5th SASTech
Iran |Mashhad
May 12 - 17 | 2011
5th Symposium on Advances in Science & Technology

A Framework in the Design of Fault-Tolerant Computing Systems

S.H.Hamidi, A.Vafaei, S.A.H.Monadjemi
Department of Computer Engineering
University of Isfahan
{Hamidi, Vafaei, Monadjemi}@ui.ac.ir
Paper Reference Number: 1112-330
Name of the Presenter: H.Hamidi

Abstract

This paper, we present a framework for Algorithm-based fault tolerance (ABFT) methods in design of fault tolerant computing systems. In this paper, a technique is employed that enable the combination of several codes, in order to obtain flexibility in the design of error correcting codes. Code combining techniques are very effective, which one of these codes are turbo codes. The ABFT techniques that to detect errors rely on the comparison of parity values computed in two ways, the parallel processing of input parity values produce output parity values comparable with parity values regenerated from the original processed outputs, can apply turbo codes for the redundancy.

Key words: Algorithm-based fault tolerance (ABFT), convolutional codes, turbo codes, redundancy.

1. Introduction

The important method of code combination is known as product. This product method produces excellent codes. Very low-rate convolutional codes can be constructed by taking products of binary convolutional codes and block repetition codes. Example of codes in this class includes turbo codes (Berrou et al, 1993). In this paper we present methods for employ turbo codes into systematic forms. ABFT techniques are most effective when applying a systematic form. The redundancy necessary for the ABFT method is commonly defined by real number codes, generally of the block type (Huang, and Abraham, J. A, 1984; Hamidi et al, 2009; Chen, 2008; Zhang and Liu, 2009; Sundaram and Hadjicostis, 2008). ABFT introduced by K.H.Huang and Abraham, (1984), was at first devoted to matrix operations on systolic arrays. It has been used to reduce redundant hardware. ABFT technique is distinctive by three characteristics: (a) Encoding the input sequence, (b) Plan again of the algorithm to act on the encoded input sequence, (c) Distribution of the redundant computational steps among the individual computational units in order to adventure maximum parallelism. The input sequences are encoded in the form of error detecting or correcting codes. The modified algorithm operates on the encoded data and produces encoded data output, from which useful information can be recovered very easily. Obviously, the modified algorithm will take more

time to operate on the encoded data when compared to the original algorithm; this time redundant must not be excessive. ABFT for arithmetic and numerical processing operations is based on linear codes. G. Bosilca et al. (2009) for high-performance computing (HPC), propose a new ABFT method based on a parity check coding. In (Roche et al. 2009) is the application of Low Density Parity Check (LDPC) based ABFT, it compare and analyses the use of LDPC to classical Reed-Solomon (RS) codes with regards to different fault models. But, (Roche et al. 2009) did not provide a method for constructing LDPC codes algebraically and systematically, such as RS and BCH codes are constructed, and LDPC encoding is very complex due to the miss of appropriate structure. ABFT methodologies used in (Redinbo, 1998) present parity values dictated by a real convolutional code for protecting linear processing systems. Paper (Redinbo, 2003) introduces a class of convolutional codes which is called burst-correcting convolutional codes; these codes provide error detection in a continuous mode using the same computational resources as the algorithm progresses. Redinbo (Redinbo, 2010) presented a method for Wavelet Codes into systematic forms for Algorithm-Based Fault Tolerance applications. This method employs high-rate wavelet codes along with low-redundancy which uses continuous checking attributes to detect the errors, in this paper since their descriptions are at the algorithm level can be applied in hardware or software. But, this technique is suited to image processing and data compression applications and is not a general method. Also, other constraint is on burst-error due to computational load high relatively. Moreover, there is onerous analytical approach to exact measures of the detection performances of the ABFT technique applying wavelet codes.
We make the following contributions in this paper: In section 2, we discuss our block diagram of the ABFT technique, In section 3, we propose the usage of codes, turbo codes, for ABFT technique, In section 4, Error Correction System are presented, In section 5 results are described, In section 6, discussion of the conclusions.

2. ABFT Scheme

For error correction purposes, redundancy must be inserted in some form and, using the ABFT, turbo parity codes will be employed. A systematic form of turbo codes is especially profitable in the ABFT detection plan because no redundant transformations are needed to achieve the processed data after the detection operations. To achieve fault detection and correction properties of turbo code in data processing with the minimum additional computations, we propose the block diagram in Fig 1. This figure summarizes an ABFT technique employing a systematic turbo code to define the parity values. The k is the basic block size of the input data, and n is block size of the output data, new data samples are accepted and (n - k) new parity values produced.

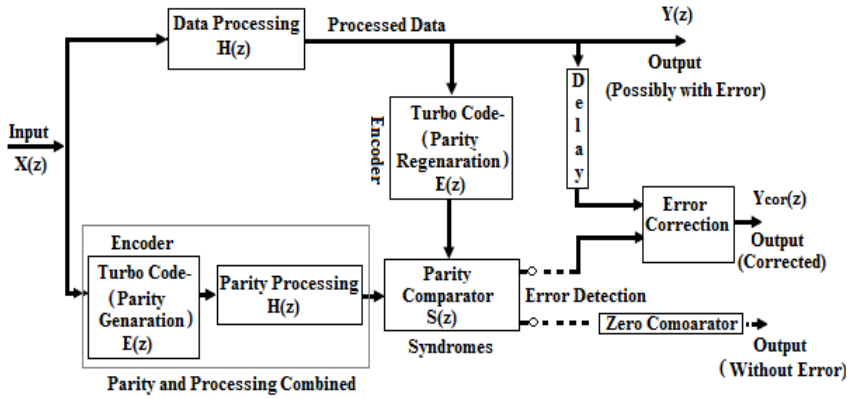

Figure 1: Block diagram of the Algorithm-based fault tolerance technique

The upper way, Fig. 1, is the processed data flow which passes through the process block (data processing block) and then feeds the turbo encoder (parity regeneration) to make parity values. On the other hand, the comparable parity values are generated efficiently and directly from the inputs (parity and processing combined, see fig.1), without producing the original outputs. The difference in the comparable two parity values, which are computed in different ways, is called the syndrome; the syndrome sequence is a stream of zero or near zero values. The turbo code's structure is designed to produce distinct syndromes for a large class of errors appearing in the processing outputs. Fig. 1 employs turbo code parity in detecting and correcting processing errors.

3. Using of Codes for ABFT Technique

3.1. Turbo Encoder

Early Most of the researchers tend to focus on methods are structured, such as RM and BCH codes with very strong algebraic structures, or topological, such as convolutional codes (Moosavie Nia and Mohammadi, 2007). Any way, structures does not always result in the best distance properties for code, and can be produced very complex operations. Special types of convolutional codes, called recursive systematic convolutional codes (RSC), are used as the building blocks of a turbo code encoder, Fig.2 (a). The basic turbo code encoder is built using two identical recursive systematic convolutional (RSC) codes with parallel concatenation (Berrou et al, 1993). The two component encoders are separated by an interleaver (π), only one of the systematic outputs from the two component encoders is used, because the systematic output from the other component encoder is just a permuted version of the chosen systematic output, Fig. 2(b). A turbo code encoder with two component codes is shown in the Fig. 2(b).

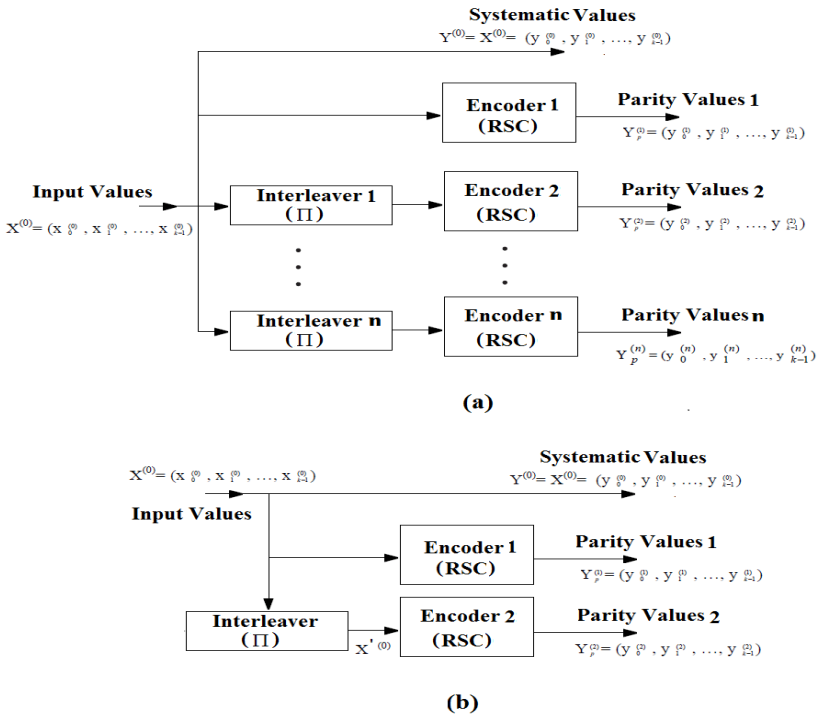

Figure 2: (a) Block diagram of the generalized turbo encoder, (b) Block diagram of the two-component turbo encoder

In the general case, the code consists of two parts: the uncoded input values and a set of parity sequences generated by passing interleaved versions of the information bits through convolutional encoders. Typically, the encoders used are Recursive Systematic encoders; also, in most turbo codes the encoders used are the same (making the Turbo code symmetric), and two sets of parity values are used, one which is generated from the non-interleaved data sequence, and one which is generated from an interleaved sequence. This structure is shown schematically in Fig. 2(b).

3.2. Iterative Turbo Code Decoder

In this paper, the turbo code decoder is based on a modified Viterbi algorithm that includes reliability values to improve decoding performance. The Viterbi algorithm produces the majority logic (ML) output value for convolutional codes. This algorithm provides optimal sequence estimation for one stage convolutional codes. For concatenated convolutional codes, there are two main disadvantages to conventional Viterbi decoders. First, the inner Viterbi decoder produces bursts of bit errors which reduce the performance of the outer Viterbi decoders (Snasel et al.2008; El Gamal et al.2001). Second, the inner Viterbi decoder produces hard decision outputs which prevent the outer Viterbi decoders from deriving the advantage of soft decisions (Berrou et al, 1993; Proakis, 2001; Wu, 2001; Hagenauer and Hoher, 1989). Both of these drawbacks can be reduced and the performance of the overall concatenated decoder can be improved if the Viterbi decoders are able to produce reliability (soft-output) values (Berrou et al, 1993). The reliability values are passed on to subsequent Viterbi decoders as apriori information to improve decoding performance. The Viterbi algorithm was modified to output bit reliability information (Hagenauer and Hoher, 1989).

The basic structure of an iterative decoder with two component codes is shown in Figure 3. Each iteration consists of two phases, one decoding phase per component decoder. First phase, in the first decoding iteration the soft-in soft-out decoder for the first component code computes the a posteriori LLR. This decoder computes the extrinsic information for each information symbol, $\Lambda^{(1)}{}_{i,e}(X^{(0)})$, on the basis of the part of the received sequence that corresponds to the parity symbols, $\bar{r}_{p1}$, and sends the result to the second decoder.

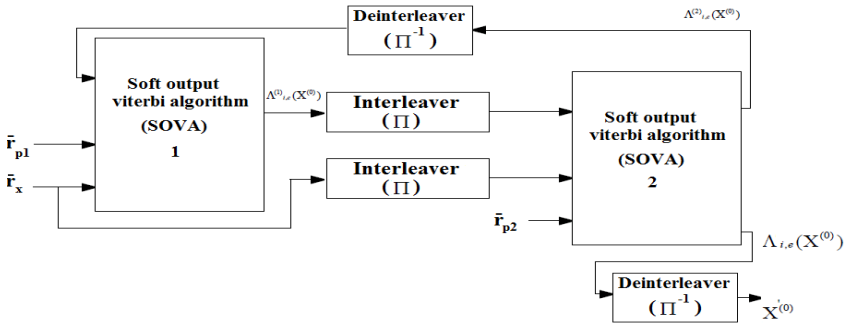

Figure 3: An iterative decoder for a parallel turbo code

In the second phase of the first decoding iteration, the permuted (or interleaved) output information from the first decoder is used as a priori LLR, $\pi.\Lambda^{(1)}{}_{i,e}(X^{(0)})$. Extrinsic information $\Lambda^{(2)}{}_{i,e}(X^{(0)})$ is computed on the basis of the part of the received sequence that corresponds to the parity values of the second component code, $\bar{r}_{p2}$, thus conclude the first decoding iteration. At this point, a decision can be made on an information symbol, on the basis of its a posteriori LLR $\Lambda_{i,e}(X^{(0)})$. In subsequent iterations, the first decoder uses the deinterleaved extrinsic information from the second decoder, $\pi^{-1}.\Lambda^{(2)}{}_{i,e}(X^{(0)})$, as a priori LLR for the computation of the soft-output (the a posteriori LLR), $\Lambda_{i,e}(X^{(0)})$. This procedure can be repeated until either a stopping criterion is met (Hagenauer et al., 1996; Sadjadpour et al., 2001) or a maximum number of iterations are performed. It should be noted that making decisions on the information symbols after the first decoder saves one deinterleaver.

3.3. Error Performance

With iterative decoding, (Berrou et al, 1993), the error performance improves. Typical of turbo coding schemes is the fact that increasing the number of iterations results in a monotonically decreasing improvement in coding gain. Increasing the number of iterations from 2 to 6 gives an improvement in SNR of 1.7 dB, whereas going from 6 to 18 iterations yields only a 0.3 dB improvement in coding gain. Since the appearance of turbo codes, advances have taken place in understanding the bit error rate (BER) behavior of turbo codes (Hokfelt et al., 2001). There appears to be a consensus among researchers on why turbo codes offer such an superior error performance, (1) Turbo codes have a weight distribution that approaches, for long interleavers, that of random codes, (2) Recursive convolutional encoders and proper interleaving map most of the low-weight information sequences into high-weight coded sequences, and (3) Systematic encoders allow the effective use of iterative decoding

techniques utilizing constituent SOVA decoders. Information symbol estimates are available directly from the channel.

4. Error Correction System

Error correction system, Fig. 4, provides a more detailed view of some subassemblies in Fig. 1, in Fig. 4. The processed data $\bar{d}_i$ can include errors $\bar{e}_i$ and the error correction system will subtract their estimates $\bar{e}_i'$ as indicated in the corrected data output of the error correction system. If one of the computed parity values, $\bar{p}_{u_i}$ or $\bar{p}_{l_i}$ in Fig. 4, comes from a failed subsystem, the error correction system's inputs may be incorrect. Since the data are correct under the single failed subsystem assumption, the data contain no errors and the error correction system is operating correctly. The error correction system will observe the errors in the syndromes and properly estimate them as limited to other positions. Moreover, an excessive number of error estimates $\{\bar{e}_i'\}$ could be deduct from correct data, yielding $\{\bar{d}_i - \bar{e}_i'\}$ values at the Error Correction System's output, which the regeneration of parity values produces $\{\bar{p}_{u_i}'\}$, as shown in Fig. 4 at the final output.

$$\bar{p}_{u_i}' = \sum_{j=0}^{l} p_j (\bar{d}_{i-j} - \bar{e}_{i-j}') \qquad (1)$$

Simultaneously, if the errors do not affect the parallel parity values $\{\bar{p}_{l_i}\}$, its value is correct.

$$\bar{p}_{l_i} = \sum_{k=0}^{l} p_k \bar{d}_{i-k} \qquad (2)$$

The output checking syndromes $\{\bar{s}_i'\}$ will become nonzero at the beginning of errors because of the burst-detecting nature of the code.

$$\bar{s}_i' = \bar{p}_{u_i}' - \bar{p}_{l_i} = \sum_{j=0}^{l} (-1) p_j \bar{e}_{i-j} \qquad (3)$$

Therefore, there are several indicators that will detect errors in the error correction system's input syndromes $\{\bar{s}_i\}$. The checking syndromes $\{\bar{s}_i'\}$ must indicate the beginning of errors, so the error correction system cannot subtract incorrect, even overwhelming errors from otherwise correct data without observation. The limited checking features inserted in and around the corrector will always detect its unsuitable behavior.

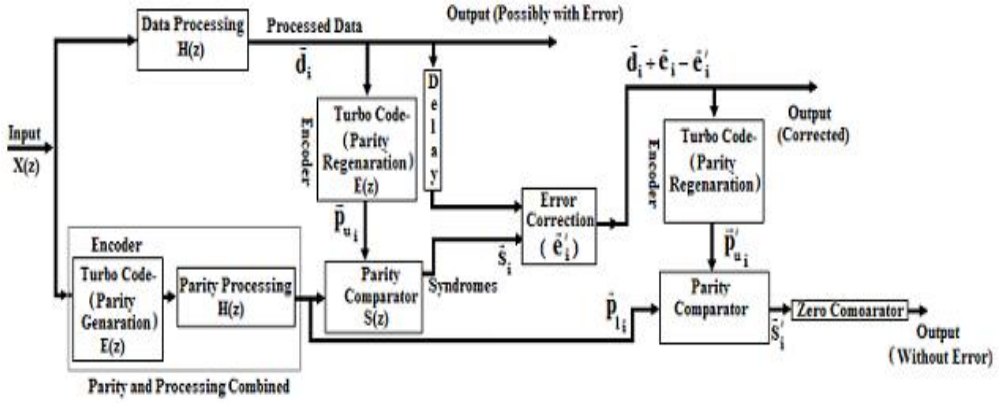

Figure 4: Block diagram of the ABFT technique along with error correction system

5. Results

It is an easy matter to construct MATLAB to implement the BP convolutional codes and turbo codes. Thus, a series of simulations provide appraise of the probability of detection and failure. Several simulation schemes modeling the ABFT method for detecting numerical level errors were described in MATLAB, version 2010a, where the modeling errors were assumed Gaussian with zero means and statistically independent from symbol to symbol. Errors were allowed in the parity values computed by the combined data parity generator, Fig. 1, and in the processed data symbols. Very researches were performed to verify the iterative decoding technique. The error modeling provides a strong set of conditions of failure effects and is completely general. An example of the G and H^T matrices for n = 4 is extracted after execution of these scripts. The encoding matrix G is $4\times(3.8)$ with its top three rows containing zeros and a 4×3 identity matrix in the rightmost three columns. The last row of G exhibit the additive identity as ,(-1,-1,-1,-1,-1,0,-1,0,-1,0,0,0,-1,0,0,0,-1,0,0,0,-1,0,0,0). The significant parity check part H^T, which dictates the parity values, is $1\times(4.8)$; (l+1) =$2n$ in this case. Its single row is, (1,0,1,0,1,1,0,0,1,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1).The simulation code randomly inserts a burst of errors in each block of input symbols, representing an encoded block. The choice of the burst is controlled by probability parameter ρ. Once an error situation is established in the simulation, n symbols representing a burst are determined using a uniform distribution and then are added to the n code symbols to model a burst. Many simulation steps were executed at various error rates ρ. For a high value of ρ, the error bursts are frequent enough that they may sometimes happen so close as to violate the protective band requirement for correction, leading to incorrect. A typical test used n = 7 with $\rho = 10^{-2}$ and employed 10^5 codeword blocks. The experimental ratio of bursts introduced in one run of 10^5 was 0.000101, and the experimental conditional probability of correction was 0.9832561, whereas that of failure was 0.0167439. When the probability of a burst was lowered to $\rho = 10^{-7}$, there were no failures for long runs.

6. Conclusions

There are many applications of ABFT. This paper provides a general method and techniques for employing turbo codes in ABFT, and bounds on the ABFT redundant computations are given. Turbo codes can be used efficiently for detecting the errors in numerical processing systems. This advantage of turbo codes over convolutional methods of coding is moderately typically over the complete range of code rates, that is, coding gain can be achieved at moderate BERs with long turbo codes of the same rate and approximately the same decoding complexity as convolutional codes.

References

Berrou,C. and Glavieux,A. and Thitimajshima,P (May 1993). “Near Shannon limit error-correcting coding and decoding: Turbo-codes,” in *Proceedings of the IEEE International Conference on Communications*, vol. 2, pp. 1064–1070.

Bosilca,G. and Delmas,R. and Dongarra , J. and Langou, J. (April 2009) " Algorithm-based fault tolerance applied to high performance computing,"Journal of Parallel and Distributed Computing, Elsevier, Vol.69, No.4, pp.410-416.

Chen,Z.(2008) "Extending Algorithm-based Fault Tolerance to Tolerate Fail-stop Failures in High Performance Distributed Environments," *Proceedings of the 22nd IEEE International Parallel & Distributed Processing Symposium, DPDNS'08 Workshop*, Miami, FL, USA, April 14-18.

El Gamal, H. and Hammons, A.R., Jr. (February 2001) "Analyzing the turbo decoder using the Gaussian approximation," *IEEE Transactions on Information Theory*, vol. 47, pp. 671–686.

Hagenauer ,J. and Hoher,F. (1989). "A Viterbi Algorithm with Soft-Decision Outputs and Its Applications, " *Proc. 1989 IEEE Global Teleconim Con& (GLOBECOM'89).* pp. 47.1.1- 47.1.7, Dallas, Texas .

Hagenauer, J. and Offer, E. and Papke,L.(1996) "Iterative decoding of binary block and convolutional codes, " IEEE Trans. Inform. Theory, vol.42, pages 429-445.

Hamidi, H. and Vafaei, A. and Monadjemi, A.H. (2009), Algorithm based fault tolerant and checkpointing for high performance computing systems, J.Applied Sci., 9:3947-3956.

Hokfelt, J. and Edfors,O. and Maseng,T. (May 2001) "On the theory and performance of trellis termination methods for turbo codes, " IEEE Journal on Selected Areas in Communications, vol. 19, pp. 838-847.

Huang, K. H. and Abraham, J. A, (1984), Algorithm-based fault tolerance for matrix operations, IEEE Transactions on Computers, vol. C-33:518-528.

Moosavie Nia, A.and Mohammadi, K. (2007)" A Generalized ABFT Technique Using a Fault Tolerant Neural Network," Journal of Circuits, Systems, and Computers 16(3): 337-356.

Proakis,J. G.(2001), " *Digital Communications*," 4th Edition. New York: McGraw Hill.

Redinbo, G.R. (Jun 1998) *"Generalized Algorithm-Based Fault Tolerance: Error Correction via Kalman Estimation",* IEEE Transactions ON Computers, vol. 47, no. 6.

Redinbo, G.R. (Nov. 2003) "Failure-Detecting Arithmetic Convolutional Codes and an Iterative Correcting Strategy, " *IEEE Transactions on Computers*, vol. 52, no. 11, pp. 1434-1442.

Redinbo, G.R. (July-Sept. 2010) "Wavelet Codes for Algorithm-Based Fault Tolerance Applications, " *IEEE Transactions on Dependable and Secure Computing*, vol. 7, no. 3, pp. 315-328.

Roche,T. and Cunche,M. and Roch, J.L (2009) . "Algorithm-Based Fault Tolerance Applied to P2P Computing Networks, " ap2ps, pp.144-149, 2009 First International Conference on Advances in P2P Systems.

Sadjadpour, H. and Sloane, N. and Salehi, M. and Nebe, G. (May 2001) " Interleaver design for turbo codes, " IEEE Journal on Selected Areas in Communications, vol. 19, pp. 831-837.

Snasel,V. and Platos,J. and Kromer,P. and Ouddane,N. (2008) "Genetic Algorithms Searching for Turbo Code Interleaver and Solving Linear Ordering Problem, " cisim, 2008 7th Computer Information Systems and Industrial Management Applications, , pp.71-77.

Sundaram, S. and Hadjicostis,C. N.(September 2008)"Fault-Tolerant Convolution via Chinese Remainder Codes Constructed from Non-Coprime Moduli,"IEEE Transactions on Signal Processing, Vol. 56, No. 9, pp. 4244-4254.

Wu, P. H.-Y. (May 2001)."On the complexity of turbo decoding algorithms," in Proceedings *of the IEEE Vehicular Technology Conference-Spring*, vol. 2, pp. 1439–1443.

Zhang, C.N. and Liu, X.W (June 2009). "An algorithm based mesh check-sum fault tolerant scheme for stream ciphers," International Journal of Communication Networks and Distributed Systems, Vol.3, No.3, pp.217-233.