FPGA Implementation of Hard Error Correction Technique using Fault Tolerant Method

Swetha k
PG student, Dept. of ECE, DBIT
Bangalore, Karnataka, India

Murugesh k
Asst. Professor, Dept. of ECE, DBIT
Bangalore, Karnataka, India

***Abstract*—As the complexity of communications and signal processing systems increases, so does the number of blocks or elements that they have. In many cases, some of those elements operate in parallel performing the same processing on different signals. A typical example of those elements is digital filters. The increase in complexity also poses reliability challenges and creates the need for fault tolerant implementations. 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 brief it is shown that how parallel filters can be protected using the ECC technique. Here the bits are given as the inputs and the technique is presented. This technique shows how efficiently it works**

Keywords: Soft error,Error correction codes

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 [1]. 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, the use of reduced precision replicas was proposed to reduce the cost of implementing modular redundancy in FIR filters. In [4], 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.

There are many filters operate in parallel manner and its common as well as its increasing rapidly. There are many cases which is given below an example and communication system.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 previous work, 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. By using all this technique cost reduction when compared to TMR technique.

Basically there are some techniques which are present. We can see inearlier work, the same response with parallel filters that process different input signals are taken into consideration. The new technique 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 in and enables more efficient implementations when the number of parallel filters is large. If there are any failures in multiple modules Error Correction codes can be used.

The rest of brief introduction to the new scheme by first summarizing the parallel filters considered in Section II. Then, in Section III, the proposed methodology is presented. Section IV presents a case study to illustrate the effectiveness of the approach. Finally, the conclusions and future work are summarized in Section V.

II. PARALLEL FILTERS WITH THE SAME RESPONSE

An discrete time filter implementation is shown in below equation:

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

where $x[n]$ is the input signal, $y[n]$ is the output, and $h[l]$ is the impulse response of the filter. 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 the same filter $h[l]$.

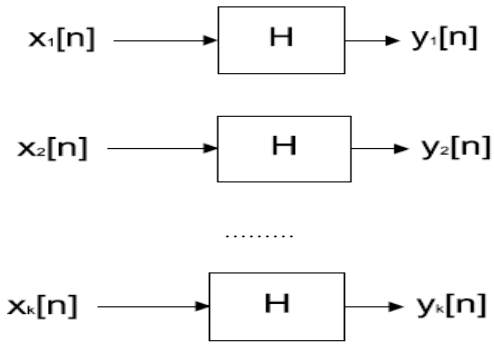


Fig. 1. Parallel filters with the same response.

For example

$$Y1[n]+y2[n]=\sum_{l-0}^{\infty}(x1[n-l]+x2[n-l).h(l) \quad (2)$$

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 [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:

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

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 recomputing 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 $y = x \bullet G$ and error detection is done by computing $s = y \bullet HT$, where the operator $\bullet$ 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

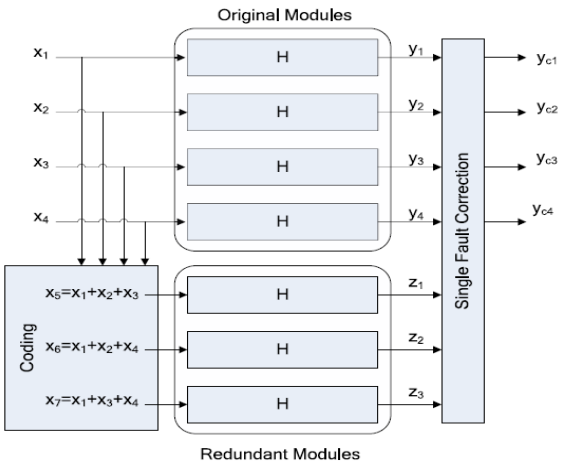


Fig. 2. Proposed scheme for four filters and a Hamming code.

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 $z\,j$. For the case of four filters$y1$, $y2$, $y3$, $y4$ and the Hamming code, the check filters would be

$$Z1[n] = \sum_{l=0}^{\infty}(x1[n-l]+x2[n-l]+x3[n-l]).h[l]$$

$$Z2[n]=\sum_{l=0}^{\infty}(x1[n-l]+x2[n-l]+x4[n-l]).h[l]$$

$$Z3[n]=\sum_{l-0}^{\infty}(x1[n-l]+x3[n-l]+x4[n-l]).h[l] \quad (6)$$

and the checking is done by testing if

$$z1[n] = y1[n] + y2[n] + y3[n]$$

$$z2[n] = y1[n] + y2[n] + y4[n]$$

$$z3[n] = y1[n] + y3[n] + y4[n] \quad (7)$$

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

$$Yc[n]=z1n]-y2[n]-y3[n] \quad (8)$$

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 $s = yH^T$ 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. This is illustrated in Section IV 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 is 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 case, errors will only affect one of the zi outputs and according to Table I, 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. 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

IV. CASE STUDY

To evaluate the effectiveness of the proposed scheme, a case study is used. A set of parallel FIR filters with 16 coefficients is considered. The input data and coefficients are quantized with 8 bits. The filter output is quantized with 18 bits. For the check filters zi, since the input is the sum of several inputs $x\ j$, the input bit-width is extended to10 bits. A small threshold is used in the comparisons such that errors smaller than the threshold are not considered errors. As explained in Section III, no logic sharing was used in the computations in the encoder and decoder logic to avoid errors on them from propagating to the output. Two configurations are considered. The first one is a block off our parallel filters for which a Hamming code with k = 4 and n = 7is used. The second is a 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 Vertex 4XC4VLX80 device.

The first evaluation is to compare the resources used by the proposed 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 proposed technique provides significant savings (from 26% to 41%) for all the resource types (slices, flip-flops, and LUTs) compared with the TMR. The benefits are larger for the second configuration as expected with values exceeding 40% for all resource types. In that case, the relative number of added check filters $(n - k)/n$ is smaller. When compared with the arithmetic code technique proposed 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 cost significantly compared with the TMR and provides also reductions when compared with other methods such as that in [7]. As discussed before, the reductions are larger when the number of filters is large.

The second evaluation is to assess the effectiveness of the scheme to correct errors. To that end, fault injection experiments have been conducted. In particular, errors have been randomly inserted in the coefficients and inputs of the filters. In all cases, single errors were detected and corrected. In total, 8000 errors for inputs and 8000 errors for filter coefficients were inserted in the different simulation runs. This confirms the effectiveness of the scheme to correct single errors.

V. CONCLUSION

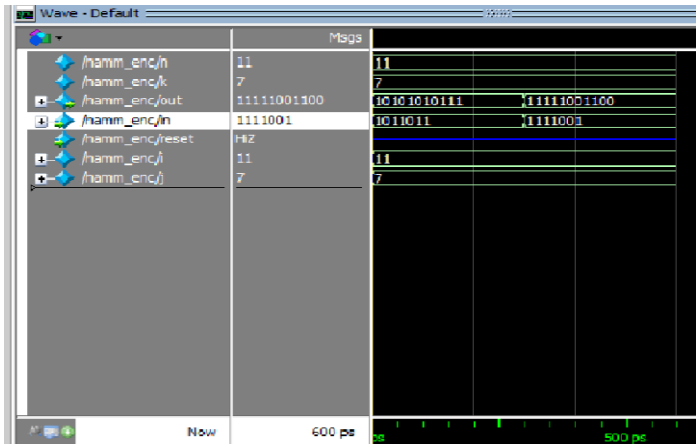
In modern signal processing system the proposed method is more frequently used. The detailed study is done here in this approach. To detect and correct errors we use this ECCs .This scheme can also be used were the same response and process it for various input signals. It also shows that this system is more effective when compared to other techniques.

This system is more efficient and effective when number of input signals are more .this method can also applied for the 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 .here in this scheme ECC is applied to 7 bit data. It has been enhanced to 128 bits and encoding and decoding is calculated. Here the comparison is done with other technique and show that proposed technique is more efficient and effective. For future work we can consider the pipeline technique instead of parallel filter. The proposed scheme can also be combined with the reduced precision replica approach presented in [3] to reduce the overhead required for protection.

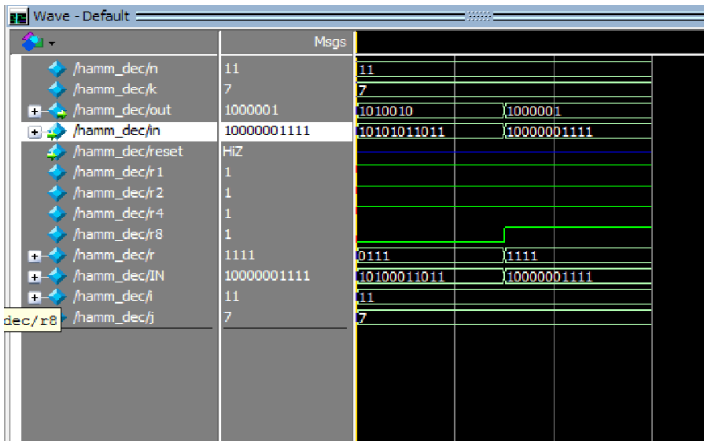
Another interesting topic to continue this brief is to explore the use of more powerful multi bit ECCs, such as Bose–Chaudhuri–Hocquenghem codes, to correct errors on multiple filters. We can also give numbers instead of bits then it is converted to bits and encoding and decoding can be obtained.

VI. SIMULATION RESULTS

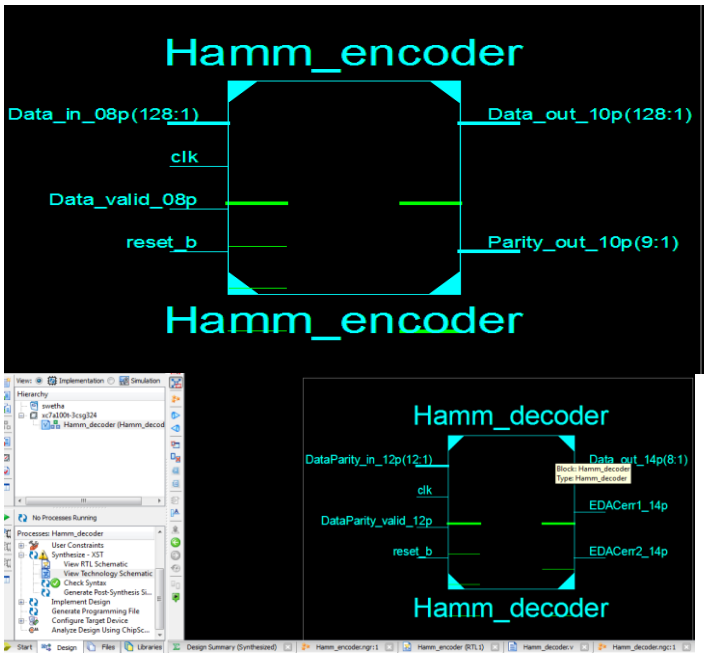
Encoding technique for seven bit data.

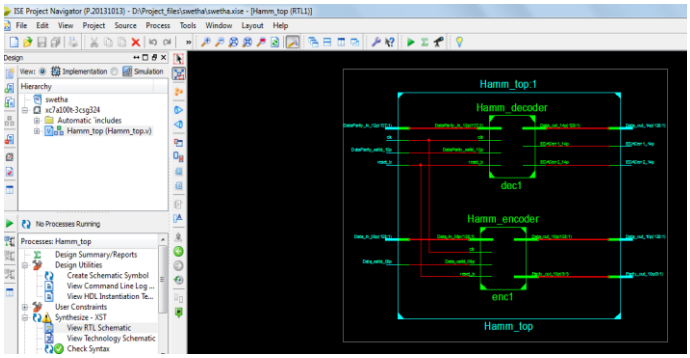


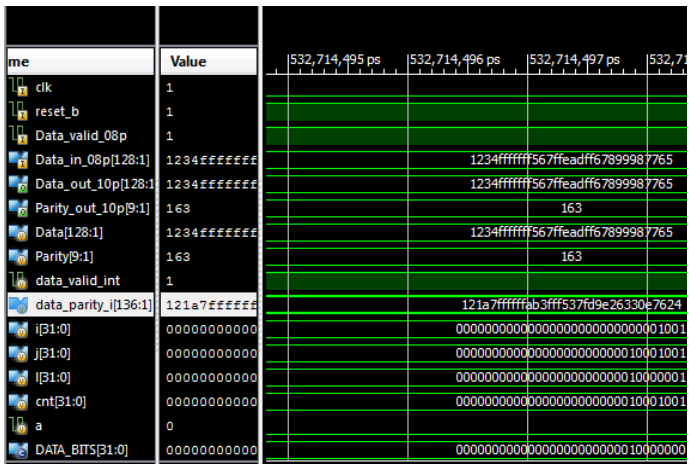
Decoder technique for seven bit data



RTL SCHEMATIC







REFERENCES

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

[2] P. Reviriego, J. A. Maestro, C. Bleakley and S. Pontarelli "Area efficient concurrent error detection and correction for parallel filters,"IET Electron. Lett., vol. 48, no. 20, pp. 1258–1260, Sep. 2012.

[3] P. Reviriego, C. J. Bleakley, and J. A. Maestro, "Structural 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.

[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] A. Sibille, A. Zanella ,and C. Oestges MIMO: From Theory to Implementation. San Francisco, CA, USA: academic Press, 2010.

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

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