Very-low-complexity concurrent error detection for transform-based filters

P. Reviriego, C.J. Bleakley and J.A. Maestro

A very-low-complexity concurrent error detection technique for filters implemented in the frequency domain using the fast Fourier transform and the overlap-add method is presented. The proposed approach checks the first or last sample of every output block by comparing it with the equivalent value calculated in the time domain. The check requires only one additional multiplication per output block. Thus, the computational complexity of the check is negligible. It is shown that the method is capable of detecting a significant percentage of errors.

Introduction: Decreases in process geometry and operating voltage are making soft errors more common. This jeopardises the reliability of digital systems [1]. Algorithm level fault tolerance has been used for many years in digital signal processing (DSP) systems to detect and correct errors [2]. Owing to its importance in DSP applications, many schemes for detection of errors in the computation of FFT-based convolutions have been proposed (see [3] and References in it). Let us consider filtering a long input sequence $x[n]$ using a filter with impulse response $h[n]$ such that only $h[0]$ to $h[M-1]$ are nonzero. This filtering operation can be implemented by linear convolution in the time domain to give the output sequence $y[n]$ as follows:

$$y[n] = \sum_{i=0}^{M-1} h[n]x[n-i] \tag{1}$$

Alternatively, to reduce computational complexity, the filter output can be computed using the frequency domain. This is achieved by dividing the input sequence $x[n]$ into blocks of length $N-M+1$ samples, zero padding each block to N samples, computing the FFT of each block, multiplying the FFT output by the FFT of the filter response and finally taking the inverse FFT (IFFT) of the product [4]. The results for consecutive blocks are then combined to obtain the final output by overlapping the block outputs by $M-1$ samples and adding the overlapped samples. The technique is based on a well-known property of FFTs, that is, multiplication in the frequency domain is equivalent to circular convolution in the time domain.

Proposed technique: Since multiplication in the frequency domain is equivalent to circular convolution in the time domain, it can be seen that the first and last output samples are equal to

$$\begin{aligned} y_b[0] &= x_b[0]h[0] + \sum_{i=1}^{N-1} x_b[N-i]h[i] \\ y_b[N-1] &= x_b[N-M]h[M-1] + \sum_{i=M}^{N-1} x_b[N-1-i]h[i] \\ &\quad + \sum_{i=0}^{M-2} x_b[N-1-i]h[i] \end{aligned} \tag{2}$$

where $x_b[n]$, $y_b[n]$ and $h[i]$ are the block input, output and filter response sequences, respectively.

Since $x_b[n]$ and $h[n]$ are zero padded from sample numbers $N-M+1$ and $M-1$, respectively, to sample $N-1$, only the first term in both equations in (2) will be nonzero. Therefore (2) can be simplified to

$$\begin{aligned} y_b[0] &= x_b[0]h[0] \\ y_b[N-1] &= x_b[N-M]h[M-1] \end{aligned} \tag{3}$$

In the proposed method, one of these time-domain calculations is used to check the output of the frequency domain convolution calculation. In the proposed method only one output sample is checked since, as is explained in the following Section, checking both samples does not significantly improve coverage.

Since rounding typically occurs in the FFT and IFFT computations, the frequency- and time-domain results for the first and last samples can differ by a small amount, even in the errorless case. Therefore, the proposed method only flags an error when the absolute difference between the frequency- and time-domain results exceeds a predetermined threshold [2]. This leads to a small loss in detection rate.

Analysis: Fig. 1 shows the structure of an IFFT implementation illustrating how, for example, an error in stage 2 propagates to the output causing errors in four output samples. In general, it can be observed that an error in stage i would propagate to 2^{S+1-i} output samples where $S = \log_2(N)$. Only errors that propagate to the first output sample are detected. Therefore the proposed method would achieve the following error coverage on the IFFT:

$$error\_coverage|_{IFFT} = \frac{\sum_{i=1}^{S+1}(2^{S+1-i}/2^S)}{S+1} \simeq \frac{2}{S+1} \tag{4}$$

This means that the coverage decreases as the block size N, and consequently S, increases. Errors in the multiplication stage are equivalent to errors in the first stage of the IFFT and therefore:

$$error\_coverage|_{Multiplication} = 1 \tag{5}$$

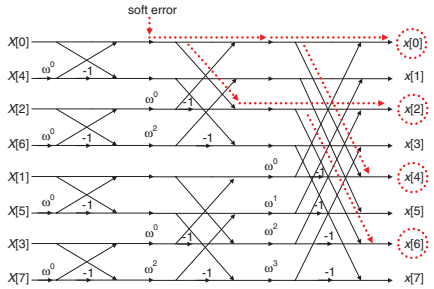

Fig. 1 *Structure of 8-point IFFT showing propagation of error in stage 2 to outputs*

Finally, for errors in the FFT, the simplest way to analyse error propagation is to consider the error pattern at the input that could have created the actual error within the FFT. Let us consider an error in the first butterfly of the second stage of the FFT. This error could have been caused by errors in $x_b[0]$ and $x_b[4]$. Since an error in $x_b[0]$ will manifest itself as an error in the first output sample (see (2)), it can be deduced that the error in the first butterfly of the second stage will be detected. However, in contrast, an error in the third butterfly of the second stage will not be detected since its equivalent input error would be at samples $x_b[1]$ and $x_b[5]$, which do not affect the first output sample if the filter is three samples or less in length (see (2)). In general, an error in stage i is equivalent to 2^{i-1} errors at the input. The equivalent error inputs will be spaced by 2^{S+1-i} samples.

If an input sample $x_b[N-k]$ suffers an error, the error will only affect the first output sample if $h[k]$ is nonzero. This will happen for M out of the N samples. Therefore the coverage in stage i can be computed as

$$error\_coverage|_{FFT\_stage} = \frac{\min(M \times 2^{i-1}, 2^S)}{2^S} \tag{6}$$

And for the complete FFT

$$error\_coverage|_{FFT} = \frac{\sum_{i=1}^{S+1}(\min(M \times 2^{i-1}, 2^S)/2^S)}{S+1} \tag{7}$$

That completes the analysis of the error detection using only the first output sample. A similar analysis applies when only the last output sample is used for detecting errors. The methods provide similar coverage. Using both checks at the same time provides only an incremental improvement in error coverage. In the FFT, the last output sample check detects errors in the final M input samples, while the first output sample check detects errors in the last $M-1$ input samples and on the first input sample. Therefore combining both checks only increases the number of input samples covered from M to $M+1$. Some additional coverage is also obtained in the IFFT but again no major improvement is made. Therefore, we concentrate on the case in which only one check is used.

Results: In this Section simulation results are presented with three objectives: first, validation of the coverage equations presented in the preceding Section, secondly, evaluation of coverage as a function of M and N and, thirdly, assessment of the impact of rounding and error thresholding on coverage.

The proposed scheme was implemented in Matlab and errors were randomly inserted in each of the computational elements. A threshold level of 10^{-5} and 100 000 fault injections were used in all the experiments. The inserted errors, the input sequence $x[n]$ and the filter coefficients $h[i]$ are uniformly distributed between -0.5 and 0.5. The error coverage results obtained and the predictions of (4), (5) and (7) are shown in Tables 1–3. It can be observed that the simulation results closely match the theoretical estimates in all cases. This means that (4), (5) and (7) can be used to provide initial coverage estimates without the need for simulations.

Table 1: Error coverage obtained by simulation in per cent for FFT for different values of N and M (values in parenthesis are theoretical estimates)

N	$M = N/16$	$M = N/8$	$M = N/4$	$M = N/2$
64	56.21 (56.25)	69.56 (69.64)	81.91 (82.14)	92.74 (92.86)
128	61.38 (61.72)	73.16 (73.44)	84.34 (84.38)	93.67 (93.75)
256	65.81 (65.97)	76.23 (76.39)	85.91 (86.11)	94.28 (94.44)
512	68.95 (69.38)	78.45 (78.75)	87.10 (87.50)	94.74 (95.00)
1024	71.82 (72.16)	80.61 (80.68)	88.40 (88.64)	95.24 (95.45)

Table 2: Error coverage obtained by simulation in per cent for multiplication stage for different values of N and M (values in parenthesis are theoretical estimates)

N	$M = N/16$	$M = N/8$	$M = N/4$	$M = N/2$
64	99.88 (100)	99.89 (100)	99.89 (100)	99.86 (100)
128	99.72 (100)	99.75 (100)	99.74 (100)	99.75 (100)
256	99.44 (100)	99.47 (100)	99.46 (100)	99.48 (100)
512	98.94 (100)	98.95 (100)	98.96 (100)	98.98 (100)
1024	97.95 (100)	97.96 (100)	97.89 (100)	97.95 (100)

Table 3: Error coverage obtained by simulation in per cent for IFFT for different values of N and M (values in parenthesis are theoretical estimates)

N	$M = N/16$	$M = N/8$	$M = N/4$	$M = N/2$
64	28.24 (28.34)	28.19 (28.34)	28.35 (28.34)	28.10 (28.34)
128	24.73 (24.90)	24.89 (24.90)	24.80 (24.90)	24.87 (24.90)
256	21.97 (22.18)	21.92 (22.18)	21.93 (22.18)	22.09 (22.18)
512	20.00 (19.98)	19.59 (19.98)	19.81 (19.98)	19.68 (19.98)
1024	17.87 (18.17)	17.68 (18.17)	17.83 (18.17)	17.92 (18.17)

The results for each of the blocks show poor coverage for the IFFT and good coverage for the multiplication stage, as was expected. In the IFFT, coverage decreases as the block size grows, as predicted by (4). For the multiplication stage, coverage is less than 100% owing to round-off effects. This reduction is larger for larger transform sizes.

The coverage of the FFT module greatly depends on the value of M. For values of $M = N/2$ or above, good coverage is obtained. This means that, in some cases, the proposed scheme can effectively detect errors in the FFT and multiplication stage and provide some coverage in the IFFT. That is an interesting result as the cost of implementing the proposed check is very small, namely a multiplication and a comparison.

Conclusions: A novel concurrent error detection (CED) technique for filters implemented with FFTs has been proposed. The scheme exploits zero padding in both the filter and the input data to implement a check that requires only one multiplication and a comparison. The check involves computing the first (or last) sample of every block output in the time-domain and comparing it with the frequency-domain result. An analysis of the scheme shows that the method can effectively detect errors in the FFT and multiplication stages when the filter length is half of the transform length or larger. Additionally, some errors in the IFFT are also detected. This finding has been corroborated by simulation. The proposed approach can be a useful CED solution in some applications as it provides CED at minimal cost.

Acknowledgment: This work was supported by a Seed Funding grant from University College Dublin.

23 September 2010
doi: 10.1049/el.2010.2653
One or more of the Figures in this Letter are available in colour online.

P. Reviriego and J.A. Maestro (*Departamento de Ingeniería Informática, Universidad Antonio de Nebrija, C. Pirineos 55, Madrid, Spain*)

E-mail: previrie@nebrija.es

C.J. Bleakley (*School of Computer Science and Informatics, University College Dublin, Belfield, Dublin4, Ireland*)

References

1 Baumann, R.: 'Soft errors in advanced computer systems', *IEEE Des. Test Comput.*, 2005, **22**, (3), pp. 258–266
2 Reddy, A., and Banarjee, P.: 'Algorithm-based fault detection for signal processing applications', *IEEE Trans. Comput.*, 1990, **39**, (10), pp. 1304–1308
3 Sundaram, S., and Hadjicostis, C.N.: 'Fault-tolerant convolution via Chinese remainder codes constructed from non-coprime moduli', *IEEE Trans. Signal Process.*, 2008, **56**, (9), pp. 4244–4254
4 Oppenheim, A.V., and Schafer, R.: 'Discrete time signal processing' (Prentice Hall, 1999)