORDIC algorithm are Digital Signal Processing, Image Processing, 3D Graphics and Robotics.

The base architecture proposed by Zhen Gao et.al. [10], has been modified; (1) with a scalability aspect of N point FFTs and (2) IFFTs. With reference to the base architecture, the rotation factor generators / twiddle factor is amalgamated into CORDIC. The angle rotation computation has been conducted using trigonometric functions without any losses in functionality.

The feature of dual port random access memory (RAM) with addressable memory is used. The ports provide high bandwidth and buffering. The other key characteristics are (a) interference mismatch resolution; (b) Lower power dissipation (c) provision of flexible interferences (like port independent clock source, Input /Output standards, and bus widths) (d) multiple parallel reads and or writes (unlikely single-ported RAM which only allows one access at a time).

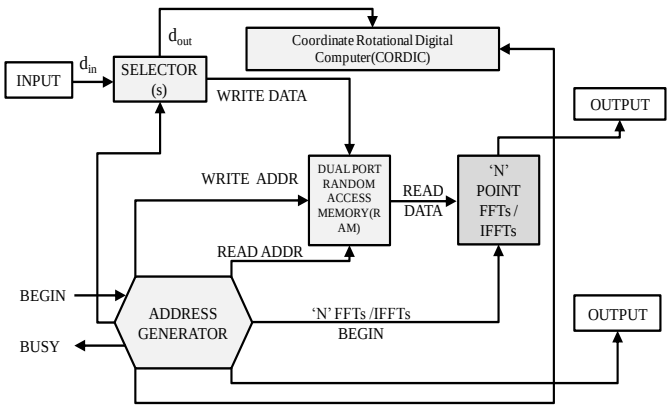

Figure 2: Modified Architecture of Scalable FFTs/IFFTs

The Address generator module has also been continued. This points the location of required elements and while the search operation is in progress it indicates the "busy" status to the next element in the waiting.

The scalable FFTs/IFFTs module of this paper executes in three techniques-

1. ECC by hamming code,
2. Parity SOS – individually; as well as
3. Combinations of both i.e. to realize the objective of fault tolerant FFTS using the potent combination of ECC and parity SOS.

Especially for scalability aspect, the N point FFTs / IFFTs have been demonstrated for 4 point, 8 point and 16 point.

The simulations executed in connection to this paper take 4, 8, 16-Point FFTs/IFFTs and feed them the inputs of 8-bit wide and the observed outputs are also 8-bit wide. The simulation was done using MATLAB-14b. Xilinx was also used to implement and synthesize. The objective of this was to observe the Delay and area of the "modified architecture of FFTs & IFFTs". This design was done in Virtex-4 xc4vlx80-12ff1148 FPGA. Further the EDA tool, 'Cadence' was used to-(a) the area, (b) power consumption and (c) the speed.

Use of Hamming Code as ECC for Protection Of FFTs / IFFTs

With reference to the below Figure 3; where input of 4 point FFTs / IFFTs (in the form of X1, X2, X3, and X4) is given and outputs of FFTs are Z1, Z2, Z3, and Z4. P1, P2, and P3 are the parity bits of hamming code with the XOR operation; being forwarded as inputs to the redundant module of FFTs / IFFTs. C1, C2, C3, and C4 are the check bits; which are the outputs of redundant module. Signal fault correction is done to get the output of 4 point parallel FFTs / IFFTs and hamming code is used to detect and correct the errors using Error Correction Codes to get Yc1, Yc2, Yc3, and Yc4.

Here single error correction "Hamming code" is used. In the redundant modules three FFTs/IFFTs are included for detection and correction of errors. Also, the inputs of the three redundant modules are linear combinations.

In here, these three are used to verify the outputs of linear combinations. The calculation of error correction hamming code is

$$P1 = X1 \oplus X2 \oplus X3$$

Each equation's output will become the input to the each redundant module. The calculations of three redundant modules are as shown below:

$$C1 = Z1 \oplus Z2 \oplus Z3$$

The C1, C2, and C3 redundant modules are denoted as check bits. By observing the differences in each of the checks, the error will appear on any one of the modules.

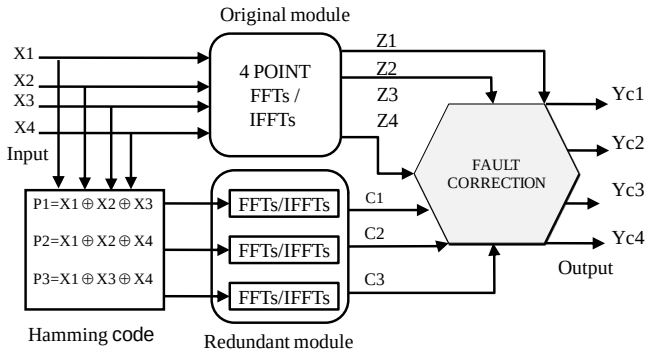

Figure 3: Parallel FFTs / IFFTs using hamming code for ECC with protection (first technique)

If the error in the module appears, that error- will be corrected by reusing the output of redundant module with the remaining modules. Consider, an error in C1 as shown in (1), similarly errors in C2 & C3.

$$Z2 = C1 - Z1 - Z3 \ldots\ldots\ldots (1)$$

For the protection of elements in hamming code ECC technique was elaborated in [11].

Detection of Error Using Parity SOS in Fault Tolerant FFTs / IFFTs

In fault tolerant parallel FFTs / IFFTs, the parity Sum Of Squares (SOS) check [4] detect the errors alone. If an error is detected through the parity SOS, the parity FFT output can be used to correct the error. The explanation of this is in Figure 4. It is the basic structure of parity SOS.

The Figure 4 can be elaborated as- Parity SOS checks for equality between (a) product of - square of number of points and sum of squares of inputs in FFTs / IFFTs; with (b) sum of square of outputs with rotational factor generators in FFTs/IFFTs.

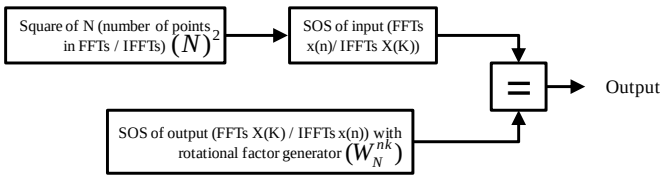

Figure 4: Parity Sum Of Squares (SOS) check

Parseval's check/ parity SOS check:

The FFT is given by,

$$x(n) = \sum_{n=0}^{N-1} X(k) * W_N^{nk}; k = 0,1,2,...N-1$$

Where $W_N^k = e^{-j*2*\Pi*k/N}$ is the N[th] root of unity and is called the twiddle factor. Parseval's check for digital sequences, can be expressed as

$$(N * \sum_{n=0}^{N-1} x(n))^2 = (\sum_{k=0}^{N-1} X(k) * W_N^{nk})^2$$

Where x (n) are the samples of the signal in the time domain, X (k) are the samples of the signal in the frequency domain. In Figure 5, each input of four points fault tolerant parallel FFT/IFFT is used in the parity SOS check that has the sum of squares of inputs of the original FFTs X1, X2, X3, and X4 is equated to the sum of squares of each output in FFTs / IFFTs Z1, Z2, Z3, and Z4. The result of this can get the error location in four points FFTs. Same procedure is implemented in IFFTs.

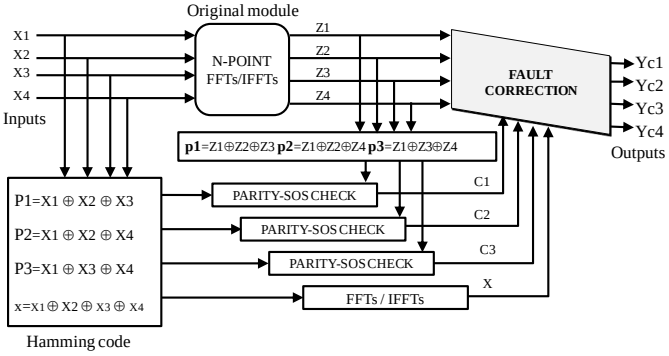

Figure 5: Fault tolerant parallel FFTs / IFFTs using Parity Sum of Square (SOS) check (second technique)

While comparing, if these are equal consider output as '0' (which means there is no error) otherwise it is considered as '1' (which means error is detected among any of these P1, P2, P3, and P4). The correction is done by recompiling the FFT in error using the output of the parity FFT (X) and the rest of the FFT outputs. In parity SOS check, error is detected. If an error is detected in the FFTs / IFFTs at P1 can be corrected with this equation.

$$X1 = X - X2 - X3 - X4 \ldots\ldots.. (2)$$

If error is detected, that error- will be corrected by reusing the inputs in FFTs. Consider, an error in P1 as shown in (2), similarly errors in P2, P3 & P4

To Realize the Objective of Fault Tolerant parallel FFTs & IFFTs Using the Potent Combination of ECC and Parity SOS

To enhance the performance and protection levels- the merger of parity SOS check and the hamming code of ECC is done. This is explained in the schematic below Figure 5.

Where, an extra Fault Tolerant parallel FFTs / IFFTs is used to correct the errors (similar to the description provided in Figure 5: parity-SOS for detection of errors) has been introduced. It is introduced along with Parity checks of Hamming Code. In this proposed technique there will be a reduction of the number of SOS check-runs against standalone parity-SOS technique. Error address detection process is similar to the parity-SOS technique as shown in Figure 5. The Hamming code process in the below figure is similar to the Hamming code for ECC methodology as explained in Figure 3.

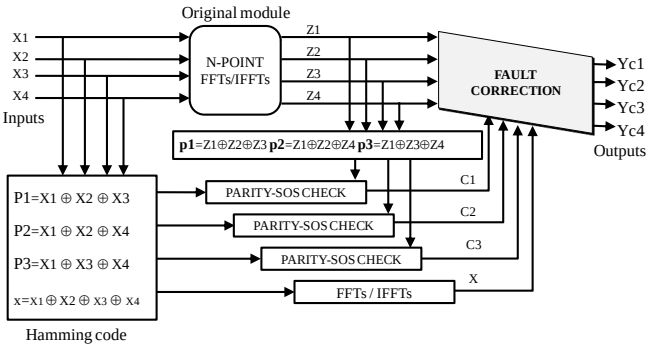

Figure 6: Fault tolerant parallel FFTs / IFFTs using ECC & Parity Sum Of Square (SOS) check (third technique)

As explained earlier, hamming code in ECC will operate its calculations in XOR operation. These outputs (P1, P2 and P3) of hamming code are given as inputs to the parity-SOS check. Whereas the outputs of original module N- point FFTs / IFFTs (Z1, Z2, Z3, and Z4) will perform linear combinations among themselves in XOR operation; and are termed as p1, p2 and p3 (parity bits). These (p1, p2 and p3) will perform parity-SOS check operation and their outputs are C1, C2, and C3. Through fault correction- error is detected and corrected and gets reflected as the corrected output in the form of -Yc1, Yc2, Yc3, and Yc4.

The final observation here is- the proposed ECC scheme can detect all errors without any predefined threshold limits. Also this proposed architecture of SOS checks detects most errors and with high probability detects all errors. Therefore, to validate the three techniques' performance with the existing literature [10] within the provided parameters, a summary of results in tabulated below.

Comparative Analysis of Results Vis-À-Vis Existing Literature

The parameters provided by the existing literature are

1. Slices
2. Flip-Flops
3. 4-Look up Tables (LUT)
4. Delay

Table 1: Parametric comparison between existing and proposed 4 point-FFTs performance

Parameters	ECC		Parity SOS		Parity SOS ECC	
	Prop	Ref.[10]	Prop.	Ref.[10]	Prop.	Ref.[10]
Slices	286	9,890	371	9,009	376	8,552
Flip Flops	454	7,780	484	6,047	491	5,988
4-LUT	492	18,188	633	16,968	656	16,092
Delay (ns)	3.972	Not. Avlb.	6.455	Not. Avlb.	6.455	Not Avlb.

The same have been captured from the simulations conducted (in Xilinx) and compared with the values of the base architecture [10] as tabulated in Table 1 and Table 2.

Table 2: Parametric comparison between existing and proposed 8 point -FFTs performance

Parameters	ECC		Parity SOS		Parity SOS ECC	
	Prop.	Ref.[10]	Prop.	Ref.[10]	Prop.	Ref.[10]
Slices	739	17,439	861	17,127	580	15,382
Flip Flops	644	13,580	688	11,095	324	10,858
4-LUT	1290	32,265	1613	32,319	1079	29,164
Delay (ns)	11.118	Not Avlb.	15.615	Not Avlb.	12.162	Not Avlb.

SUMMARY OF RESULTS

The same parameters as indicated in the above tables (captured along other details) have been tabulated below. Firstly, Table 3 indicates-16 point- Fault Tolerant Parallel FFTs performance with proposed architecture as derived from Xilinx.

Table 3: Parametric performance of 16 point –FFTs

Parameters	ECC	Parity SOS	Parity SOS ECC
Slices	2236	2054	2832
Flip Flops	127	128	1082
4-LUT	4106	3763	5377
Delay (ns)	37.167	32.672	27.720

Secondly, Table 4 indicates- 16 point Fault Tolerant Parallel FFTs is compared with IFFTs (Parity SOS-ECC)

Table 4: Parametric comparison between 16 point –FFTs and 16 point IFFTs (Parity SOS ECC).

Parameters	16 point FFTs-Parity SOS ECC	16 point IFFTs-Parity SOS ECC
Slices	2832	903
Flip Flops	1082	671
4-LUT	5377	1714
Delay (ns)	27.720	21.919

The simulation is further extended using EDA tool i.e. Cadence.

Table 5: Parametric performance comparison of 4 point – FFTs' combinations

Parameters	ECC	Parity SOS	Parity SOS ECC
Area	217191	24988	6686
Delay (ps)	1	0	38
Power (nw)	$0.2475*10^9$	$0.329*10^9$	$0.1131*10^9$

Wherein the parameters surmised in Table 5, Table 6 and Table 7 are 1. Area, 2. Power and 3. Delay.

Table 6: Parametric performance comparison of 8 point – FFTs' combinations

Parameters	ECC	Parity SOS	Parity SOS ECC
Area	73038	8941	65540
Delay (ps)	1	437	1771.92
Power (nw)	$0.7125*10^9$	$0.1368*10^9$	$0.3090*10^9$

Table 7: Parametric performance comparison of 16 point – FFTs' combinations

Parameters	ECC	Parity SOS	Parity SOS ECC
Area	227283	223907	362538
Delay (ps)	0	0	0
Power (nW)	$1.23*10^9$	$1.03*10^9$	$1.04*10^9$

CONCLUSION

In this paper, the architecture of fault tolerant parallel FFTs /IFFTs are studied and a modified architecture is proposed. The proposal is efficient as indicated from the comparison and tabulated results conclude that this is effective. And the intended objective of detecting and correcting the Soft errors has been achieved with the extensive use of these three techniques

(a) Hamming code in ECC

(b) Parity-SOS and

(c) A combination of both (a) & (b)

Complexity of implementation is found to be reduced in 4, 8 and 16 point FFTs & IFFTs. The potential of MATLAB and XILINX has also helped in running the iterations fast and optimize the code. The EDA tool Cadence, helped quantify the parameters of area, power consumption and process speed in 4, 8 and16-point FFTs.

Also the effectiveness of the proposal is acknowledged by the reduction of number of slices e.g. 286 vs. 9890, thus an effective reduction of 97% in ECC of 4-point FFTs.

APPLICATIONS

This study can be applied

- in the field of-wireless systems & broadcasting through Multiple Input Multiple Output – Orthogonal Frequency Division Multiplexing (MIMO-OFDM) [12], for example-
  1. long-term evolution mobile systems [13] and also on WiMax [14]
  2. in digital TV broadcasting (DVB); digital radio/audio broadcasting (DAB)
- for receiver (FFT) and transmitter(IFFT) of OFDM
- Spectrum analysis
- Noise reduction in mobile telephony

Speech & audio signals.

REFERENCES

[1] Digital Signal Processing Principles, Algorithms, and Applications by John G. Proakis, Dimitris G. Manolakis

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

[3] 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.

[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] Design of Hamming Code Encoding and Decoding Circuit Using Transmission Gate Logic: International Research Journal of Engineering and Technology (IRJET)

[6] DIGITAL SIGNAL PROCESSING by S Salivahanan, A Valavaraj

[7] DIGITAL SIGNAL PROCESSING by Ramesh Babu

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

[9] Dawid, Herbert, and Heinrich Meyr. "CORDIC algorithms and architectures." Digital Signal Processing for Multimedia Systems (1999): 623-655.

[10] Fault Tolerant Parallel FFTs Using Error Correction Codes and Parseval Checks, IEEE Transactions On Very Large Scale Integration (VLSI) Systems, Vol. 24, No. 2, Zhen Gao, Pedro Reviriego, Zhan Xu, Xin Su, Ming Zhao, Jing Wang, and Juan Antonio Maestro, February 2016

[11] 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.

[12] A. Sibille, C. Oestges, and A. Zanella, MIMO: From Theory to Implementation. San Francisco, CA, USA: Academic, 2010.

[13] 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.

[14] M. Ergen, Mobile Broadband—Including WiMax and LTE. New York, NY, USA: Springer-Verlag, 2009.