International Journal of Research
Available at https://edupediapublications.org/journals
Special Issue on Conference Papers
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 06
March 2018

Adaptive Technic for implementing Fault Tolerant Parallel Filters

[1]*Madasu.Niranjan,* [2]*V.Niveditha*

Assistant professor, Dept. Electronic & Communication engineering
Mother Theressa College Of Engg & Tech. Peddapalli, , India
E-Mail: niru451@gmail.com[1] , niveditha.vijayagiri@gmail.com[2]

Abstract: Digital filters are widely used in signal processing and communication systems. In some cases, the reliability of those systems is critical, and fault tolerant filter implementations are needed. Over the years, many techniques that exploit the filters' structure and properties to achieve fault tolerance have been proposed. As technology scales, it enables more complex systems that incorporate many filters. In those complex systems, it is common that some of the filters operate in parallel, for example, by applying the same filter to different input signals. Recently, a simple technique that exploits the presence of parallel filters to achieve fault tolerance has been presented. In this brief, that idea is generalized to show that parallel filters can be protected using error correction codes (ECCs) in which each filter is the equivalent of a bit in a traditional ECC. This new scheme allows more efficient 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 protection and implementation cost.

Keywords: Parallel Filters, Coding, Soft Errors.

I. INTRODUCTION

Electronic circuits are increasingly present in automotive, medical, and space applications where reliability is critical. In those applications, the circuits have to provide some degree of fault tolerance. This need is further 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 protect a circuit from errors. Those range from modifications in the manufacturing process of the circuits to reduce the number of errors to adding redundancy at the logic or system level to ensure that errors do not affect the system functionality. To add redundancy, a general technique known as triple modular redundancy (TMR) can be used. The TMR, which triplicates the design and adds voting logic to correct errors, is commonly used. However, it more than triples the area and power of the circuit, something that may not be acceptable in some applications. When the circuit to be protected has algorithmic or structural properties, a better option can be to exploit those properties to implement fault tolerance. One example is signal processing circuits for which specific techniques have been proposed over the years. Digital filters are one of the most commonly used signal processing circuits and several techniques have been proposed to protect them from errors. Most of them have focused on finite-impulse response (FIR) filters. For example, in, the use of reduced precision replicas was proposed to reduce the cost of implementing modular redundancy in FIR filters.

In, a relationship 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 achieve fault tolerance. The use of residue number systems and arithmetic codes has also been proposed 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 proposed. In all the techniques mentioned so far, the protection of a single filter is considered. However, it is increasingly common to find systems in which several filters operate in parallel. This is the case in filter banks and in many modern communication systems. For those systems, the protection of the filters can be addressed at a higher level by considering the parallel filters as the block to be protected. This idea was explored in, where two parallel filters with the same response that processed different input signals were considered. It was shown that with only one redundant copy, single error correction can be implemented. Therefore, a significant cost reduction compared with TMR was obtained. In this brief, a general scheme to protect parallel filters is presented. As in, parallel filters with the same response that process different input signals are considered. The new approach is based on the application of error correction codes (ECCs) using each of the filter outputs as the equivalent of a bit in and ECC codeword. This is a generalization of the scheme presented and enables more efficient implementations when the number of parallel filters is large. The scheme can also be used to provide more powerful protection using advanced ECCs that can correct failures in multiples modules.

II. PARALLEL FILTERS WITH THE SAME RESPONSE A discrete time filter implements the following equation:

$$y[n] = \sum_{l=0}^{\infty} x[n-l].h[l]$$

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

In the following, a set of k parallel filters with the same response and different input signals are considered. These parallel filters are illustrated in Fig. 1. This kind of filter is found in some communication systems that use several channels in parallel. In data acquisition and processing applications is also common to filter several signals with the same response. An interesting property for these parallel filters is that the sum of any combination of the outputs yi [n] can also be obtained by adding the corresponding inputs xi [n] and filtering the resulting signal with

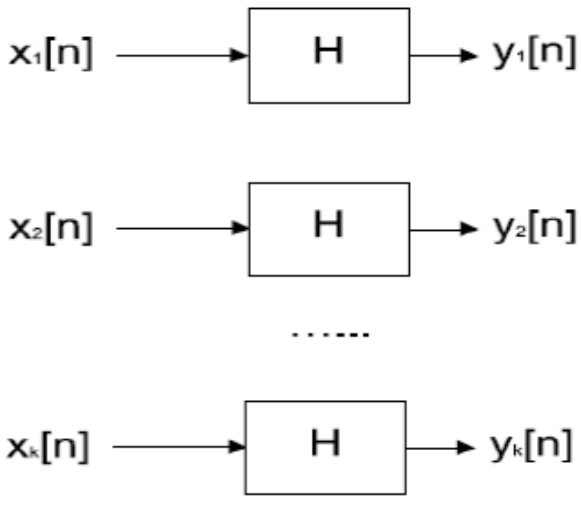

Fig.1. Parallel filters with the same response.

the same filter h[l]. For example

$$y_1[n] + y_2[n] = \sum_{l=0}^{\infty}(x_1[n-l] + x_2[n-l]).h[l] \quad (2)$$

This simple observation will be used in the following to develop the proposed fault tolerant implementation.

III. PROPOSED SCHEME The new technique is based on the use of the ECCs. A simple ECC takes a block of k bits and produces a block of n bits by adding n−k parity check bits. The parity check bits are XOR combinations of the k data bits. By properly designing those combinations it is possible to detect and correct errors. As an example, let us consider a simple Hamming code 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:

$$\begin{aligned} p1 &= d_1 \oplus d_2 \oplus d_3 \\ p2 &= d_1 \oplus d_2 \oplus d_4 \\ p3 &= d_1 \oplus d_3 \oplus d_4 \end{aligned} \quad (3)$$

The data and parity check bits are stored and can be recovered later even if there is an error in one of the bits. This is done by re-computing the parity check bits and comparing the results with the values stored. In the example considered, an error on d1 will cause 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. Therefore, the data bit in error can be located and the error can be corrected. This is commonly formulated in terms of the generating G and parity check H matrixes. For the Hamming code considered in the example, those are

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \quad (4)$$

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

Encoding is done by computing y = x • G and error detection is done by computing s = y • HT, where the operator • is based on module two addition (XOR) and multiplication. Correction is done using the vector s, known as syndrome, to identify the bit in error. The correspondence of values of s to error position is captured in Table I.

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	d_1	correct d_1
1 1 0	d_2	correct d_2
1 0 1	d_3	correct d_3
0 1 1	d_4	correct d_4
1 0 0	p_1	correct p_1
0 1 0	p_2	correct p_2
0 0 1	p_3	correct p_3

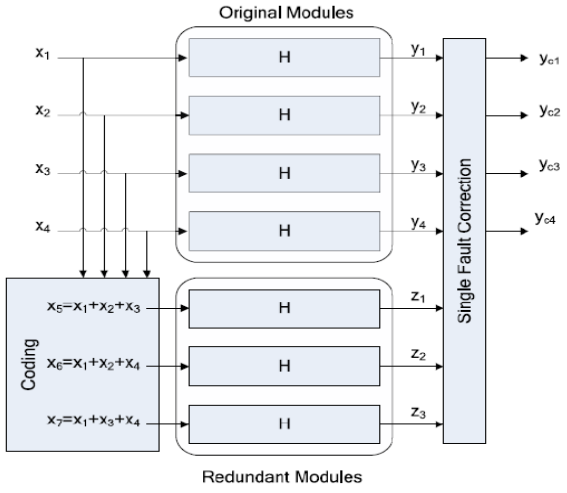

Fig.2. Proposed scheme for four filters and a Hamming code.

Once the erroneous bit is identified, it is corrected by simply inverting the bit. This ECC scheme can be applied to the parallel filters considered by defining a set of check filters zj. For the case of four filters y1, y2, y3, y= and the Hamming code, the check filters would be

$$z_1[n] = \sum_{l=0}^{\infty}(x_1[n-l] + x_2[n-l] + x_3[n-l]).h[l]$$
$$z_2[n] = \sum_{l=0}^{\infty}(x_1[n-l] + x_2[n-l] + x_4[n-l]).h[l]$$
$$z_3[n] = \sum_{l=0}^{\infty}(x_1[n-l] + x_3[n-l] + x_4[n-l]).h[l] \quad (6)$$

and the checking is done by testing if

$$\begin{aligned} z_1[n] &= y_1[n] + y_2[n] + y_3[n] \\ z_2[n] &= y_1[n] + y_2[n] + y_4[n] \\ z_3[n] &= y_1[n] + y_3[n] + y_4[n] \end{aligned} \quad (7)$$

For example, an error on filter y1 will cause errors on the checks of z1, z2, and z3. Similarly, errors on the other filters will cause errors on a different group of zi. Therefore, as with the traditional ECCs, the error can be located and corrected. The overall scheme is illustrated on Fig. 2. It can be observed that correction is achieved with only three redundant filters. For the filters, correction is achieved by reconstructing the erroneous outputs using the rest of the data and check outputs. For example, when an error on y1 is detected, it can be corrected by making

$$y_{c1}[n] = z_1[n] - y_2[n] - y_3[n] \quad (8)$$

Similar equations can be used to correct errors on the rest of the data outputs. In our case, we can define the check matrix as

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & -1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & -1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & -1 \end{bmatrix} \quad (9)$$

and calculate s = yHT to detect errors. Then, the vector s is also used to identify the filter in error. In our case, a nonzero value in vector s is equivalent to 1 in the traditional Hamming code. A zero value in the check corresponds to a 0 in the traditional Hamming code.

It is important 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 widely studied for different filter structures. The interested reader is referred to for further details. Therefore, a threshold must be used in the comparisons so that values smaller than the threshold are classified as 0. This means that small errors may not be corrected. This will not be an issue in most cases as small errors are acceptable. The detailed 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 reader can get more details on this type of analysis. With this alternative formulation, it is clear that the scheme can be used for any number of parallel filters and any linear block code can be used. The approach is more attractive when the number of filters k is large. For example, when k = 11, only four redundant filters are needed to provide single error correction. This is the same as for traditional ECCs for

which the overhead decreases as the block size increases.

The additional operations required for encoding and decoding are simple additions,subtractions, and comparisons and should have little effect on the overall complexity of the circuit. This is illustrated in which a case study is presented. In the discussion, so far the effect of errors affecting the encoding and decoding logic has not been considered. The encoder and decoder include several additions and subtractions and therefore the possibility of errors affecting them cannot be neglected. Focusing on the encoders, it can be seen that some of the calculations of the zi share adders. For example, looking at (6), z1 and z2 share the term y1 + y2. Therefore, an error in that adder could affect both z1 and z2 causing a mis-correction on y2. To ensure that single errors in the encoding logic will not affect the data outputs, one option is to avoid logic sharing by computing each of the zi independently. In that cases, errors will only affect one of the zi outputs and according to Table I, the data outputs yj will not be affected. Similarly, by avoiding logic sharing, single errors in the computation of the s vector will only affect one of its bits. The final correction elements such as that in (8) need to be tripled to ensure that they do not propagate errors to the outputs. However, as their complexity is small compared with that of the filters, the impact on the overall circuit cost will be low. This is confirmed by the results presented for a case study.

TABLE II: Resource Comparison for Four Parallel FIR Filters

	Unprotected	TMR	Method in [7]	Proposed
Slices	2944	9020	7740	6409
Flip-flops	1224	3984	3980	2941
LUTs	5692	17256	13640	12032

TABLE III: Resource Comparison for Eleven Parallel Fir Filters

	Unprotected	TMR	Method in [7]	Proposed
Slices	8096	24805	21285	14422
Flip-flops	3366	10956	10945	6478
LUTs	15653	47454	37510	28331

IV. EVALUATION

The case studies presented in the previous section have been implemented in HDL and mapped to a Xilinx Virtex 4 FPGA XC4VLX80. The added logic for coding and error correction is protected with TMR to ensure that errors do not affect the corrected filters outputs. The parallel filters are FIR filters with 16 coefficients. The input data and coefficients are quantized with 8 bits. The filter output is quantized with 18 bits. To avoid saturation, the inputs for the first and second redundant filters are expanded to 10 bits and 12 bits for the first case study, and to 11 bits and 14 bits for the second. In addition to the proposed scheme, the ECC based scheme presented it has also been implemented to assess the cost benefits of the new technique. The results in terms of resource usage are summarized on Tables IV and V. It can be observed that for both case studies, the proposed scheme reduces the implementation cost for all resource types compared to the ECC based scheme. That scheme needs three and four redundant filters in each case as opposed to only two in the new scheme. This explains the observed reductions. Ideally, if the error detection part is not considered, the savings should be approximately 1/7 (or 14.3%) for the first

case study and 2/12 (or 16.6%) in the second. However, the actual saving in slices and lookup tables isa little bit lower. This is because: 1) the error detection/correction logic is more complex in the new scheme than and2) the complexity of the redundant filters is slightly higher than that in ECC scheme due to the wider inputs.

TABLE IV: Resource Comparison for Four Parallel FIR Filters

	Unprotected	ECC based [13]	Proposed	Saving
Slices	2944	6611	5792	12.4%
Flip-flops	1328	3196	2408	24.7%
LUTs	5692	12401	11171	9.9%

TABLE V: Resource Comparison for Eight Parallel FIR Filters

	Unprotected	ECC based [13]	Proposed	Saving
Slices	5888	11589	9872	14.8%
Flip-flops	2656	5286	3994	24.4%
LUTs	11384	21872	19136	12.5%

Fault injection experiments have been conducted to assess the effectiveness of the scheme to correct errors. In the experiments, 10000 single event upsets have been randomly inserted respectively in the coefficients and inputs of each filter, and a tolerance level of 1 is used for the checks. Results show that all faults that introduce errors out of the range [-1, 1] can be detected and corrected. This confirms the effectiveness of the scheme to correct single errors which is similar to the protection provided by the previous ECC scheme.

V. CONCLUSION

This brief has presented a new scheme to protect parallel filters that are commonly found in modern signal processing circuits. The approach is based on applying ECCs to the parallel filters outputs to detect and correct errors. The scheme can be used for parallel filters that have the same response and process different input signals. A case study has also been discussed to show the effectiveness of the scheme in terms of error correction and also of circuit overheads. The technique provides larger benefits when the number of parallel filters is large. The proposed scheme can also be applied to the IIR filters. Future work will consider the evaluation of the benefits of the proposed technique for IIR filters. The extension of the scheme to parallel filters that have the same input and different impulse responses is also a topic for future work. The proposed scheme can also be combined with the reduced precision replica approach presented to reduce the overhead required for protection. This will be of interest when the number of parallel filters is small as the cost of the proposed scheme is larger in that case. Another interesting topic to continue this brief is to explore the use of more powerful multi-bit ECCs, such as Bose–Chaudhuri–Hocquenghem codes, to correct errors on multiple filters.

VI. REFERENCES

[1] Zhen Gao, Pedro Reviriego, Zhan Xu, Xin Su, Jing Wang and Juan Antonio Maestro, “Efficient Coding Schemes For Fault Tolerant Parallel Filters”, IEEE Transactions on Circuits and Systems II: Express Briefs, 2015. [2] P.P Vaidyanathan. “Multirate Systems and Filter Banks”, Prentice Hall, 1993. [3] A. Sibille, C. Oestges and A. Zanella “MIMO: From Theory to Implementation”, Academic Press, 2010. [4] N. Kanekawa, E. H. Ibe, T. Suga and Y. Uematsu, “Dependability in Electronic Systems: Mitigation of Hardware

Failures, Soft Errors, and Electro-Magnetic Disturbances", Springer Verlag, 2010. [5] M. Nicolaidis, "Design for soft error mitigation", IEEE Trans. on Device and Materials Reliability, vol. 5, no. 3, pp. 405–418, Sept. 2005. [6] C. L. Chen and M. Y. Hsiao, "Error-correcting codes for semiconductor memory applications: a state-of-the-art review", IBM J. of Research and Development, vol. 28, no. 2, pp. 124-134, 1984. [7] A. Reddy and P. Banarjee "Algorithm-based fault detection for signal processing applications", IEEE Trans. on Computers, vol. 39, no. 10, pp. 1304-1308, Oct. 1990. [8] S. Pontarelli, G. C. Cardarilli, M. Re, and A. Salsano, "Totally fault tolerant RNS based FIR filters," in Proc. of IEEE International Online Test Symposium (IOLTS), 2008, pp. 192–194. [9] 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. of the IEEE International Online Test Symposium (IOLTS), 2012. [10] B. Shim and N. Shanbhag, "Energy-efficient soft error-tolerant digital signal processing," IEEE Trans. on Very Large Scale Integration Systems, vol. 14 no. 4, pp. 336–348, 2006. [11] Y.-H. Huang, "High-Efficiency Soft-Error-Tolerant Digital Signal Processing Using Fine-Grain Sub word- Detection Processing," IEEE Trans. on Very Large Scale Integration Systems, vol. 18, no 2, pp. 291-304, Feb. 2010. [12] P. Reviriego, C. J. Bleakley, and J. A. Maestro, "Structural DMR: A Technique for Implementation of Soft- Error-Tolerant FIR Filters," IEEE Trans. on Circuits and Systems-II: Express Briefs, vol. 58, no. 8, pp. 512-516, Aug. 2011.