Error Correction Codes For Redundant Parallel Filters

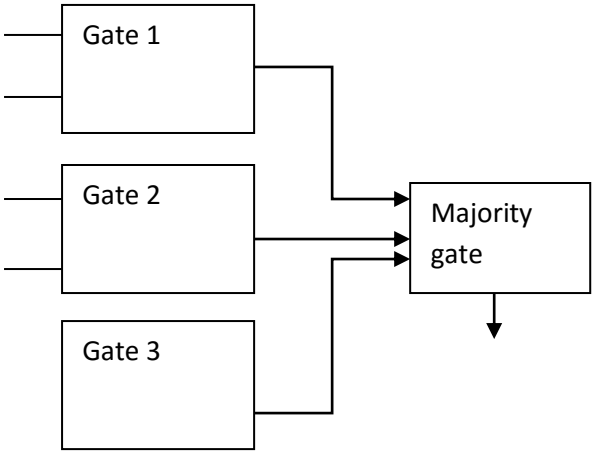
Veena S[1], M Seshananda Reddy[2], T Nikhileswara Reddy[3], V R Sudarshan[4], Vinay Kumar K[5]
Assist.proff[1] UG SJCIT

veenasdl@gmail.com[1] kevinseshu329@gmail.com[2] nikhileswarreddy097@gmail.com[3] vrsudarshan77@gmail.com[4]
vinaykondeti.vk@gmail.com[5]

ABSTRACT
The objective of this paper is to propose a simple technique that exploits the presence of parallel filters to achieve fault tolerance. Fault tolerance means a system will continue to work properly even if the fault has occurred in the system.
Now a day's digital filter are used in the signal processing systems and in communication systems. In some situations the consistently perform of the system becomes difficult, hence the fault tolerant implementation is needed. Over the period many techniques has been evolved to achieve the fault tolerance. A new technique has been developed, in which the fault tolerance will enables more complex systems tha include many filters. in those complex system, most commonly the parallel filters are used. In brief ,the fault tolerance is used to protect the parallel filters from the Error Correction Codes (ECCs).The new technique allows more efficient protection when the number of parallel filters are more in the system.
The technique is proposed using parallel finite impulse response filters showing the effectiveness in terms of protection and implementation cost.

Keywords
Error Correction Codes (ECCs), Parallel filters, digital filters

1. INTRODUCTION
Now a days Electrical circuits consisting parallel filters are increasing presently in automobile, medical and in communication where reliability is difficult. Hence some degree of fault tolerant has to be produced in those circuits. The need is further increased by the intrinsic reliability changes of advance CMOS technology. The technique to protect the circuits from the errors varies from range from modifications in the Manufacturing process of the circuits. The number of errors can be reduced by adding the redundancy at the logic or system level and we have to ensure that the errors do not affect any system functionality [1]. Triple modular redundancy (TMR) is the technique used to add the redundancy. The TMR uses the voting method to correct the errors.TMR consumes more power and area of the circuit, which is in compatible in some applications and circuits. The TMR block diagram is shown below



Digital filters are used in many signal processing circuits and there is need to protect the circuits from the errors that affect the system. In this we mainly focused on finite impulse response (FIR) filters. For example, in [3], this was proposed to reduce the cost of implementing modular redundancy in FIR filters. In [4], a relationship was established between the memory elements of an FIR filter, the input sequence was used to detect errors. Other techniques have exploited the FIR properties to achieve fault tolerance [5]. To protect filters we use number systems [6] and arithmetic codes [7] Finally, the use of different implementation structures of the FIR filters to correct errors with only one redundant module has also been proposed [8].
Now a days the systems with the several filters that operate in parallel are increasing. Fault tolerance is used in filter banks [9] and in many modern communication systems [10]. The idea was explored in [11], where two parallel filters with the same response that processed different input signals were considered. Here single error correction can be implemented. Therefore, a significant cost reduction compared with TMR was obtained.
Here a scheme is proposed to protect parallel filters is presented. As in [11], parallel filters with the same response and different input signals are considered. The application of error correction codes (ECCs) using each of the filter outputs as the equivalent of a bit in and ECC codeword is the new approach. The scheme presented in [11] and enables more efficient protection when the number of parallel filters is large. Hence fault tolerance is used to provide more powerful protection using advanced ECCs which correct failures in multiples modules.
The rest of this introduces the new scheme by first summarizing the parallel filters and then the proposed scheme is presented. Followed by a case study to illustrate the effectiveness of the approach. Finally, the conclusions.

2. PARALLEL FILTERS WITH THE SAME RESPONSE

The equation for the discrete time filter is given by:

$$y[n] = \sum_{l=0}^{\infty} x[n-l] \cdot h[l] \qquad (1)$$

Where $x[n]$ is the input signal, $y[n]$ is the output, and $h[l]$ is the Impulse response of the filter [12].
The filter is known as a FIR filter if the impulse response $h[l]$ is nonzero, only for a finite number of samples, otherwise it is an infinite impulse response (IIR) filter. We can implement both FIR and IIR filters using several algorithms.

Here it is considered a set of *k* parallel filters with the same response and different input signals as shown in figure 2.The filters with the same impulse response are shown in some communication system and in processing applications.
By adding the by adding the corresponding inputs *xi* [*n*] and filter with the same impulse response h[l] we can obtain any combination of the outputs *yi* [*n*].

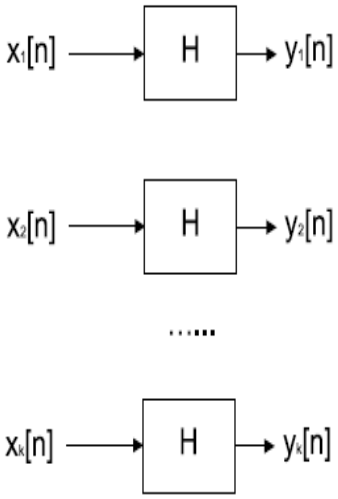


Fig. 1. Parallel filters with the same response.

Let us consider an example,

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

Above results will be helpful for the further development of proposed fault tolerant technique.

3. PROPOSED SCHEME

By using ECCs the new technique has been proposed. Let us consider simple ECC which takes a block of *k* bits and produces a block of *n* bits by adding *n*−*k* parity check bits [13]. The parity check bits are generated by combining K data bits using XOR operations In order to detect and correct errors we need to use those XOR combinations which have been properly designed. For an example, let us consider a simple Hamming code [14] with $k = 4$ and $n = 7$. By the use of (7,4) hamming code can detect and correct single bit errors and only detects two bit errors (correction of two bit errors is not possible). From the above example, three parity check bits i.e p1, *p*2, *p*3 can be determined as a function of the data bits *d*1, *d*2, *d*3, *d*4 as follows:

$$p1 = d1 \oplus d2 \oplus d3$$
$$p2 = d1 \oplus d2 \oplus d4$$
$$p3 = d1 \oplus d3 \oplus d4. \qquad (3)$$

The data and parity check bits are stored which can be recovered later if necessary, even if there is any single bit errors. This can be achieved by recalculating the parity check bits, further comparing the results with the values stored previously. In the example considered, an error on *d*1 will effect i.e. cause errors on the three parity checks ; similarly errors on *d*2 and *d*3 only effects *p*1 , *p*2 and *p*1 , *p*3 respectively; and finally an error on *d*4 in *p*2 and *p*3.Therefore, error can be located and corrected in the data bit. This is commonly prepared methodically in terms of generating *G* and parity check *H* matrixes. For the Hamming code considered in the example, those are

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

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

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

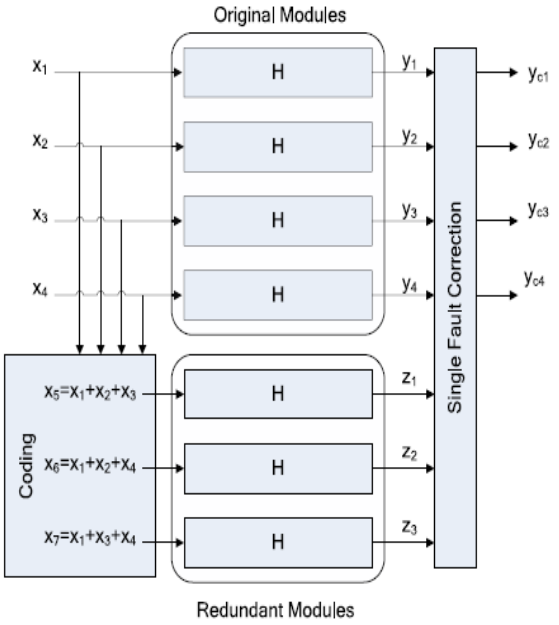


Fig. 2. Proposed scheme for four filters and a Hamming code.

Once the error bit is recognized, it is corrected by inverting the bit.
The ECC technique can be applied to the parallel filters by defining a set of check filters *z j*. For four filters *y*1, *y*2, *y*3, *y*4 and the Hamming code, the check filters can be shown as

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

And checking is done by testing if

$$z_1[n] = y_1[n] + y_2[n] + y_3[n]$$
$$z_2[n] = y_1[n] + y_2[n] + y_4[n]$$
$$z_3[n] = y_1[n] + y_3[n] + y_4[n]. \quad (7)$$

Consider an example, the error on filter *y*1 will effect on the checks filters of *z*1, *z*2, and *z*3. Similarly, errors on the other filters will cause errors on a different group of check filters *zi* . Therefore, the error can be located and corrected.by the Error Correction Codes.

The overall structure is shown in fig 2 ,three redundant filters are used for the correction of errors.

For the filters, correction can be done by reconstruction of erroneous outputs using remaining data and check outputs. Suppose, when an error on *y*1 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 this case, we can define the check matrix as follows

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

and calculate $s = yH^T$ to detect errors. After calculation and detection the vector *s* is also used to identify the filter in error. A nonzero value in vector *s* is equivalent to 1 in the traditional Hamming code and A zero value in the check corresponds to a 0 in the traditional Hamming code.

The one Important thing is to observe that due to different finite precision effects in the original and check filter implementations, the comparisons in (7) can show small differences and those differences will depend on the quantization effects in the filter implementations that have been widely studied for different filter structures. Referred to [12] for further details. A Threshold must be used in the comparisons. The values which are smaller than the threshold are classified as 0. As the threshold value are classified as 0 small errors may not be corrected. This will not become a problem because in many of the cases small errors are acceptable. The effect of these small errors on the signal to noise ratio at the output of the filter is left for future work. Refer to [3] for more details on this type of analysis.With the help of this alternative formulation we can clearly say that the scheme can be used for any number of parallel filters and any linear block code can be used. This approach will become more attractive when the number of filters *k* is large. Let us consider one example, when *k* = 11, to provide single error detection only four redundant filters are needed. This will be same as for traditional ECCs for which the overhead decreases as the block size increases [13].

For encoding and decoding the extra operations required are simple additions, subtractions, comparisons and should have a little effect for overall complexity of the circuit and this is clarified in Section IV in which a case study is presented. As we seen in the above discussion the effect of errors affecting the encoding and decoding logic has not been considered. Both the encoder and the decoder includes several additions and subtractions. For this purpose the possibility of errors affecting them cannot be neglected. It can be seen on the encoders that some of the calculations of the *zi* share adders. A small example for this is looking at (6), *z*1 and *z*2 share the term *y*1 + *y*2. An error in that adder could affect both *z*1 and *z*2 which causes a miscorrection on *y*2. To make sure that single errors in the encoding logic will not affect the data outputs the one option is to avoid logic sharing by computing each of the *zi* independently. For the above case the errors will only affect one of the *zi* outputs and data outputs *yj* will not be affected according to Table I. Likewise by avoiding logic sharing the single errors in the computation of the *s* vector will only affect one of its bits. Final correction elements such as that in (8) need to be tripled to make sure that they do not propagate errors to the outputs. As their complexity is small compared with that of the filters the overall circuit cost will be low. This is confirmed by the results presented in Section IV 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

4. CASE STUDY

A case study is used to evaluate the effectiveness of the proposed scheme. Let us considered a set of parallel FIR filters with 16 coefficients is considered. The input data, the coefficients are quantized with 8 bits and the filter output is quantized with 18 bits. For the check filters *zi* the input is the

sum of several inputs *xj*, and the input bit-width is extended to 10 bits. A Small threshold is used in the comparisons because of that errors which are smaller than the threshold are not considered errors. As before explained in Section II, no logic sharing was used in the computations in the encoder and in the decoder logic to avoid errors on them from propagating to the output.

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

Two evaluations can be done in this, the first evaluation is to compare the resources used by the proposed scheme with those used by TMR, the protection method which is proposed in [7] (with $m = 7$) and by an unprotected filter implementation. The results are presented in Tables II and III for each of the configurations considered. The proposed technique provides significant savings i.e. from 26% to 41% for all the resource types (slices, flip-flops, and LUTs) compared with the TMR. For the second configuration the benefits are larger as expected with values exceeding 40% for all resource types. The relative number of added check filters $(n - k)/n$ is smaller. After comparing with the arithmetic code technique proposed in [7], the savings are smaller but still significant ranging from 11% to 40%. For the second configuration again the larger savings are obtained.

In brief the case study results confirm that the proposed scheme can reduce the implementation cost significantly compared with the TMR and also provides reductions when compared with other methods such as that in [7]. When the number of filters is large the reductions are larger.

The second evaluation is to determine the effectiveness of the scheme and to correct the errors. Fault injection experiments have been conducted to that end. Specifically errors have been randomly inserted in the coefficients and inputs of the filters. Single errors were detected and corrected in all the cases. Totally 8000 errors for inputs and 8000 errors for filter coefficients were inserted in the different simulation runs and it confirms the effectiveness of the scheme to correct single errors.

5. SIMULATION AND SYNTHESIS

HDL is an integral part of such tools and offers the designer a very efficient tool for implementing and synthesizing designs on chips. The two widely used hardware description languages are VHDL and Verilog. VHDL stands for Very High Speed Integrated Circuit. Design synthesis from higher level of abstraction to lower level of abstraction .The Fault Tolerant Parallel Filters Based on Error Correction Codes is carried out on Spartan-3FPGA development kit. In our implementation, Xilinx Spartan-3 with FPGA is used fig 3.

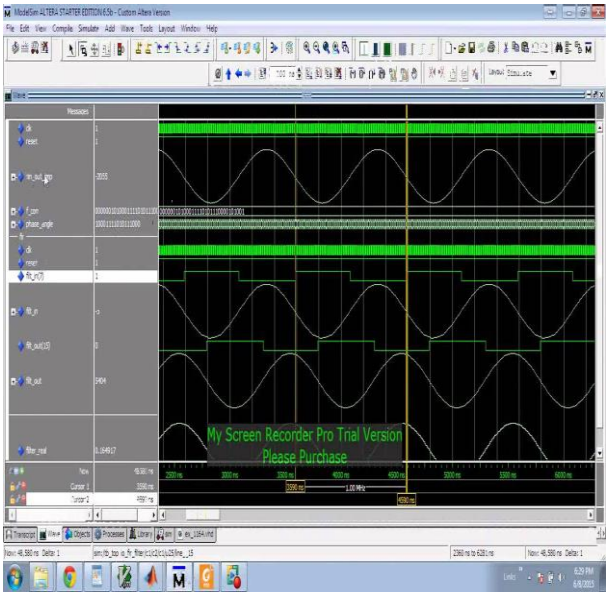


Fig 3: Simulation results

6. CONCLUSIONS

This brief description has given a new method for the protection of parallel filters which were similarly found in modern signal processing circuits. The main scheme of approach for detection and correction of errors is purely based on applying ECCs to the parallel filters outputs. The scheme can be used for parallel filters that have the same response and process different input signals and where the parallel filters are used in large number.

A brief discussion shows the effectiveness of the scheme in terms of error correction and also of circuit overheads. When the number of parallel filters is large these schemes provide much more benefits.

Not only the proposed scheme can be applied to the IIR filters but also further future work will be providing the evaluation of the benefits of the proposed technique for IIR filters. The scheme can be extended to parallel filters that have the same input and different impulse responses will be carried out by doing future work. Whenever the number of parallel filters is small as the cost of the proposed scheme is larger, in that case there is a proposed scheme which is a combination of the reduced precision replica approach presented in [3] which provides the protection by reducing the overhead i.e. required for protection. The extension of above topic can be made by the use of more powerful multibit ECCs, such as Bose–Chaudhuri–Hocquenghem codes, to correct errors on multiple filters.

7. ACKNOWLEDGEMENT

We express our sincere thanks to Dr. K M Ravi kumar HOD, Dept of ECE, and veena s, assistant professor, Dept of ECE.

8. 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 signal 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 processing 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. Vaidyanathan. *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 Implementation*. 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] R. W. Hamming, "Error correcting and error detecting codes," *Bell Syst. Tech. J.*, vol. 29, pp. 147–160, Apr. 1950