

Design and Implementation of Fault Tolerant Parallel FFTs using Error Correction Codes

G.Divya¹ and P.Nagarajan²

¹PG Student, Department of ECE, Vivekanandha College of Engineering for Women, India. Email: divya30ece@gmail.com

²Assistant Professor, Department of ECE, Vivekanandha College of Engineering for Women, India. Email: greennagarajan@gmail.com

Article Received: 17 February 2017

Article Accepted: 27 February 2017

Article Published: 01 March 2017

ABSTRACT

In recent electronic circuits pose a reliability threat, a lot of applications are necessary to use protection against soft errors, there is no exceptions in the field of communications and signal processing systems. For several applications is to use Algorithmic Based Fault Tolerant techniques (ABFT) that try to expand the algorithmic properties to detect and correct the errors. The ABFT technique is well suitable for communications and signal processing applications. One of the examples is Fast Fourier Transforms that are a input building block in various systems. Some protection schemes are proposed. Among this, probably the use of the Parseval or sum of squares check is most broadly known. In modern communication systems are impressively use several blocks that blocks are operated in parallel. This type of technique is well suitable for FFTs protection. Consequently two enhanced protection schemes that merge the use of error correction codes and Parseval checks are proposed. The proposed technique to reduce the implementation cost of protection.

Keywords: Error correction codes (ECCs), Fast fourier transforms (FFTs) and Soft errors.

1. INTRODUCTION

The difficulty of communications and signal processing circuits increases every year. This is made possible by the CMOS technology scaling that enables the addition of more and more transistors on a single device. This increased complexity makes the circuits more susceptible to errors. At the same time, the scaling means that transistors operate with lower voltages and are more vulnerable to errors caused by noise and manufacturing variations. The meaning of radiation-induced soft errors also increases as technology scales. Soft errors can change the logical value of a circuit node creating a short term 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. These contain 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 probable to add redundancy at the system level to detect and correct errors. One typical example is the use of Triple Modular Redundancy (TMR) that triples a block and votes between the three outputs to detect and correct errors. For example TMR, the overhead is >200%. This is because the insecure module is virtual three times (which requires a 200% overhead versus the unprotected module), and additionally, voters are required to correct the errors making the overhead >200%. Another approach is to try to utilize the algorithmic properties of the circuit to detect/correct errors. This is typically referred to as Algorithm-Based Fault Tolerance (ABFT), this policy can decrease the overhead necessary to protect a circuit. Signal processing and communications circuits are well suited for ABFT as they contain normal structures and many algorithmic properties. Over the years, many ABFT techniques have been planned to protect the basic blocks that are commonly used in those circuits. Some works have

considered the protection of digital filters. For example, the use of duplication using intense precision copies of the filter has been proposed as an option to TMR but with a lesser rate. In this brief, the security of parallel FFTs is considered. In particular, it is assumed that there can only be a single error on the method at any given point in time. This is a general statement while considering the protection against radiation-induced soft errors. There are three main contributions in this brief.

- 1) The estimation of the ECC method for the protection of equivalent FFTs showing its efficiency in terms of overhead and protection effectiveness.
- 2) The request of a new method based on the use of Parseval or sum of squares (SOSs) checks join with a parity FFT.
- 3) The proposal of a new method on which the ECC is used on the SOS checks as an alternative of on the FFTs.

2. EXISTING METHOD

The protection of filters has also been generally studied, occurrence of parallel filters to creates an opportunity to implement ABFT techniques for the whole collection of parallel modules as an alternative for each one independently. This has been studied for digital filters originally in where two filters were considered. In recent times, a broad scheme based on the use of error correction codes (ECCs) has been proposed. The design of filter implementation is complex. This will be denoted as $c1$ check. The same analysis applies to the other two redundant modules that will provide checks $c2$ and $c3$. Based on the differences observed on each of the checks, the module on which the error has occurred can be determined. The dissimilar patterns and the corresponding errors are summarized in Table 1. Once the module in error is known, the error can be corrected by reconstructing its output

using the residual modules. Related correction equations can be used to correct errors on the other modules. More advanced ECCs can be used to correct errors on many modules if that is required in a given application. The overhead of this technique, as discussed in, is lower than TMR as the number of redundant FFTs is related to the algorithm.

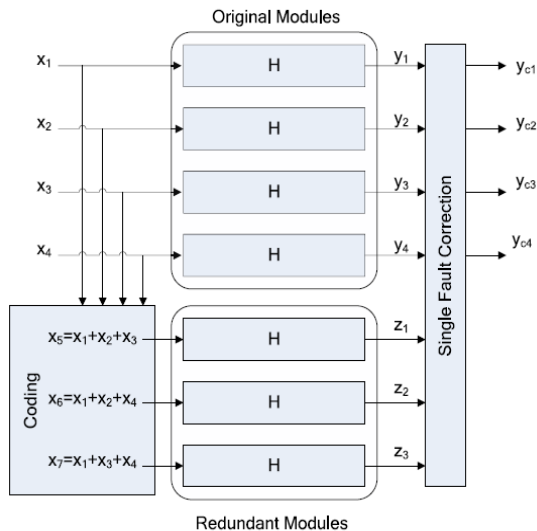


Fig.1. Parallel Filter Protection (Hamming Code)

To protect four FFTs, three redundant FFTs are required, but to protect eleven, the number of redundant FFTs is only four. This shows how the overhead reduces 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 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 filter can be used to correct the error. This is better explained with an example. The first proposed scheme is illustrated for the case of four parallel filters. A redundant (the parity) filter is added that has the sum of the inputs to the original filter as input.

The two proposed techniques offer new alternatives to protect parallel filters that can be more capable than caring each of the filters independently. The proposed schemes have been evaluated using FPGA implementations to evaluate the protection overhead. The results explain that by combining the use of ECCs and Parseval checks, the safety overhead can be reduced compared with the use of just ECCs as proposed.

3. PROPOSED PROTECTION SCHEMES

The initial point for our work is the protection scheme based on the use of ECCs that was presented in for digital filters. This scheme is shown in Fig. 2. In this example, a simple single error correction Hamming code is used. The new scheme 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,

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

Given that the DFT is a linear operation, its output z_5 can be used to verify that,

$$z_5 = z_1 + z_2 + z_3 \quad (2)$$

This shows how the overhead reduced with the number of FFTs. In this section, it has been mentioned that more than the years, a lot of techniques have been planned to protect the FFT. One of them is the Sum of Squares (SOSs) check that can be used to spot errors. The SOS check is based on the Parseval theorem that states that the SOSs of the inputs to the FFT are identical to the SOSs of the outputs of the FFT not including for a scaling aspect. This relationship can be used to detect errors through low overhead as one multiplication is needed for each input or output model (two multiplications and adders for SOS per sample).

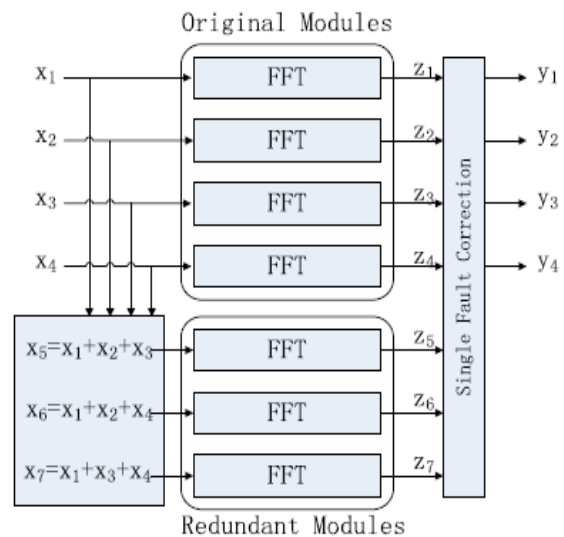


Fig.2. Parallel FFT protection using ECCs

The overhead of this technique, as discussed in, is poorer than TMR as the number of redundant FFTs is linked to the algorithm of the number of original FFTs. For example, to keep four FFTs, three redundant FFTs are needed, but to protect eleven, the number of unneeded FFTs is only four.

Parallel FFTs, the SOS check can be combined with the ECC approach to decrease the protection overhead. Since the SOS check can just detect errors, the ECC part should be able to apply the correction. This can be completed using the

equivalent of a simple parity bit for all the FFTs. In addition, the SOS check is used on every FFT to notice errors. When an error is detected, the output of the parity FFT can be used to exact the error. This is better explained with an example. In Fig. 2, the first proposed scheme is illustrated for the instance of four parallel FFTs.

This combination of a parity FFT and the SOS verify 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 corresponding-SOS. Another possibility to combine the SOS check and the ECC approach is instead of using an SOS check per FFT, make use of an ECC for the SOS checks.

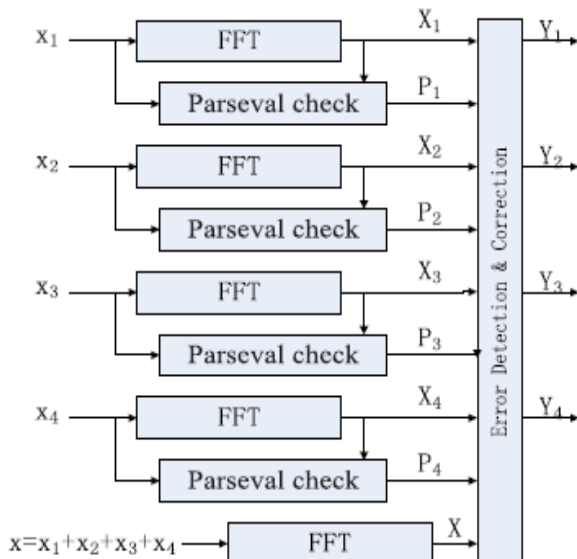


Fig.3. Parity-SOS (first technique) fault-tolerant parallel FFTs

A redundant FFT is added to facilitate has the sum of the inputs to the original FFTs as input. The 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 completed by re-computing the FFT in error using the output of the parity FFT (X) and the rest of the FFT outputs.

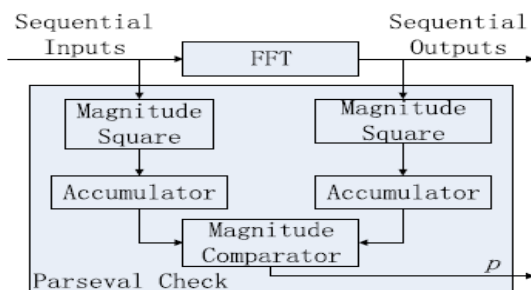


Fig.4. Implementation of the SOS check

$$X_{1c} = X - X_2 - X_3 - X_4 \quad (3)$$

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 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 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.

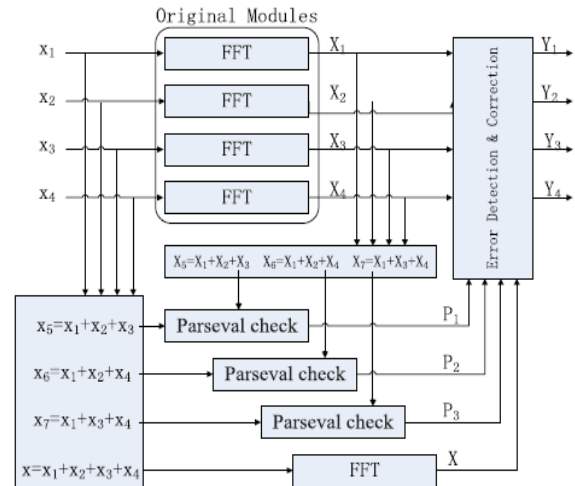


Fig.5. Parity-SOS-ECC fault-tolerant parallel FFTs

Table.1. Resources Usage for a Single FFT and SOS Check

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

Tables 1 explain the results when different number of parallel FFTs is protected. The point is to illustrate how the relative overheads of the different techniques vary with the number of parallel FFTs. In parentheses, the cost virtual to an unprotected implementation is also provided. The results show that all techniques have a rate factor of <2 . This demonstrates that the ECC-based technique is also competitive to keep 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.

The FFT and the various protection techniques have been implemented using Verilog. The results obtained are first table provides the resources needed to implement a single FFT and an SOS check. The results show that the FFT is more complex than the SOS check as expected. The difference will be much larger when a totally parallel FFT implementation is used. The parity-SOS-ECC scheme, the number of SOS checks also grows logarithmically and they are simpler to implement than FFTs. Therefore, it remains 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. It can be observed that eight is the value for which parity-SOS and ECC have almost the same cost. In each simulation run, one error is inserted to mimic the behavior of

soft errors that occur in isolation. For ECC protected parallel FFTs, a tolerance level of 1 is used for the equation checks. For the parity-SOS and parity-SOS-ECC schemes, the fault coverage is determined by the tolerance level τ used in the Parseval check.

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.

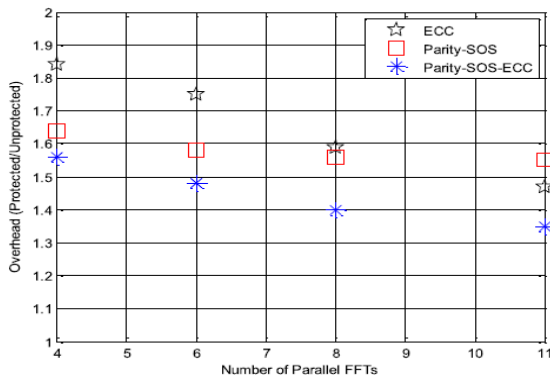


Fig.6. Overhead Comparison for the Number of Slices

Simulation Result for Parallel FFTs (Input)

		1000000																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																													
name	value	fft1000000	fft2000000	fft3000000	fft4000000	fft5000000	fft6000000	fft7000000	fft8000000	fft9000000	fft10000000	fft11000000	fft12000000	fft13000000	fft14000000	fft15000000	fft16000000	fft17000000	fft18000000	fft19000000	fft20000000	fft21000000	fft22000000	fft23000000	fft24000000	fft25000000	fft26000000	fft27000000	fft28000000	fft29000000	fft30000000	fft31000000	fft32000000	fft33000000	fft34000000	fft35000000	fft36000000	fft37000000	fft38000000	fft39000000	fft40000000	fft41000000	fft42000000	fft43000000	fft44000000	fft45000000	fft46000000	fft47000000	fft48000000	fft49000000	fft50000000	fft51000000	fft52000000	fft53000000	fft54000000	fft55000000	fft56000000	fft57000000	fft58000000	fft59000000	fft60000000	fft61000000	fft62000000	fft63000000	fft64000000	fft65000000	fft66000000	fft67000000	fft68000000	fft69000000	fft70000000	fft71000000	fft72000000	fft73000000	fft74000000	fft75000000	fft76000000	fft77000000	fft78000000	fft79000000	fft80000000	fft81000000	fft82000000	fft83000000	fft84000000	fft85000000	fft86000000	fft87000000	fft88000000	fft89000000	fft90000000	fft91000000	fft92000000	fft93000000	fft94000000	fft95000000	fft96000000	fft97000000	fft98000000	fft99000000	fft100000000	fft101000000	fft102000000	fft103000000	fft104000000	fft105000000	fft106000000	fft107000000	fft108000000	fft109000000	fft110000000	fft111000000	fft112000000	fft113000000	fft114000000	fft115000000	fft116000000	fft117000000	fft118000000	fft119000000	fft120000000	fft121000000	fft122000000	fft123000000	fft124000000	fft125000000	fft126000000	fft127000000	fft128000000	fft129000000	fft130000000	fft131000000	fft132000000	fft133000000	fft134000000	fft135000000	fft136000000	fft137000000	fft138000000	fft139000000	fft140000000	fft141000000	fft142000000	fft143000000	fft144000000	fft145000000	fft146000000	fft147000000	fft148000000	fft149000000	fft150000000	fft151000000	fft152000000	fft153000000	fft154000000	fft155000000	fft156000000	fft157000000	fft158000000	fft159000000	fft160000000	fft161000000	fft162000000	fft163000000	fft164000000	fft165000000	fft166000000	fft167000000	fft168000000	fft169000000	fft170000000	fft171000000	fft172000000	fft173000000	fft174000000	fft175000000	fft176000000	fft177000000	fft178000000	fft179000000	fft180000000	fft181000000	fft182000000	fft183000000	fft184000000	fft185000000	fft186000000	fft187000000	fft188000000	fft189000000	fft190000000	fft191000000	fft192000000	fft193000000	fft194000000	fft195000000	fft196000000	fft197000000	fft198000000	fft199000000	fft200000000	fft201000000	fft202000000	fft203000000	fft204000000	fft205000000	fft206000000	fft207000000	fft208000000	fft209000000	fft210000000	fft211000000	fft212000000	fft213000000	fft214000000	fft215000000	fft216000000	fft217000000	fft218000000	fft219000000	fft220000000	fft221000000	fft222000000	fft223000000	fft224000000	fft225000000	fft226000000	fft227000000	fft228000000	fft229000000	fft230000000	fft231000000	fft232000000	fft233000000	fft234000000	fft235000000	fft236000000	fft237000000	fft238000000	fft239000000	fft240000000	fft241000000	fft242000000	fft243000000	fft244000000	fft245000000	fft246000000	fft247000000	fft248000000	fft249000000	fft250000000	fft251000000	fft252000000	fft253000000	fft254000000	fft255000000	fft256000000	fft257000000	fft258000000	fft259000000	fft260000000	fft261000000	fft262000000	fft263000000	fft264000000	fft265000000	fft266000000	fft267000000	fft268000000	fft269000000	fft270000000	fft271000000	fft272000000	fft273000000	fft274000000	fft275000000	fft276000000	fft277000000	fft278000000	fft279000000	fft280000000	fft281000000	fft282000000	fft283000000	fft284000000	fft285000000	fft286000000	fft287000000	fft288000000	fft289000000	fft290000000	fft291000000	fft292000000	fft293000000	fft294000000	fft295000000	fft296000000	fft297000000	fft298000000	fft299000000	fft300000000	fft301000000	fft302000000	fft303000000	fft304000000	fft305000000	fft306000000	fft307000000	fft308000000	fft309000000	fft310000000	fft311000000	fft312000000	fft313000000	fft314000000	fft315000000	fft316000000	fft317000000	fft318000000	fft319000000	fft320000000	fft321000000	fft322000000	fft323000000	fft324000000	fft325000000	fft326000000	fft327000000	fft328000000	fft329000000	fft330000000	fft331000000	fft332000000	fft333000000	fft334000000	fft335000000	fft336000000	fft337000000	fft338000000	fft339000000	fft340000000	fft341000000	fft342000000	fft343000000	fft344000000	fft345000000	fft346000000	fft347000000	fft348000000	fft349000000	fft350000000	fft351000000	fft352000000	fft353000000	fft354000000	fft355000000	fft356000000	fft357000000	fft358000000	fft359000000	fft360000000	fft361000000	fft362000000	fft363000000	fft364000000	fft365000000	fft366000000	fft367000000	fft368000000	fft369000000	fft370000000	fft371000000	fft372000000	fft373000000	fft374000000	fft375000000	fft376000000	fft377000000	fft378000000	fft379000000	fft380000000	fft381000000	fft382000000	fft383000000	fft384000000	fft385000000	fft386000000	fft387000000	fft388000000	fft389000000	fft390000000	fft391000000	fft392000000	fft393000000	fft394000000	fft395000000	fft396000000	fft397000000	fft398000000	fft399000000	fft400000000	fft401000000	fft402000000	fft403000000	fft404000000	fft405000000	fft406000000	fft407000000	fft408000000	fft409000000	fft410000000	fft411000000	fft412000000	fft413000000	fft414000000	fft415000000	fft416000000	fft417000000	fft418000000	fft419000000	fft420000000	fft421000000	fft422000000	fft423000000	fft424000000	fft425000000	fft426000000	fft427000000	fft428000000	fft429000000	fft430000000	fft431000000	fft432000000	fft433000000	fft434000000	fft435000000	fft436000000	fft437000000	fft438000000	fft439000000	fft440000000	fft441000000	fft442000000	fft443000000	fft444000000	fft445000000	fft446000000	fft447000000	fft448000000	fft449000000	fft450000000	fft451000000	fft452000000	fft453000000	fft454000000	fft455000000	fft456000000	fft457000000	fft458000000	fft459000000	fft460000000	fft461000000	fft462000000	fft463000000	fft464000000	fft465000000	fft466000000	fft467000000	fft468000000	fft469000000	fft470000000	fft471000000	fft472000000	fft473000000	fft474000000	fft475000000	fft476000000	fft477000000	fft478000000	fft479000000	fft480000000	fft481000000	fft482000000	fft483000000	fft484000000	fft485000000	fft486000000	fft487000000	fft488000000	fft489000000	fft490000000	fft491000000	fft492000000	fft493000000	fft494000000	fft495000000	fft496000000	fft497000000	fft498000000	fft499000000	fft500000000	fft501000000	fft502000000	fft503000000	fft504000000	fft505000000	fft506000000	fft507000000	fft508000000	fft509000000	fft510000000	fft511000000	fft512000000	fft513000000	fft514000000	fft515000000	fft516000000	fft517000000	fft518000000	fft519000000	fft520000000	fft521000000	fft522000000	fft523000000	fft524000000	fft525000000	fft526000000	fft527000000	fft528000000	fft529000000	fft530000000	fft531000000	fft532000000	fft533000000	fft534000000	fft535000000	fft536000000	fft537000000	fft538000000	fft539000000	fft540000000	fft541000000	fft542000000	fft543000000	fft544000000	fft545000000	fft546000000	fft547000000	fft548000000	fft549000000	fft550000000	fft551000000	fft552000000	fft553000000	fft554000000	fft555000000	fft556000000	fft557000000	fft558000000	fft559000000	fft560000000	fft561000000	fft562000000	fft563000000	fft564000000	fft565000000	fft566000000	fft567000000	fft568000000	fft569000000	fft570000000	fft571000000	fft572000000	fft573000000	fft574000000	fft575000000	fft576000000	fft577000000	fft578000000	fft579000000	fft580000000	fft581000000	fft582000000	fft583000000	fft584000000	fft585000000	fft586000000	fft587000000	fft588000000	fft589000000	fft590000000	fft591000000	fft592000000	fft593000000	fft594000000	fft595000000	fft596000000	fft597000000	fft598000000	fft599000000	fft600000000	fft601000000	fft602000000	fft603000000	fft604000000	fft605000000	fft606000000	fft607000000	fft608000000	fft609000000	fft610000000	fft611000000	fft612000000	fft613000000	fft614000000	fft615000000	fft616000000	fft617000000	fft618000000	fft619000000	fft620000000	fft621000000	fft622000000	fft623000000	fft624000000	fft625000000	fft626000000	fft627000000	fft628000000	fft629000000	fft630000000	fft631000000	fft632000000	fft633000000	fft634000000	fft635000000	fft636000000	fft637000000	fft638000000	fft639000000	fft640000000	fft641000000	fft642000000	fft643000000	fft644000000	fft645000000	fft646000000	fft647000000	fft648000000	fft649000000	fft650000000	fft651000000	fft652000000	fft653000000	fft654000000	fft655000000	fft656000000	fft657000000	fft658000000	fft659000000	fft660000000	fft661000000	fft662000000	fft663000000	fft664000000	fft665000000	fft666000000	fft667000000	fft668000000	fft669000000	fft670000000	fft671000000	fft672000000	fft673000000	fft674000000	fft675000000	fft676000000	fft677000000	fft678000000	fft679000000	fft680000000	fft681000000	fft682000000	fft683000000	fft684000000	fft685000000	fft686000000	fft687000000	fft688000000	fft689000000	fft690000000	fft691000000	fft692000000	fft693000000	fft694000000	fft695000000	fft696000000	fft697000000	fft698000000	fft699000000	fft700000000	fft701000000	fft702000000	fft703000000	fft704000000	fft705000000	fft706000000	fft707000000	fft708000000	fft709000000	fft710000000	fft711000000	fft712000000	fft713000000	fft714000000	fft715000000	fft716000000	fft717000000	fft718000000	fft719000000	fft720000000	fft721000000	fft722000000	fft723000000	fft724000000	fft725000000	fft726000000	fft727000000	fft728000000	fft729000000	fft730000000	fft731000000	fft732000000	fft733000000	fft734000000	fft735000000	fft736000000	fft737000000	fft738000000	fft739000000	fft740000000	fft741000000	fft742000000	fft743000000	fft744000000	fft745000000	fft746000000	fft747000000	fft748000000	fft749000000	fft750000000	fft751000000	fft752000000	fft753000000	fft754000000	fft755000000	fft756000000	fft757000000	fft758000000	fft759000000	fft760000000	fft761000000	fft762000000	fft763000000	fft764000000	fft765000000	fft766000000	fft767000000	fft768000000	fft769000000	fft770000000	fft771000000	fft772000000	fft773000000	fft774000000	fft775000000	fft776000000	fft777000000	fft778000000	fft779000000	fft780000000	fft781000000	fft782000000	fft783000000	fft784000000	fft785000000	fft786000000	fft787000000	fft788000000	fft789000000	fft790000000	fft791000000	fft792000000	fft793000000	fft794000000	fft795000000	fft796000000	fft797000000	fft798000000	fft799000000	fft800000000	fft801000000	fft802000000	fft803000000	fft804000000	fft805000000	fft806000000	fft807000000	fft808000000	fft809000000	fft810000000	fft811000000	fft812000000	fft813000000	fft814000000	fft815000000	fft816000000	fft817000000	fft818000000	fft819000000	fft820000000	fft821000000	fft822000000	fft823000000	fft824000000	fft825000000	fft826000000	fft827000000	fft828000000	fft829000000	fft830000000

parity-SOS-ECC scheme is ~99.9% and then tolerance level for SOS check is 1.

REFERENCES

- [1] Gao Z., et al., "Fault tolerant parallel filters based on error correction codes" (2015), *IEEE Trans. on Very Large-Scale Integr. (VLSI) Syst.*, Vol.23, No.2, pp.384–387.
- [2] Baumann R., "Soft errors in advanced computer systems" (2005), *IEEE Des. Test Comput.*, Vol.22, No.3, pp.258–266.
- [3] Gao Z., Yang W., Chen X., Zhao M., and Wang J., "Fault missing rate analysis of the arithmetic residue codes based fault tolerant FIR filter design" (2012), in *Proc. IEEE IOLTS*, pp.130–133.
- [4] Hitana T., and Deb A. K., "Bridging concurrent and non-concurrent error detection in FIR filters" (2004), in *Proc. Norchip Conf.*, pp. 75–78.
- [5] Jou J. Y., and Abraham J. A., "Fault-tolerant FFT networks" (1988), *IEEE Trans. Comput.*, Vol. 37, No. 5, pp.548–561.
- [6] Kim E. P., and Shanbhag N. R., "Soft N-modular redundancy" (2012), *IEEE Trans. Comput.*, Vol.61, No.3, pp. 323–336.
- [7] Nicolaidis M., "Design for soft error mitigation" (2005), *IEEE Trans. Device Mater. Rel.*, Vol. 5, No.3, pp. 405–418.
- [8] Pontarelli S., Cardarilli G. C., Re M., and Salsano A., "Totally fault tolerant RNS based FIR filters" (2008), in *Proc. 14th IEEE Int. On-Line Test Symp. (IOLTS)*, pp.192–194.
- [9] Reddy.A.L.N., and Banerjee.P., "Algorithm-based fault detection for signal processing applications" (1990), *IEEE Trans. Comput.*, Vol. 39, No. 10, pp. 1304–1308.
- [10] Reviriego P., Bleakley C. J., and Maestro J. A., "A novel concurrent error detection technique for the fast Fourier transform" (2012), in *Proc. ISSC, Maynooth, Ireland*, pp.1–5.