Fault-Tolerant 2D Fourier Transform with Checksum Encoding

Grzegorz Cieslewski, Adam Jacobs, and Alan D. George
{cieslewski, jacobs, george}@hcs.ufl.edu

High-performance Computing and Simulation (HCS) Research Laboratory
Department of Electrical and Computer Engineering, University of Florida
Gainesville, Florida 32611-6200

Abstract—Space-based applications increasingly require more computational power to process large volumes of data and alleviate the downlink bottleneck. In addressing these demands, commercial-off-the-shelf (COTS) systems can serve a vital role in achieving performance requirements. However, these technologies are susceptible to radiation effects in the harsh environment of space. In order to effectively exploit high-performance COTS systems in future spacecraft, proper care must be taken with hardware and software architectures and algorithms that avoid or overcome the data errors that can lead to erroneous results. One of the more common kernels in space-based applications is the 2D fast Fourier transform (FFT). Many papers have investigated fault-tolerant FFT, but no algorithm has been devised that would allow for error correction without re-computation from original data. In this paper, we present a new method of applying algorithm-based fault tolerance (ABFT) concepts to the 2D-FFT that will not only allow for error detection but also error correction within memory-constrained systems as well as ensure coherence of the data after the computation. To further improve reliability of this ABFT approach, we propose use of a checksum encoding scheme that addresses issues related to numerical precision and overflow. The performance of the fault-tolerant 2D-FFT will be presented and featured as part of a dependable range Doppler processor, which is a subcomponent of synthetic-aperture radar algorithms. This work is supported by the Dependable Multiprocessor project at Honeywell and the University of Florida, one of the experiments in the Space Technology 8 (ST-8) mission of NASA's New Millennium Program.[12]

TABLE OF CONTENTS

1

[2] IEEEAC paper #1426, Version 2, Updated October 31, 2006

1. INTRODUCTION

Spaceborne computer components lack the protection of the Earth's atmosphere and are constantly bombarded with various types of radiation. High-energy charged particles and ionizing electromagnetic waves are much more widespread in the emptiness of space and can cause critical errors in the unprotected digital and analog parts of satellites and other space vehicles. To protect against such problems, the majority of space-certified parts are radiation hardened, which significantly improves their resilience to both transient and permanent faults. Unfortunately, those custom specialized parts are more expensive and exhibit lower computational capability than current commercial-off-the-shelf (COTS) systems.

To meet constantly increasing computational requirements for space platforms, the shift to the COTS components has been proposed [1]. Such unprotected computers require proper care to be taken through software architecture to mitigate problems caused by soft errors. Results of the algorithms that are executed on such systems must be validated either by executing the application multiple times (spatially or temporally) or by applying algorithm-based fault tolerance (ABFT). The ABFT approach was originally developed for error detection and correction on systolic arrays [2], but the concepts can be used on any sequential machine. This method is extremely useful for space-based computing [3] as it generally has lower overhead than the more traditional N-modular redundancy (NMR) approaches.

One of the most common kernels in space-related applications is the 2D discrete Fourier transform (DFT) and its specialized implementation, the fast Fourier transform (FFT). Because of the widespread usage of processing in frequency domain, it is necessary to develop methods to detect and correct errors in the computation of 2D-DFT to ensure correct operation of a space-based system in hazardous environments. Many researchers have studied fault-tolerant 1D-FFT networks but, to the best of our knowledge, no efficient algorithm has been developed that would allow for error detection and correction that could be generalized to a multi-dimensional domain. Development of a dependable 2D-DFT kernel will enable use of other applications in space-based environments.

One such application is synthetic aperture radar (SAR), a common technique used for obtaining high-resolution images of the Earth's topography as well as for detecting and tracking ground and underground targets. This imaging method was first introduced in 1978 when NASA launched the first satellite equipped with a SAR sensor. More recently, a mapping mission to Mars has been proposed that would use SAR imaging to create detailed images of the surface [4]. The key computational subcomponent of SAR is the range Doppler processor (RDP) which is closely related to the 2D-DFT algorithm.

In this paper we propose a new method of applying ABFT concepts to the 2D-DFT algorithm and RDP. The proposed method will not only allow for error detection but also for error correction using redundant data. This method is well suited to memory-constrained systems where data fills most of the system memory and does not allow out of place operations.

In a satellite system, the overall throughput is dependent on the downlink capability. To alleviate the bandwidth bottleneck, efficient on-board processing of SAR data requires specialized hardware or a powerful supercomputer system. The Dependable Multiprocessor (DM) project aims to achieve reliable computation in space with the use of COTS technologies in order to provide a small, low-power, supercomputer in space [1]. The method proposed is well-suited for such a system, which not only supports conventional processing elements but also high-speed FPGA co-processors, which are more vulnerable to the harsh radiation environment.

The remaining sections of this paper are organized as follows. Section 2 surveys previous work relating to ABFT and fault-tolerant DFT as well as provides basic concepts for the rest of the paper. The details of the new encoding scheme for 2D-DFT are described in Section 3 followed by a description of the fault-tolerant SAR algorithm in Section 4. While Section 5 analytically compares the overhead of current methods to the ones developed, Section 6 presents experimental results. Section 7 presents conclusions and provides directions for future research.

2. RELATED WORK

Algorithm-Based Fault Tolerance

ABFT refers to a technique that can detect and/or correct errors that occur during the computation of certain linear algebra kernels. The method relies on augmenting the matrices in question with extra rows or columns containing weighted checksums that will be preserved through mathematical operations [2, 3, 5, 6]. In turn, these checksums can be then used to analyze the correctness of the result and, if an error was detected, initiate a recovery procedure.

To obtain the weighted checksums, the initial data will have to be multiplied by an encoder matrix. Without loss of generality and to simplify the notation, we assume that generic matrix is square with dimensions of N × N.

Definition 1: An encoder matrix is a matrix whose product with the data matrix will yield the desired checksums. For the remainder of this paper we will refer to the encoder matrix as E_N. The E_N used in this paper will have dimensions of $N \times 2$.

Definition 2: A column checksum matrix A_C is an initial data matrix A that has been augmented with extra rows of checksums. Such a matrix will have dimensions of $(N+2) \times N$ and has the form:

$$A_c = \begin{bmatrix} A \\ E_N^T \cdot A \end{bmatrix} \quad (1)$$

A column checksum matrix can also be obtained by multiplying a matrix that is already augmented by another matrix B of compatible size.

$$A_C \cdot B = \begin{bmatrix} A \\ E_N^T \cdot A \end{bmatrix} \cdot B = \begin{bmatrix} A \cdot B \\ E_N^T \cdot (A \cdot B) \end{bmatrix} \quad (2)$$

The associative property of the matrix product allows for verification of the multiplication procedure by simply recalculating the checksums and comparing them with ones obtained through the matrix multiply. In general, an operation that preserves weighted checksums is called checksum-preserving and the matrix product is an example of such function.

Definition 3: A row checksum matrix A_R is an initial data matrix that has been augmented with extra columns of checksums. Such a matrix will have dimensions of $N \times (N+2)$ and has the form:

$$A_R = \begin{bmatrix} A & A \cdot E_N \end{bmatrix} \quad (3)$$

The transpose operation is also a checksum-preserving operation. Transpose of the row checksum matrix will yield the column checksum matrix and vice versa as shown in Eq. (4). This property allows for verifying the correctness of the transpose operation.

$$A_R^T = \begin{bmatrix} A & A \cdot E_N \end{bmatrix}^T = \begin{bmatrix} A^T \\ (E_N^T \cdot A^T) \end{bmatrix} = B_c \quad (4)$$

There have been extensive studies that investigate different encoding matrices (also known as weight vectors) with respect to numerical and error detection/correction

properties [5]. The focus of this paper is the new encoding scheme for data and for that purpose we will use one of the previously investigated weight vectors in a new way. Our approach can also be used with different sets of weight vectors as long as they allow for the similar type of encoding.

Discrete Fourier Transform

The basic form of the discrete Fourier transform of the vector x can be expressed in terms of the matrix product operation

$$\vec{X} = \vec{x} \cdot F_N \tag{5}$$

where x is the input row vector, X is the output row vector, and F_N is defined as the DFT matrix of the form:

$$F_N = \begin{bmatrix} W^0 & W^0 & W^0 & \cdots & W^0 \\ W^0 & W^1 & W^2 & \cdots & W^{(n-1)} \\ W^0 & W^2 & W^4 & \cdots & W^{2(n-1)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ W^0 & W^{(n-1)} & W^{2(n-1)} & \cdots & W^{(n-1)(n-1)} \end{bmatrix} \tag{6}$$

$$\text{where} \quad W^q = e^{-j(2\pi \cdot q/N)} \tag{7}$$

The key property of the DFT is linearity, which implies that DFT of the weighted sum of two or more signals is equal to the sum of the weighted DFTs of each signal. Eq. (8) demonstrates that property in matrix form where x and y are row vectors and a and b are arbitrary constants.

$$(a\vec{x} + b\vec{y}) \cdot F_N = a \cdot (\vec{x} \cdot F_N) + b \cdot (\vec{y} \cdot F_N) \tag{8}$$

The FFT is an implementation of the DFT that removes the redundant computation from calculating the Fourier transform. The method presented in this paper is independent of the implementation of the DFT.

Fault-tolerant Fourier Transform

Many schemes have been presented for creating fault-tolerant FFT networks. Choi and Malek have developed an approach based on recalculation of the transform [7]. The key drawback to such an approach is that it automatically incurs a large computational overhead when implemented on a sequential machine. Reddy and Banerjee have proposed an alternative approach that exploits Parseval's theorem and compares the power of the input of the transform to the power of the output [8]. An advantage of such a scheme is that it can be applied not only to one-dimensional transforms, but also to a multi-dimensional one. The method has low computational complexity, but the only way to correct the error is through full recomputation. It also suffers higher rates of false-positive errors than other methods. In order to address problems with previous approaches, an ABFT-inspired method, known as concurrent error detection (CED), has been developed. Many variants of the CED form of the fault-tolerant Fourier transform are described in [2, 9, 16] and have been developed by using the ABFT concepts presented to formulate an efficient way to verify the calculation of 1D Fourier transforms.

The general idea behind the CED scheme is to pre-compute encoding v_e and decoding v_d row vectors that will make the following equation hold

$$\vec{v}_e \cdot \vec{x}^T = \vec{v}_d \cdot \vec{X}^T \tag{9}$$

where x and X are defined as in (5). Such an approach can achieve a low overhead and fewer false-positives, but requires the computation of new coding vector pairs for each size of DFT.

Because of the round-off error present in floating-point arithmetic, a small tolerance must be allowed during the comparison so as to prevent false positives. This small tolerance level also reduces error coverage of the fault-tolerant method and some errors might go undetected. Those errors do not usually have a large impact on the application as they occur in digits of low numerical significance. The numerical precision issues are beyond the scope of this paper and a small tolerance will be chosen based on the data used for the experiments.

3. FAULT-TOLERANT 2-D FOURIER TRANSFORM

The 2D Fourier transform is one of the most common kernels used in image processing. An important property of the 2D-DFT is that it can be divided into two sets of 1D transforms separated by a transpose operation. The current approaches for providing fault detection for the 2D transform use a simple power-comparison scheme or the CED implementation as described in the previous section. The CED scheme is applied by separating the 2D transform into multiple 1D kernels. However, such an approach does not protect the data stored in memory or during the corner-turn operation. Furthermore, upon the detection of the error, the recovery strategy is to recompute the 1D transform again, which requires additional space and memory operations as the Fourier transform has to be performed out of place.

Partial 2D Fourier Transform

The 2D-DFT can be divided into four steps as shown in the Figure 1. Steps 1-2 and 3-4 are identical with respect to the actual computation performed. This property can be leveraged by designing a more general kernel that later can be duplicated to obtain the full transform. Let us define a partial transform kernel as a Fourier transform along the desired dimension. Such a partial transform can be

expressed as (10) where A is an input matrix and B is an output matrix.

$$B = A \cdot F_N \tag{10}$$

The new encoding scheme focuses on making the partial transform fault-tolerant. The full fault-tolerant 2D-DFT is performed by using two fault-tolerant partial transforms and two transposes.

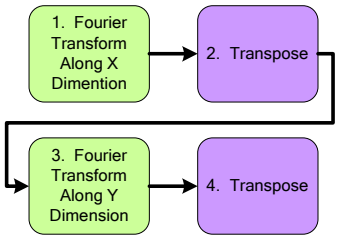

Figure 1. 2D-DFT Computation Flow

Error Model

During the matrix multiplication, all resulting entries are computationally independent of each other. This fact means that if an error occurs during a computation then only one resulting value would be corrupted. If one were to use the DFT algorithm as specified in Eq. (5), such assumptions would hold. If the error were to occur in the FFT algorithm, it could manifest as multiple incorrect values on the output since the FFT uses an Omega-type network [17] for computation of the final result. Depending upon which complex butterfly the error occurs, it could result in a varying number of errors within the output vector. In the worst-case scenario, an error would manifest as corruption of all the values in the result.

Let us assume that only one error will occur during the computation of the 2D-DFT. As a result only one row of the matrix could get corrupted at any point in time since the computations on each row within the partial transform are independent of each other. If the error occurs during the first partial transform then it will be detected and corrected before data is fed to the second partial transform. Similarly, if the error is introduced during the second partial transform then it will be corrected before computation is finished.

Encoding Scheme for Partial Transform

To perform fault-tolerant partial transforms, an encoding scheme is required that will allow the correction of, at most, one error per column of the matrix. A column checksum matrix encoded with the encoder matrix in Eq. (11) addresses that requirement by employing a SECDED (Single Error Correcting Double Error Detecting) code.

$$E_N^T = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 3 & \cdots & N \end{bmatrix} \tag{11}$$

The encoding proposed is a form of a Hamming code and it is closely related to one developed in [2, 5]. Since the partial transform is the matrix product of the input matrix and the DFT matrix, we can conclude based on Eqs. (1) and (2) that the partial transform is checksum-preserving.

The process of performing the fault-tolerant partial transform is shown in Figure 2 and described by Eqs. (12) and (13). The input matrix A *is* augmented to form column checksum matrix A_C, which is in turn transformed through matrix product to form B_C.

$$B_C = \left(A_C \cdot F_N\right) \tag{12}$$

$$B_C = \begin{bmatrix} A \\ E_N^T \cdot A \end{bmatrix} \cdot F_N = \begin{bmatrix} A \cdot F_N \\ E_N^T \cdot \left(A \cdot F_N\right) \end{bmatrix} \tag{13}$$

The alternative way of showing that the partial transform preserves the weighted checksums is to view the input matrix as a series of vectors, the checksum vectors as the linear combinations of the input vectors, and the partial transform as the series if 1D-DFTs to be performed on the input vectors. The linearity property of the DFT transform as described in Eq. (8) proves that the partial transform is checksum-preserving.

Figure 2. Fault-tolerant Partial Transform

Error Detection and Recovery

Error detection and correction is performed on a column basis only. The Hamming encoding proposed allows for the detection of up to two errors or the correction of one error per column.

Let $y^T = (y_0, y_1, y_2 \ldots y_{N-1}, cs_1, cs_2)$ be a single column of the matrix B_C. In order to validate the calculation we have to recalculate the checksums and validate them. Let the two syndromes be defined as follows:

$$S_1 = \left(\sum_{k=0}^{N-1} y_k\right) - cs_1 \tag{14}$$

$$S_2 = \left(\sum_{k=0}^{N-1} (k+1) \cdot y_k\right) - cs_2 \tag{15}$$

Then the detection and correction procedure can be described by the following:

- If $S_1=S_2=0$ then there is no error.
- If $S_1\neq 0$ and $S_2=0$ then the error occurred in checksum S_1. The new checksum can be recomputed based on the data in the column.
- If $S_1=0$ and $S_2\neq 0$ then the error occurred in checksum S_2. The new checksum can be recomputed based on the data in the column.
- If $S_1\neq 0$ and $S_2\neq 0$ then the error is in the data. The ratio of the two checksums will allow us to localize the error. If t is defined as in Eq. (16) then the error has occurred in y_t.

$$t = \frac{S_2}{S_1} - 1 \tag{16}$$

There are two methods of correcting the error. The first method involves subtracting S_1 from y_t and is commonly used in fixed-point problems. However, this approach may not yield good results in floating-point arithmetic when the magnitude of the error is much larger than the original value of y_t. A method that will yield better results can be described as follows:

$$y_t = cs_1 - \left(\sum_{k=0}^{t-1} y_k\right) + \left(\sum_{k=t+1}^{N-1} y_k\right) \tag{17}$$

The approach in Eq. (17) calculates the values of y_t based on the difference of cs_1 and the remaining data in the column.

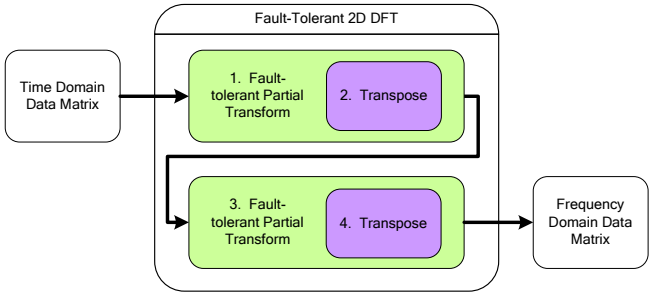

Figure 3. Computation Flow of Fault-tolerant 2D-DFT

Computational Process

Figure 3 shows the process that must be applied to compute the fault-tolerant 2D-DFT. The first step involves fault-tolerant partial transform followed by a transpose step. To protect the data during the transpose operation the transpose can be taken before the results of the partial transform are verified. The transpose operation preserves the checksum encoding by turning the column check-summed matrix into row check-summed one as shown in Eq. (4). Steps 3 and 4 are equivalent to steps 1 and 2, and are performed in the same manner.

4. Synthetic Aperture Radar

Range Doppler Processor

The algorithms for processing SAR images have been well studied. The most common one is the RDP [10-14] and variations thereof. In order to correctly focus the SAR data, filtering in the frequency domain is performed. The coefficients of the filters are based on the radar frequency and various other parameters that are dependent on the orbit of the satellite. It is possible to resolve the SAR data in the time domain through 2D convolution but such implementations would require a time-domain-specific filter for each range cell [11]. The resulting algorithm would be computationally inefficient, and as such, is not used. The RDP takes advantage of the properties of the SAR data which make the range and azimuth data nearly orthogonal to each other. This approach allows for independent resolution of the range and azimuth data by performing filtering in the frequency domain. The key steps for the RDP algorithm are presented in Figure 4.

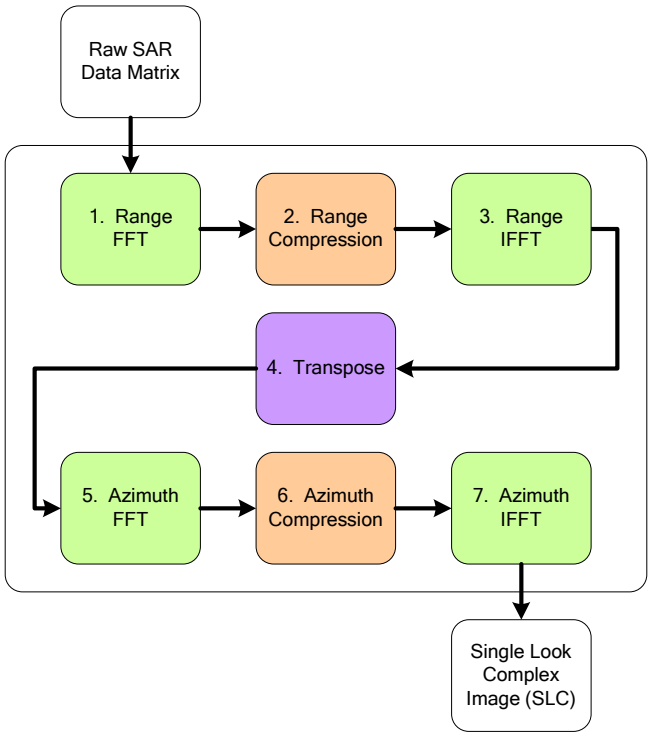

Figure 4. A Typical RDP Algorithm

The datasets collected by the SAR-enabled satellites are very large and could range from a million data points to a few billion. A typical data set collected by the ESR satellite is 5616 × 28000 complex points [15].

The RDP algorithm is very similar to the 2D-DFT in that that it uses partial transform and inverse partial transform to perform the bulk of the processing. There are two distinct phases to the RDP processing algorithm, range filtering (steps 1-3) and azimuth filtering (steps 5-7), which are separated by a transpose operation step (4). Both phases involve identical computational steps: partial transform, followed by the modified Hadamard product, and inverse partial transform. The modified Hadamard product can be defined as follows:

$$B = A \otimes \vec{x} \qquad (18)$$

where $\vec{x}$ is a row vector of length N, A and B are matrices of size $N \times N$, and the modified Hadamard product is denoted by $\otimes$. Hadamard product is also called element-wise matrix vector multiplication. Each element of matrix B in Eq. (18) is given by $b_{ij} = a_{ij} + x_j$

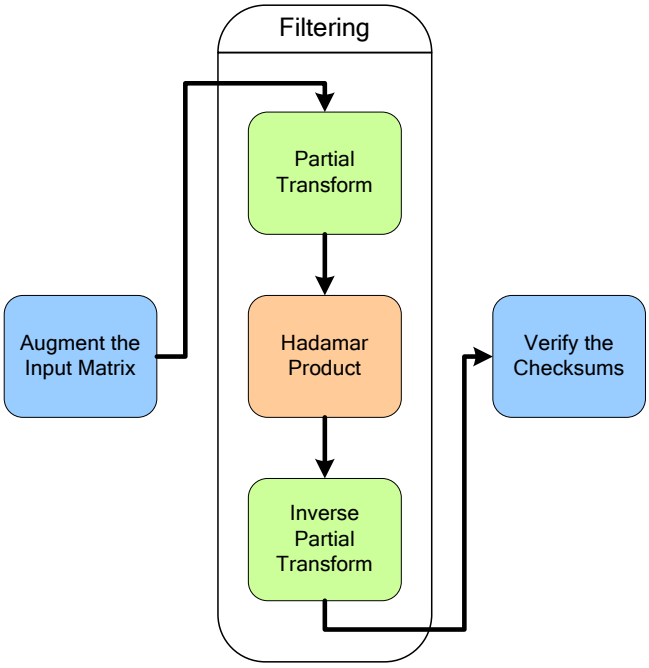

Figure 5. Fault-tolerant Filtering in Frequency Domain

Fault-tolerant Range Doppler Processor

Fault tolerance for the RDP is a relatively new topic because technologies allowing such high computational power have only recently become available to be used in space and, as such, not studied thoroughly. Fang, Le, and Taft [14] have investigated the use of FPGA-based systems for RDP that employ TMR for fault protection. On sequential machines, such an approach would incur overhead of over 200%.

As shown in Figures 1 and 4, the phases of 2D DFT and RDP involve some identical steps. Therefore, we could use a similar approach as was used in the design of the fault-tolerant 2D DFT algorithm to design a fault-tolerant RDP algorithm. We augment the input data matrix with additional checksums, perform the range or azimuth filtering, and then verify the checksums. This process is shown in Figure 5.

Let A_C be the augmented input data using the encoder matrix specified in Eq. (11). As shown in Eq. (13) we can safely perform the partial transform on an augmented matrix with the checksums being preserved. Using the same arguments it stands to reason that the inverse partial transform would also preserve the checksums. The particular definition of the modified Hadamard product is also checksum-preserving as shown in Eq. (19),

$$\left(A_C \otimes \vec{x}\right) = \begin{bmatrix} A \otimes \vec{x} \\ \left(E_N^T \cdot A\right) \otimes \vec{x} \end{bmatrix} = \begin{bmatrix} A \otimes \vec{x} \\ E_N^T \cdot \left(A \otimes \vec{x}\right) \end{bmatrix} \qquad (19)$$

where x is a row vector containing filter coefficients. Looking at Eq. (19) from a different perspective, each column in the matrix A is scaled by the same constant and, since scaling by a constant does not affect the validity of the checksums, the operation above is checksum-preserving.

With the introduction of a fault-tolerant frequency domain filtering scheme that can be used to perform range and azimuth compression, we can now design a fault-tolerant RDP. As shown in Figure 6, the dependable RDP algorithm consists of two filtering steps separated by the transpose operation. In order to protect the data during the transpose operation we can include the transpose step in the first filtering stage. Such an operation would yield a row checksum matrix as shown in Eq. (3).

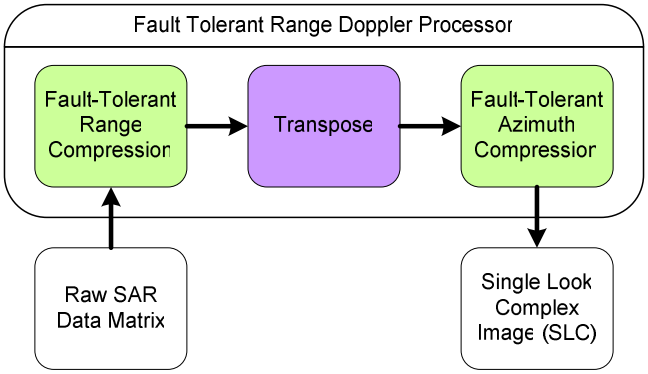

Figure 6. Fault-tolerant RDP

5. Overhead Analysis

Overhead for Fault-tolerant 2D-FFT

In Table 1, we present the overhead comparison of our fault-tolerant 2D-FFT cited as CE (checksum encoding). We compare it against the power-based method [8], the implementation using the efficient fault-tolerant 1D-FFT CED scheme as presented in [9], and the self-checking pair (SCP) approach. The SCP is a type of NMR method with two members. The three most crucial parameters under

examination are the overhead during fault-free execution, the upper bound on overhead when errors are detected, and the memory requirement. The overhead in Table 1 is derived based on the number of extra computations needed as compared to the conventional, non-fault-tolerant 2D FFT algorithm.

According to [12, 13] the majority (over 85%) of RDP computation is focused in the FFTs, IFFTs and Hadamard product. The other computations needed include matched filter generation. For the purpose of the experiments, the generation of matched filters is ignored to focus on the most computationally demanding portions of the algorithm.

Table 1. Overhead Comparison of Selected 2D-FFT Methods

Size N × N	Overhead [%]							
	No Error				Worst Case with Error			
	MR	PW	CD	CE	MR	PW	CD	CE
64	100	27	34	70	200	153	34	86
128	100	23	29	59	200	146	29	73
256	100	20	25	51	200	140	25	63
512	100	18	22	45	200	136	22	56
1024	100	16	20	40	200	132	20	50
2048	100	15	18	36	200	129	18	46
4096	100	13	17	33	200	127	17	42
8192	100	12	15	31	200	125	15	38
16384	100	11	14	29	200	123	14	36
32768	100	11	13	27	200	121	13	33
65536	100	10	13	25	200	120	13	31
MR – Self-checking pair NMP scheme								
PW – Power-based method [8]								
CD – CED-based scheme [9]								
CE – Featured scheme with checksum encoding								

The detailed complexities of the algorithms presented are shown in the Table 2. All of the computations are assumed to be performed on complex numbers. As such, a complex multiply requires six operations and a complex add requires two. Also, the linear and constant terms were dropped from the equations as they are strongly implementation-dependent and do not influence the overhead significantly for large values of N.

Table 2. Computational and Memory Complexities of Selected 2D-FFT Methods

Function	Operations Count	Memory Requirement
2D-FFT	$10\,N^2\,log_2(N)$	N^2
MR	$20\,N^2\,log_2(N)$	$3\,N^2$
PW	$10\,N^2\,log_2(N)+16\,N^2$	$2\,N^2$
CD	$10\,N^2\,log_2(N)+20\,N^2$	$N^2 + N$
CE	$10\,N^2\,log_2(N)+40\,N^2+$ $+20\,N\,log_2(N)$	$N^2 + N$

The four schemes presented do not provide the same fault protection for the data and as such the comparison requires more insight. The NMR method has the highest level of data protection but it also has the highest memory and computational overhead. Consequently, it could be used on systems where the processing power and memory are plentiful. The power-based scheme shows the lowest computational overhead but incurs a tremendous penalty when an error occurs. Additionally, the power scheme has high spatial complexity requiring the transform to be out of place. The CED-based scheme has low computational and memory overhead but requires each 1D-FFT to be performed out of place and therefore requires a large number of memory operations. Furthermore, it also does not protect data stored in memory or during the transpose operations. The featured scheme has slightly higher computational and memory overhead than the CED scheme but it addresses all of the shortcomings of the other schemes. For large data sets it is characterized by low computational overhead of 25% and low memory footprint. It does not require extra memory operations for out-of-place transforms and the 7% extra overhead required to correct errors is also relatively low. Most importantly, the scheme proposed protects the data at all times, starting when the checksums are first computed through both partial transforms and both transpose operations and ending when data leaves the system.

Table 3. Computational Complexity of RDP

Function	Operations
RDP	$20\,N^2\,log_2(N) + 12\,N^2$
RDP with CED	$20\,N^2\,log_2(N) + 68N^2$
RDP with CE	$20\,N^2\,log_2(N) + 52N^2 + 40\,N\,log_2(N)$

Overhead for Fault-tolerant RDP

In Tables 3 and 4 we compare the overhead of the featured fault-tolerant RDP with CE versus other methods. The TMR approach guarantees high penalties on sequential systems but it is easily parallelizable as demonstrated in [14]. The specialized system developed allows for high throughput at the expense of area and power. An alternative approach to the problem would involve using the CED scheme for the 1D-FFTs and the SCP scheme for the Hadamard multiplication. This technique would provide reliable calculations but would not protect data from being corrupted in memory or during the transpose. Our method not only protects data during the transpose but throughout the entire RDP algorithm. In addition, this novel method shows the lowest error-free overhead of the algorithms studied.

Table 4 shows that the overhead for fault-free execution of our CE approach would be only 16% for images of 64k × 64k pixels, which is the best of all three methods. For the same size image, the worst-case scenario, a whole erroneous row incurs only 23% penalty, which is slightly higher then the CED scheme. The errors occur relatively infrequently to the time required to perform the RDP calculations, which makes the featured algorithm more attractive as the common case has 5% lower overhead.

Table 4. Overhead Comparison of Fault-tolerant RDP's

Size N × N	Overhead [%]					
	No Error			Worst Case with Error		
	TMR	CED	CE	TMR	CED	CE
64	200	57	47	200	58	63
128	200	49	39	200	49	53
256	200	43	33	200	43	46
512	200	38	29	200	38	40
1024	200	34	26	200	34	36
2048	200	31	24	200	31	33
4096	200	28	22	200	28	30
8192	200	26	20	200	26	28
16384	200	24	19	200	24	26
32768	200	23	17	200	23	24
65536	200	21	16	200	21	23
TMR – Triple modular redundancy approach [14]						
CED – Scheme with CED [4] and SCP						
CE – Featured scheme with checksum encoding						

6. Experimental Results

The fault-tolerant 2D-FFT and the RDP applications have been implemented on the Dependable Multiprocessor (DM) testbed. The DM cluster computer is a prototype for the DM flight system that is being developed by researchers at Honeywell and the University of Florida.

The DM testbed is configured with two Orion CPC7510 and four Orion CPC7520 single-board computers in a CompactPCI chassis interconnected by a pair of independent Gigabit Ethernet networks. The six PowerPC nodes are configured as one System Controller, four Data Processors, and one Mass Data Store node. The Data Processor nodes contain 1GB of system memory each and are augmented with ADM-XRC-4 FPGA coprocessors which can be used for accelerated processing. The testing was performed on a single Data Processor node without the assistance of the FPGA. The algorithms have been implemented using freely available libraries. The FFTW library was used for the FFT kernel, and for matrix operations necessary for the checksum calculations the GSL and BLAS libraries were used.

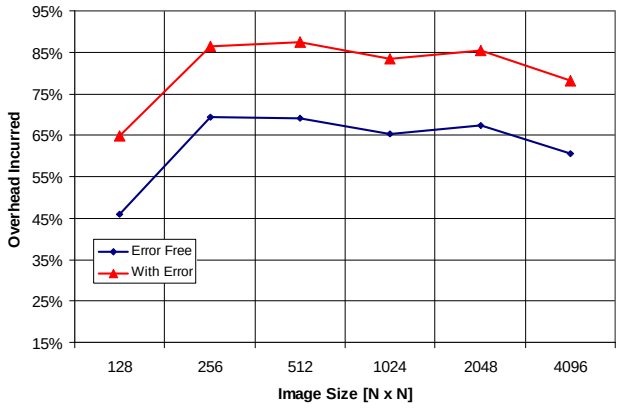

Figure 7. Experimental Overhead of Fault-tolerant 2D FFT Algorithm vs. a Fault-intolerant Version

Figure 7 shows the overhead of the fault-tolerant 2D-FFT as executed on the DM testbed. Due to the memory constraints of the system the maximum possible data set is a 4096 × 4096 matrix filled with double-precision, floating-point, complex values. The figure shows two plots, one depicting overhead in an error-free environment and the second representing the worst-case overhead incurred due to one error.

The graphs show a modest increase in overhead when changing the size of the input matrices from 128 × 128 to 512 × 512 that may seem counter-intuitive. This behavior is related to the cache size of the processor used as well as the large linear and constant complexity terms that are part of the implementation. As long as the whole data matrix can fit in the cache, the calculation and validation of the checksums is greatly expedited.

For data sizes above 512 × 512, the data shows a decrease in overhead as predicted in the theoretical analysis of the algorithm. The theoretical analysis shows the lower bound on the overhead for the large data sizes where the high complexity terms dominate.

A similar experiment has been performed for the RDP algorithm as shown in Figure 8.

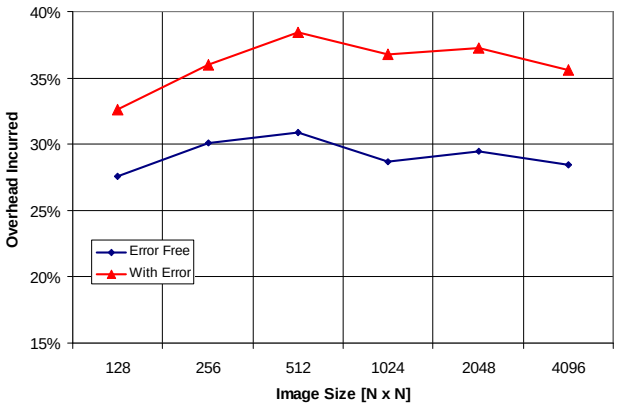

Figure 8. Experimental Overhead of Fault-tolerant RDP vs. a Fault-intolerant Version

The initial increase of the overhead is similar as in the 2D-FFT algorithm and can also be attributed to the cache size and the large linear and constant coefficients.

Both graphs show low overhead for the execution of the fault-tolerant kernels as compared to SCP techniques which would automatically incur 100% overhead since each data element is calculated twice. For the largest data size the error-free overhead for our fault-tolerant 2D-FFT is 61% and for RDP algorithm is only 28%. In the worst-case scenario where the whole row is corrupted, the overhead can increase 10%-20% over the fault-free case. Note that upsets are relatively infrequent occurrences and that overall

performance of the algorithm will be similar to that of the error-free case.

7. Conclusions and Future Work

In this paper we presented a novel way of performing fault-tolerant 2D-DFT and SAR algorithms, which are commonly used kernels in space science. Traditional ABFT concepts are used and extended to allow the error correction and detection of single errors that occur during computation. We have shown theoretically and experimentally that both fault-tolerant kernels have low overhead. For large datasets the proposed fault-tolerant 2D-FFT algorithm has only 25% overhead and requires little additional memory. The experimental results on the smaller data sets show an overhead of 61%, which is lower than methods using NMR techniques. The proposed fault-tolerant SAR algorithm has shown an impressively low theoretical overhead of only 16% and experimentally only 28%.

In comparison to other fault-tolerant algorithms for the 2D-FFT and the SAR algorithms, our approach shows excellent characteristics missing from others. Comparatively, the proposed kernels have very low computational overhead and memory footprints. They do not require any out-of-place 2D or 1D transforms. Most importantly, they protect the coherence of data not only during the computation but also during memory operations.

The experiments were performed on double-precision, floating-point numbers. Due to rounding issues, a further study is required to analyze the numeric properties of the methods proposed. Such a study would include the proper selection of small tolerance values during comparisons, and precision of the corrected values.

The structure of the fast Fourier transform lends itself to efficient hardware implementation. It is possible to use low-cost reconfigurable devices (FPGAs) to implement a hardware FFT engine with commodity parts. Using FPGAs, it is possible to achieve over 5× speedup compared to current high-end microprocessors and over 20× speedup on processors for future space missions, such as the Dependable Multiprocessor [1]. One of the drawbacks to using these reconfigurable devices is that they are significantly more susceptible to transient errors than microprocessors. To mitigate that disadvantage, the TMR system has been proposed [13]. While providing necessary levels of fault-tolerance, the TMR method incurs a high power penalty due to use of three FPGAs. Use of the ABFT approach could dramatically reduce the power consumption while maintaining the appropriate level of fault tolerance.

Acknowledgments

This work was supported in part by the NMP Program at NASA, our Dependable Multiprocessor project partners at Honeywell Inc., and the Florida High-Technology Corridor Council. The authors would like to thank all members of the HCS lab for their support in writing this paper.

REFERENCES

[1] J. Samson, J. Ramos, I. Troxel, R. Subramaniyan, A. Jacobs, J. Greco, G. Cieslewski, J. Curreri, M. Fischer, A. George, V. Aggarwal, M. Patel, and R. Some, "High-Performance, Dependable Multiprocessor," *Proc. of IEEE/AIAA Aerospace*, Big Sky, MT, Mar 4-11, 2006

[2] Juang-Hua Huang and J. A. Abraham, "Algorithm-Based Fault Tolerance For Matrix Operations," IEEE Transactions on Computers, vol. C-33, no. 6, pp. 518-528, Jun 1984.

[3] M. Trumon, R. Granat, and D. Katz, "Software-Implemented Fault Detection for High-Performance Space Applications," Proceedings International Conference on Dependable Systems and Networks (DSN), pp.107-116, New York, NY, Jun 25-28, 2000.

[4] T. W. Thompson, J. J. Plaut, R. E. Arvidson, P. Paillou, "Orbital Synthetic Aperture Radar (SAR) for Mars Post Sample Return Exploration," 31st Annual Lunar and Planetary Science Conference, Houston, Texas, March 13-17, 2000.

[5] V. S. S. Nair and J. A. Abraham, "Real-Number Codes for Fault-Tolerant Matrix Operations On Processor Arrays" IEEE Transactions on Computers, vol. 39, no. 4, pp. 426-435, Apr 1990.

[6] Jing-Yang Jou and J. A. Abraham, "Fault-tolerant matrix arithmetic and signal processing on highly concurrent computing structures," Proceedings of the IEEE, vol. 74, no. 5, pp. 732-741, May 1986.

[7] Y. H. Choi and M. Malek, "A fault-tolerant FFT processor," IEEE Transactions Computing, vol. C-37, no. 5, pp. 617-621, May 1988.

[8] A. L. Narasimha Reddy and P. Banerjee, "Algorithm-Based Fault Detection for Signal Processing Applications" IEEE Transactions on Computers, vol. 39, no. 10, pp. 1304-1308, Oct 1990.

[9] Sying-Jyan Wang and N. K. Jha, "Algorithm-Based Fault Tolerance for FFT Networks," IEEE Transactions on Computers, vol. 43, no. 7, pp. 849-854, Oct 1994.

[10] A. Hein, "Processing of SAR data: fundamentals, signal processing, interferometry", Berlin: Springer-Verlag, 2004

[11] R. Raney, H. Runge, R. Balmer, I. Cumming, and F. Wong, "Precision SAR Processing Using Chirp Scaling," IEEE Trans. on GeoScience and Remote Sensing, 32(4), Jul 1994, pp. 786-799.

[12] P. G. Meisl, M. R. Ito, I. G. Cumming, "Parallel Synthetic Aperture Radar Processing on Workstation Networks," Proc. of 10th International Parallel Processing Symp. (IPPS), Washington, DC, Apr 15-19, 1996.

[13] J. Greco, G. Cieslewski, A. Jacobs, I. A. Troxel, A. D. George, "Hardware/software Interface for High-performance Space Computing with FPGA Coprocessors", *Proc. of IEEE/AIAA Aerospace*, Big Sky, MT, Mar 4-11, 2006

[14] W. Fang, C. Le, S. Taft, "On-board fault-tolerant SAR processor for spaceborne imaging radar systems" IEEE International Symposium on Circuits and Systems (ISCAS), Kobe, Japan, May 23-26, 2005

[15] D. T. Sandwell, "SAR IMAGE FORMATION: ERS SAR PROCESSOR CODED IN MATLAB," http://topex.ucsd.edu/insar/sar_image_formation.pdf, 2002.

[16] D. L. Tao and C. R. P. Hartmann, "A Novel Concurrent Error Detection Scheme for FFT Networks," IEEE Transactions on Parallel and Distributed Systems, vol. 4, no. 2, pp. 198-221, Feb 1993.

[17] J. W. Cooley and J. W. Tukey, "An algorithm for machine calculation of complex Fourier series," vol. 19, pp. 297-301, 1965

BIOGRAPHIES

Grzegorz Cieslewski is a graduate student at the University of Florida where he is currently pursuing a Ph.D. degree in Electrical and Computer Engineering. As a research assistant he is a member of the Advanced Space Computing and Reconfigurable Computing groups at the High-performance Computing & Simulation Research Laboratory. His research interests include computer architecture, reconfigurable, fault-tolerant and distributed-computing as applied to linear algebra and signal processing problems. He is a student member of IEEE.

Adam Jacobs is a Ph.D. student in Electrical and Computer Engineering at the University of Florida. He is a research assistant in the Advanced Space Computing and Reconfigurable Computing groups at the High-Performance Computing and Simulation Research Laboratory. His research interests include fault-tolerant FPGA architectures and high-performance computing. He is a student member of the IEEE.

Alan D. George is Professor of Electrical and Computer Engineering at the University of Florida, where he serves as Director of the HCS Research Lab and Director of the new NSF Center for High-performance Reconfigurable Computing (CHREC). He received the B.S. degree in Computer Science and the M.S. in Electrical and Computer Engineering from the University of Central Florida, and the Ph.D. in Computer Science from the Florida State University. Dr. George's research interests focus upon high-performance architectures, networks, services, and systems for parallel, reconfigurable, distributed, and fault-tolerant computing. He is a senior member of IEEE and SCS, a member of ACM and AIAA, and can be reached by e-mail at *ageorge@ufl.edu*.