FPGA Based Parallel Filters Error Correction Codes Using Hamming Code

RAJU KATRU[1], Dr.M.CHANDRASEKHER[2]
[1] *Ph D scholar in Rayalaseema University, Kurnool, Andhra Pradesh, India*
[2]*Manager(D&E),Bharath Dynamite Ltd,Khanchnbagh,Ministry of Defence,Hyderabad,India*

Abstract— Digital filters are generally used in signal processing and communication systems. In a few cases, the consistency of those systems is important, and error tolerant filter implementations are necessary. Over the years, several techniques that develop the filters' structure and properties to achieve fault tolerance have been proposed. When technology scale, it enables more complex systems that integrate several filters. In those difficult systems, it is general that a few of the filter make active in parallel, for example, by apply the same filter to different input signal. in recent times, an easy method that exploits the occurrence of parallel filters to achieve fault tolerance has been presented. In this brief, that idea is general to show that parallel filters can be protected using fault correction codes in which each filter is the equivalent of a bit in a usual ECC. This new scheme allows more professional protection when the number of parallel filters is large. The technique is evaluated using a case study of parallel finite impulse response filters showing the effectiveness in terms of safety and implementation cost.

Keywords— Error correction codes (ECCs), filters, soft errors

I. INTRODUCTION

Electronic circuits are increasingly present in and space applications where dependability is critical. In those applications, the circuit have to give several degree of fault tolerance. This requires is additional increased by the intrinsic reliability challenges of advanced CMOS technologies that include, e.g., manufacturing variations and soft errors. A number of techniques can be used to save from harm a circuit from errors. Those range from modifications in the manufacturing process of the circuits to decrease the number of errors to adding redundancy at the logic or system level to ensure that errors do not affect the system functionality [1]. To add redundancy, a general technique known as triple modular redundancy can be used. The TMR, which triplicates the design and add voting logic to right errors, is frequently used. However, it more than triples the area and power of the circuit somewhat that may not be acceptable in some application. When the circuit to be protected has algorithmic or structural properties, a improved option can be to exploit those properties to design error tolerance. One example is signal processing circuits used for which exact techniques have been designed over the many years [2].

Digital filters are one of the main commonly used signal processing circuits and more techniques have been planned to defend them from fault. Most of them have determined on finite-impulse response filters. For example, in [3], the use of reduced accuracy replicas was proposed to reduce the cost of implementing modular redundancy in FIR filters. In [4], a connection between the memory elements of an FIR filter and the input sequence was used to detect errors. Other schemes have exploited the FIR properties at a word level to also attain error tolerance [5]. The use of remains number systems [6] and arithmetic codes [7] has also been planned to protect filters. Finally, the use of different implementation structures of the FIR filters to correct errors with only one redundant module has also been planned [8]. In all the techniques mentioned so far, the protection of a single filter is measured.

However, it is increasingly general to find systems in which several filters operate in parallel. This is the case in filter banks [9] and in several modern communication systems [10]. For those systems, the security of the filters can be addressed at a higher level by considering the parallel filters as the block to be protected. This thought was explored in [11], where double parallel filters with the similar response that processed different input signals were measured. It was shown that with single one redundant copy, single error correction can be implemented. Thus, a important cost reduction compared with TMR was designed.

In this brief, a common scheme to protect parallel filters is obtainable. As in [11], parallel filters with the similar response that process different input signals are considered. The new approach is based on the application of error correction codes using each of the filter outputs as the corresponding of a bit in and ECC secret code. This is a generality of the plan presented in [11] and enables more efficient implementations when the number of parallel filters is bulky. The scheme can also be used to offer more powerful security using advanced ECCs that can correct failures in multiples modules. The rest of this

brief introduce the latest scheme by first summarizing the parallel filters considered in Section II. Then, in Section III, the planned scheme is accessible. Section IV presents a case study to illustrate the efficiency of the approach. Finally, the conclusions are summarized in Section V

II.PARALALL FILTERS WITH THE SAME RESPONSE

A discrete time filter design the followed equation

$$y(n)= \sum_{l=0}^{\infty} x(n-l).h(l) \quad \text{----------(1)}$$

where $x[n]$ is the input data, $y[n]$ is the output, and $h[l]$ is the impulse response of the filter [12]. When the reaction $h[l]$ is nonzero, only for a limited number of samples, the filter is known as a FIR filter, otherwise the filter is an infinite impulse response (IIR) filter. There are different structures to implement both FIR and IIR filters.

In the subsequent, a set of k parallel filters with the equal response and dissimilar input signals are measured. These parallel filters are illustrated in Fig. 1. This type of filter is create in some communication systems that use several channels in parallel. In data acquirement and processing application is also common to filter different signals with the similar response.

An attractive property for these parallel filters is that the addition of any grouping of the outputs $y_i[n]$ can also be obtained by adding the corresponding inputs $x_i[n]$ and filtering the resulting signal with

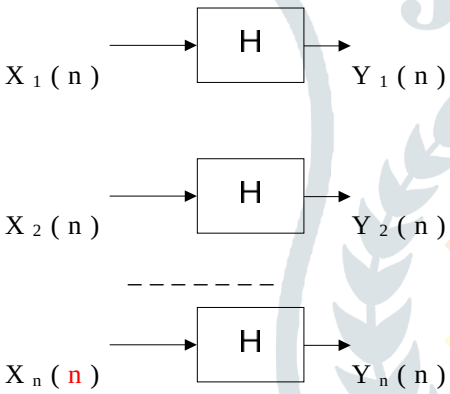

Fig. 1. Parallel filters response for the same.

the same filter $h[l]$. For example

$$y1[n]+y2[n]= \sum_{l=0}^{\infty} x_1(n-l)+X_2(n-l) \quad -(2)$$

This easy observation will be used in the following to expand the proposed fault tolerant implementation.

III.PROPOSED SCHEME

The new method is based on the make use of of the ECCs. A easy ECC takes a block of k bits and generate a block of n bits by adding n k parity check bits [13]. The parity check bits are XOR combination of the k information bits. By correctly manipulative those combinations it is possible to sense and correct errors. One of the model, let us consider a simple Hamming code [14] with $k = 4$ and $n = 7$. In this case, the three parity check bits $p1$, $p2$, $p3$ are computed as a function of the data bits $d1$, $d2$, $d3$, $d4$ as follows:

$$p1 = d_1 \oplus d_2 \oplus d_3$$

$$p2 = d_1 \oplus d_2 \oplus d_4$$

$$p3 = d_1 \oplus d_3 \oplus d_4 \quad \text{--------(3)}$$

The data and parity check bits are store and can be recovered afterwards even if there is an error in one of the bits. This is completed by recomputing the parity check bits and verifying the results with the values stored. In the model considered, an error on $d1$ will reason errors on the three parity checks; an error on $d2$ only in $p1$ and $p2$; an error on $d3$ in $p1$ and $p3$; and finally an error on $d4$ in $p2$ and $p3$. Thus, the data bit in error can be positioned and the fault can be corrected. This is usually formulated in conditions of the generate G and parity check H matrixes. For the Hamming code measured in the model, those are

$$G = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \qquad H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Encoding is done by computing $\bar{g} \cdot y = x.G$ and error detection is done by computing $s = y.H^T$, here the operator is based on module two addition (XOR) and multiplication. Correction is completed using the s, known as syndrome, to recognize the bit in error. The messages of values of s to error position is captured in Table I

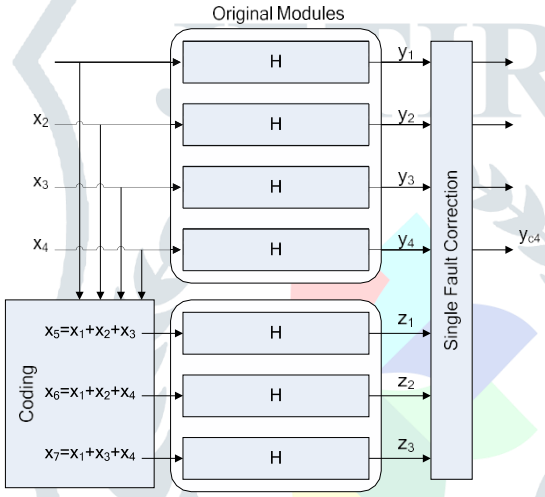

Fig. 2. Proposed method for 4 filters and a Hamming code.

y_{c3}

TABLE I
ERROR LOCATION IN THE HAMMING CODE

s_1 s_2 s_3	Error Bit Position	Action
0 0 0	No Error	None
1 1 1	d1	Correct d1
1 1 0	d2	Correct d2
1 0 1	d3	Correct d3
0 1 1	d4	Correct d4
1 0 0	P1	Correct p1
0 1 0	P2	Correct p2
0 0 1	P3	Correct p3

Once the erroneous bit is identified, it is corrected by simply inverting the bit.

This error correction code scheme can be applied to the parallel filters consider by defining a set of check filters z_j. For the case of four filters $y1, y2, y3, y4$ and the Hamming code, the test out filters would be

$Z_1(n) =$

$$\sum_{l=0}^{\infty} (x_1[n-l]+x_2[n-l]+x_3[n-l]$$

$Z_2(n)=$

$\sum_{l=0}^{\infty}$ $(x_1[n-l]+x_2[n-l]+x_4[n-l]$

$Z_3(n)=$

$\sum_{l=0}^{\infty}$ $(x_1[n-l]+x_3[n-l]+x_4[n-l]$ ------(6)

And the Check is done by testing if

$Z_1[n] = y_1[n]+y_2[n]+y_3[n]$

$Z_2[n] = y_1[n]+y_2[n]+y_4[n]$

$Z_3[n] = y_1[n]+y_3[n]+y_4[n]$ ----------(7)

For example, an error on filter $y1$ will reason faults on the checks of $z1$, $z2$, and $z3$. equally, errors on the other filters will cause errors on a different group of z_i . Therefore, as with the usual ECCs, the error can be located and corrected.

The overall method is illustrated on Fig. 2. It can be observed that modification is achieved with only three redundant filters.

For the filters, correction is achieved by reconstructing the incorrect outputs using the rest of the data and check outputs. For example, when an error on y1 is find, it can be corrected by making

$y_c1[n]=z1[n]-y2[n]-y3[n]$. *(8)*

Comparable equations can be used to correct errors on the rest of the data outputs.

In our case, we can describe the check matrix as

$$H=\begin{bmatrix} 1 & 1 & 1 & 0 & -1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & -1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & -1 \end{bmatrix}$$

and calculate $s=yH^T$ to detect errors. Then, the vector s is also used to find the filter in error. In our case, a nonzero value in vector s is equivalent to 1 in the usual Hamming code. A zero value in the check corresponds to a 0 in the usual Hamming code.

It is significant to note that due to different finite precision effects in the original and check filter implementations, the comparisons in (7) can show small differences. Those differences will depend on the quantization effects in the filter implementations that have been generally studied for dissimilar filter structures. The interested person who reads is referred to [12] for additional details. Therefore, a threshold must be used in the comparisons so that values lesser than the threshold are classified as 0. This means that small errors may not be corrected. This will not be an issue in most of the cases as small errors are acceptable. The complete study of the effect of these small errors on the signal to noise ratio at the output of the filter is left for future work. The person who reads can get more details on this type of testing in [3].

With this another formulation, it is clear that the method can be used for any number of parallel filters and any linear block code can be used. The approach is more good-looking when the number of filters k is large. For example, when k 11, only four redundant filters are essential to provide single error correction. This is the same as for usual ECCs for which the overhead decreases as the block size increases [13].

The extra operations required for encoding and decoding are simple additions, subtractions, and comparisons and should have small effect on the overall complexity of the circuit. This is illustrated in Section IV in which a case study is presented.

In the conversation, so far the effect of errors affecting the encoding and decoding logic has not be considered. The encoder and decoder include some additions and subtractions and therefore the opportunity of errors affecting them cannot be ignored. Focusing on the encoders, it can be seen that some of the calculations of the z_i split adders. For example, looking at (6), $z1$ and $z2$ share the term $y1 + y2$. Thus, an error in that adder could affect both $z1$ and $z2$ cause a miscorrection on $y2$. To ensure that single errors in

the encoding logic will not influence the data outputs, one alternative is to avoid logic sharing by computing each of the z_i independently. In that case, errors will only affect one of the z_i outputs and according to Table I, the data outputs y j will not be affected. Similarly, by avoiding logic sharing, single errors in the calculation of the s vector will only affect one of its bits. The final alteration elements such as that in (8) require to be tripled to ensure that they do not propagate errors to the outputs. However, as their difficulty is small compared with that of the filters, the impact on the generally circuit cost will be low. This is definite by the results presented in Section IV for a case study

TABLE II
RESOURCES USAGE 4 PARALLEL FIR FILTRRS

	Un Protected	TMR	Method in [7]	Proposed
Slices	2933	9020	7735	6401
Flip-Flops	1222	3980	3980	2939
LUT-4	5690	17250	13635	12030

TABLE III
RESOURCES USAGE FOR 11 PARALLEL FIR FILTRRS

	Un Protected	TMR	Method in [7]	Proposed
Slices	8090	24850	21280	14411
Flip-Flops	3333	10940	10940	6470
LUT-4	15650	47450	37500	28221

IV.CASE STUDY

To evaluate the efficiency of the projected scheme, a case study is used. A set of the parallel FIR filters with 16 coefficients are measured. The input data and coefficients are quantized with 8 bits. The filter output is quantized with 18 bits. For the check filters z_i , while the input is the addition of several inputs x_j , the input bit-width is extensive to 10 bits. A small threshold is used in the comparison such that errors lesser than the threshold are not measured errors. As explained in Section III, no logic contribution was used in the computations in the encoder and decoder logic to stay away from errors on them from propagating to the output.

Two configurations are measured. The first one is block of four parallel filters for which a Hamming code with $k = 4$ and $n = 7$ is used. The second is block of eleven parallel filters for which a Hamming code with k= 11 and n= 15 is used. Both configurations have been implemented in HDL and mapped to a Xilinx Virtex 4 XC4VLX80 device.

The first evaluation is to evaluate the resources used by the planned scheme with those used by TMR, the protection method proposed in [7] (with m 7) and by an unprotected filter implementation. Those results are presented in Tables II and III for each of the configurations considered. It can be observed that the planned technique provides important savings (from 26% to 41%) for all the resource types (slices, flip-flops, and LUTs) compared with the TMR. The benefits are big for the second configuration as expected with values exceeding 40% for all resource types. In that case, the relative numbers of additional verify filters $(n\ k)/n$ is smaller. When compared with the mathematics code technique planned in [7], the savings are smaller but still significant ranging from 11% to 40%. Again, larger savings are obtained for the second configuration.

In summary, the results of this case study confirm that the proposed scheme can reduce the implementation pf the cost significantly compared with the TMR and provides also reductions when compared with other model such as that in [7]. As discussed before, the reductions are larger than when the number of filters is large.

The second estimation is to assess the efficiency of the scheme to correct errors. To that conclusion, fault injection experiment has been conducted. In particular, errors have been randomly inserted in the coefficients and inputs of filters. In all cases, single errors were find out and corrected. In total, 7900 errors

for inputs and 7900 errors for filter coefficients were inserted in different simulation runs. This confirms the efficiency of the scheme to correct single errors.

This brief has accessible a new method to protect parallel filters that are frequently found in modern of signal processing circuits The approach based on applying ECCs to the parallel filters outputs detected and correct errors. The scheme would be used for parallel filters that have the same response and process different input signals.

A case study has also been discussed to show effectiveness of the scheme in terms of an error correction and also of circuit overheads. The technique will provides larger benefits when the number of parallel filters is large.

The proposed scheme also to be applied to the IIR filters. Future work will consider the estimation of the benefits of the planned technique for IIR filters. The addition of the scheme to parallel filters that have the similar input and different impulse responses is also a topic for future work. The proposed scheme can also be combined with the reduced precision replica scheme or approach is presented in [3] to reduce the overhead required for the protection. This will be of interest when the number of parallel filters is small as the cost of the proposed scheme is larger than in that case. Another interesting topic is to be continue in this brief is to explore the use of more powerful multibit ECCs, such as Bose–Chaudhuri–Hocquenghem codes, to correct errors on multiple filters.

REFERENCES

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

[2] A. Reddy and P. Banarjee "Algorithm-based fault detection for sig- nal processing applications," *IEEE Trans. Comput.*, vol. 39, no. 10, pp. 1304–1308, Oct. 1990.

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

[5] Y.-H. Huang, "High-efficiency soft-error-tolerant digital signal process- ing using fine-grain subword-detection processing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 2, pp. 291–304, Feb. 2010.

[6] S. Pontarelli, G. C. Cardarilli, M. Re, and A. Salsano, "Totally fault tolerant RNS based FIR filters," in *Proc. IEEE IOLTS*, Jul. 2008, pp. 192–194.

[7] Z. Gao, W. Yang, X. Chen, M. Zhao, and J. Wang, "Fault missing rate analysis of the arithmetic residue codes based fault-tolerant FIR filter design," in *Proc. IEEE IOLTS*, Jun. 2012, pp. 130–133.

[8] P. Reviriego, C. J. Bleakley, and J. A. Maestro, "Strutural DMR: A technique for implementation of soft-error-tolerant FIR filters," *IEEE Trans. Circuits Syst., Exp. Briefs*, vol. 58, no. 8, pp. 512–516, Aug. 2011.

[9] P. P. Vaie dyanathan. *Multirate Systems and Filter Banks*. Upper Saddle River, NJ, USA: Prentice-Hall, 1993.

[10] A. Sibille, C. Oestges, and A. Zanella, *MIMO: From Theory to Imple- mentation*. San Francisco, CA, USA: Academic Press, 2010.

[11] P. Reviriego, S. Pontarelli, C. 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.

[12] A. V. Oppenheim and R. W. Schafer, *Discrete Time Signal Processing*. Upper Saddle River, NJ, USA: Prentice-Hall 1999.

[13] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall. 2004.

[14] 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

[15] R. Bauman, "Soft faults, in advanced computer systems," *IEEE Des.Test Compute.*, vol. 22, no. 3, pp. 258–266, May/Jun. 2005.

Authors Profile

Mr.RAJUKATRU pursed Bachelor of Technology from University of Kakatiya University in 2003 and Master of Technology from JNTU Hyderabad University in year 2009. He is currently pursuing Ph.D in Rayalaseema University,Kurnool,Andhrapradesh and currently working as Assistant Professor in Department of Electronics and Communication Engineering in Ashoka Institute of Engineering & Technology. He has published more than 11 research papers in reputed international journals. He has 10 years of teaching experience and 5 years of Research Experience.

DrM.Chandrasekhar,BDL Manager,Ministry of defence,Hyderabad,Telangana. He has published more than 50 research papers in reputed international journals. He has 20 years of Industrial experience and 20 years of Research Experience.