ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Area Efficient and Fault Tolerant Parallel Fir Filter Based on ECC

B.Umadevi
PG Scholar,
Electronics and Communication Engineering,
Gates Institute of Technology, AP, India.

S.Ramesh Babu
Assistant Professor,
Electronics and Communication Engineering,
Gates Institute of Technology, AP, India.

ABSTRACT:
The complexity of communications and signal processing circuits increases every year. This is made possible by the CMOS technology scaling that enables the integration of more and more transistors on a single device. This increased complexity makes the circuits more vulnerable to errors. At the same time, the scaling means that transistors operate with lower voltages and are more susceptible to errors caused by noise and manufacturing variations. 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. Soft errors pose a reliability threat to modern electronic circuits. This makes protection against soft errors a requirement for many applications. Communications and signal processing systems are no exceptions to this trend.

For some applications, an interesting option is to use algorithmic-based fault tolerance (ABFT) techniques that try to exploit the algorithmic properties to detect and correct errors. Signal processing and communication applications are well suited for ABFT. A general scheme to protect parallel filters is presented. 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.

1 INTRODUCTION:
FIR filters are one of two primary types of digital filters used in digital signal processing (DSP) applications, the other type being IIR. High performance FIR filters have applications in several video processing and digital communications systems. In some applications, the FIR filter circuit must be able to operate at high sample rates, while in other applications, the FIR filter circuit must be a low-power circuit operating at moderate sample rates. The low-power or low-area techniques developed specifically for digital filters can be found in [1, 2, 3, 4, 5, 6, 7].
Traditional FIR filter uses some parallel processing technique to either increase the effective throughput or to reduce the power consumption of the original filter. Parallel processing involves the replication of hardware units.

Here the hardware implementation cost is directly proportional to the block size. At the same time if the design area is very limited this technique is not applicable. Therefore, in order to reduce the chip size and to limit the silicon area required to implement the FIR filter it is necessary to realize a new parallel FIR filtering structure that consume less area than traditional parallel FIR filtering It is common in DSP to say that a filter input and output signals are in time domain. This is because signals are usually created by sampling at regular intervals of time. But this is not the only way sampling can take place. The second most common way of sampling is at equal intervals in space. For example imagine taking simultaneous readings from an array of strain sensors mounted at one centimeter increments along the length of an aircraft wing.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Many other domains are possible; however, time and space are by far the most common. When you see the term time domain in DSP, remember that it may actually refer to samples taken over time, or it may be a general reference to any domain that the samples are taken in. Every linear filter has an impulse response, a step response and a frequency response. Each of these responses contains complete information about the filter, but in a different form. If one of three is specified, the other two are fixed and can be directly calculated. All three of these representations are important, because they describe how the filter will react under different circumstances.

The most straightforward way to implement a digital filter is by convolving the input signal with the digital filter's impulse response. All possible linear filters can be made in this manner. When the impulse response is used in this way, filters designers give it a special name: the filter kernel. There is also another way to make digital filters, called recursion. When a filter is implemented by a convolution, each sample in the output is calculated by weighting the samples in the input, and adding then together. Recursive filters are an extension of this, using previously calculated values from the output, besides points from the input. Instead of using a filter kernel, recursive filters are defined by a set of recursion coefficients.

For now the important point is that all linear filters have an impulse response, even if you don't use it to implement the filter. To find the impulse response of a recursive filter, simply feed in the impulse and see what comes out. The impulse responses of recursive filters are composed of sinusoids that exponentially decay in amplitude. In principle, this makes their impulse responses infinitely long. However the amplitude eventually drops below the round off noise of the system, and the remaining samples can be ignored. Because of these characteristics, recursive filters are also called Infinite impulse response or IIR filters. In comparison, filters carried out by convolution are called Finite impulse response or FIR filters.

2. PARALLEL FILTERS WITH THE SAME RESPONSE

A discrete time filter implements the following equation:

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

Where x[n] is the input signal, y[n] is the output, and h[l] is the impulse response of the filter [12]. 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.

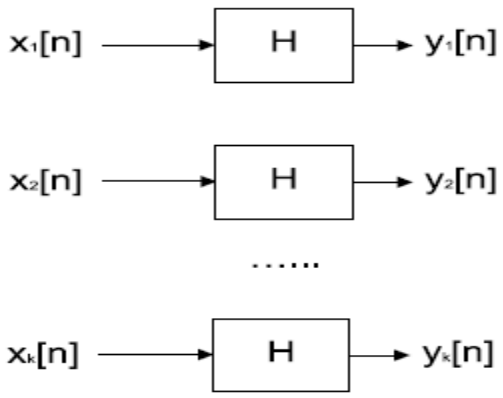


Figure 1: Parallel filters with the same response.

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 the same filter h[l]. For example

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

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

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

3. 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 [13]. 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 [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:

$$p_1 = d_1 \oplus d_2 \oplus d_3$$
$$p_2 = d_1 \oplus d_2 \oplus d_4$$
$$p_3 = d_1 \oplus d_3 \oplus d_4. \quad (3)$$

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 recomposing 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.

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

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

Table 1: 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

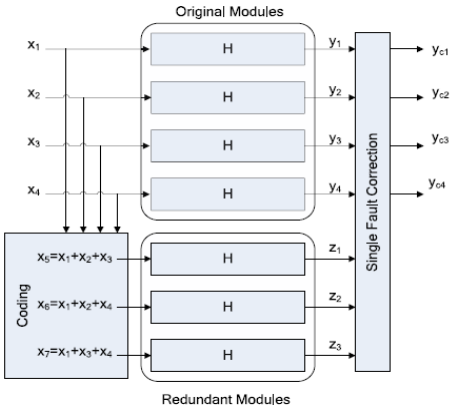


Figure 2: Proposed scheme for four filters and a Hamming code.

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

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

This ECC scheme can be applied to the parallel filters considered by defining a set of check filters z j . For the case of four filters y1, y2, y3, y4 and the Hamming code, the check filters would be and the checking is done by testing if

z1[n] = y1[n] + y2[n] + y3[n]
z2[n] = y1[n] + y2[n] + y4[n]
z3[n] = y1[n] + y3[n] + y4[n]. (7)

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

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. 4. 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.

$$yc1[n] = z1[n] - y2[n] - y3[n]. \quad (8)$$

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

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

In our case, we can define the check matrix as 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 [12] 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 in [3].

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 [13]. 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. 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 miscorrection 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 case, errors will only affect one of the zi outputs and according to Table 5, the data outputs y j 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.

4 RESULTS:

4.1 Implementation

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

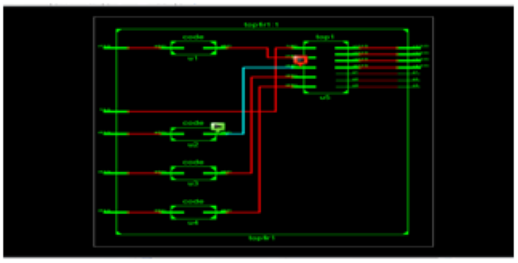
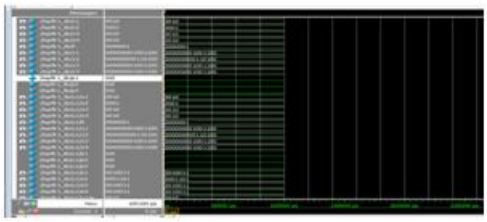


Figure 3: Internal RTL Schematic

4.2. Window shows getting an error:

This window shows, the all check bit values are same no error occure in the circuit.



Fiure 4 : Getting an NO error output

5. 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.

6.FUTURE SCOPE:

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 in [3] 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.

BLIOGRAPHY:

[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.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

[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.