The Dissertation Committee for Shih-Hsin Hu
certifies that this is the approved version of the following dissertation:

Concurrent Error Detection in 2-D Separable Linear Transform

Committee:

Jacob A. Abraham, Supervisor

Nur Touba

Earl E. Swartzlander, Jr.

Michael Orshansky

Chen He

Concurrent Error Detection in 2-D Separable Linear Transform

by

Shih-Hsin Hu, B.S.E.; M.S. Elec. & Comp. Eng.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2016

Dedicated to my beloved family

Acknowledgments

I am thankful to have the opportunity to study part-time as part of UT Austin's Ph.D. program while working full-time in the semiconductor industry. After years of work, I have greatly enjoyed the moments I have spent on campus and doing research. Balancing family, work, and school is a difficult challenge and dilemma. It is not an easy path, and there have been many times I have wanted to give up. In looking back, as I come to the end of my journey toward a doctorate, there are many people without whose support and encouragement this quest would not have been a success.

First and foremost, I would like to thank my two advisors, Prof. Jacob A. Abraham and Prof. Margarida F. Jacome. Jacob gave me the opportunity to be a part of his research team and guided me throughout my dissertation work. With his broad and profound knowledge in computer engineering, he helped me transform many raw ideas into concrete research results. Jacob gave me the independence to do things my way, which has given me a lot of confidence. He was always there with the right suggestion whenever I hit a roadblock. Margarida was my first advisor at UT. She showed me the beauty of the research world. She was always dedicated and supportive to her other students and me, even until the last days of her life. Her students and colleagues will always remember her fondly.

I would like to thank professors Prof. Nur Touba, Prof. Earl Swartzlander, Prof. Michael Orshansky and Dr. Chen He for serving on my Ph.D. committee. Their constructive criticism helped improve the quality of this dissertation. I would also like to thank all members of the Computer Engineering Research Center at the University of Texas at Austin for making my graduate education a very enriching experience. I would like to thank for Qualcomm Technologies Inc. and Freescale Semiconductor to provide me financial support during the study.

Finally, I would like to thank my beloved wife Mei-Ling for her patience, understanding, and love.

Concurrent Error Detection in 2-D Separable Linear Transform

Publication No. ______________

Shih-Hsin Hu, Ph.D.
The University of Texas at Austin, 2016

Supervisor: Jacob A. Abraham

As process technology continues to scale to smaller geometries and reduces the supply voltage, reliability of the resulting semiconductor becomes a greater concern. The effect of deep submicron noise, soft errors, variation, and aging degradation pose challenges on the functional correctness of VLSI systems and places roadblocks on reductions in scale. On the other side, as computing moves toward mobile, the energy efficiency of digital systems becomes one of the most important design metrics. However, reliability and energy efficiency are contradicting design requirements. Adding a voltage guard band is the most common method to mitigate the reliability impacts in such instances. Low power design technique like voltage over-scaling (VOS) even reduces the power by scaling the supply voltage just before data-dependant timing errors start to appear.

Concurrent error detection is the solution to tackle reliability and energy-efficiency in a unified manner. Fault tolerance can be deployed at different design hierarchies. Given its low overhead, algorithm level error detection is an attractive approach. In this work, a generic weighted checksum code based error detection algorithm targeted generic 2-D separable linear transform is proposed. This technique encodes the input array at the 2-D linear transformation level, and algorithms are designed to operate on encoded data and produce encoded output data. The proposed error detection technique is a system-level method and therefore can be used in existing hardware or software 2-D linear transformation architectures with low overhead. The mathematic proof of the algorithm is provided within the scope of this dissertation. The checksum weighting vector for several common transforms are derived as examples, error detection cost and algorithm effectiveness are analyzed.

In traditional fault tolerance study, the error is often evaluated at the boolean level. Many DSP applications, like 2-D linear transformation used in the multimedia compression system, do not require exactly correct results, but rather that the quality of the output is within the acceptable range. A generic quality aware error detection in the 2-D separable linear transform is proposed by extending the above property and defining the errors at the functional level. As an example, the quality-aware error detection technique is deployed on a low-power wavelet lifting transform architecture in JPEG2000. A low-cost Signal to Noise Ratio (SNR) aware detection logic based on proposed scheme is integrated into the discrete wavelet lifting transform architecture.

This detection logic checks whether the image quality degradation caused by voltage over-scaling induced timing errors is acceptable and determines the optimal voltage set point in operating conditions at run time. This novel quality-based error detection approach is significantly different from traditional error detection schemes which look for exact data equivalence. A simulation result for one design shows that the supply voltage can be scaled down to 75% of the nominal voltage in typical process corner without significant image quality degradation, which translates to $9.15mW$ power consumption (44% power saving).

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Reliability Challenge in Emerging Digital VLSI Systems

The rapid scaling of CMOS process technology has resulted in significant improvement in the performance and transistor density of digital electronic devices. However, feature size and supply voltage reductions make current and future VLSI systems less and less reliable. Systematic and random variations in process, supply voltage and temperature pose a major challenge for the design of future digital VLSI systems and change the design problem from deterministic to probabilistic [1] [2].

The significant reliability challenges to scaling CMOS technology are outlined in the following.

1.1.1 Deep Submicron Noise

Deep submicron (DSM) noise [3] [4] is defined as any disturbance in device node voltage or current away from a nominal value causing intermittent or permanent logic failure. The common mechanisms for such failures are increased path delay as well as accidental charge/discharge of dynamic nodes. Noise sources that have substantial impacts on the performance of digital circuits include crosstalk [5], IR drop [6], ground bounce [7], charge sharing [8], process variations [9] and charge leakage [10]. These problems worsen as technology scales further. Mezhiba's research [11] has shown both resistive and inductive noise are increasing with technology scaling. The emergence of DSM noise is making it increasingly difficult to achieve the desired level of noise immunity while maintaining the historic improvement trends in performance and energy-efficiency of integrated circuits.

1.1.2 Soft Error

In addition to DSM noise, soft errors (single event upsets) are another source of concern [12]. These errors are caused by alpha particles from contaminated packages and cosmic rays (energetic neutrons and protons) hitting silicon chips, creating a charge on the nodes that flips a memory cell or logic latch. Soft errors are transient and random. For memory, there is a relatively easy way to detect these errors with parity checking and correct these errors with the error-correcting code [13]. However, if a single event upset occurs in a logic flip-flop, it is much more difficult to detect and rectify the error. Shivakumar's research [14] has shown that soft error rates (SER) per chip of logic circuits will increase significantly due to technology scaling and superpipelined designs. The reduction in critical charge of logic circuits with decreased feature size makes the circuit more susceptible to external radiation.

1.1.3 Static and Dynamic Variations

Variations have an impact even on today's VLSI design and it is expected to get worse as technology scales. Static variations caused by random dopant fluctuations and sub-wavelength lithography result in wider distribution of transistor threshold voltages. In the case of test escape, static variations could result into system errors in the field [2]. Dynamic variations such as temperature variation and supply voltage variation (due to IR drop and di/dt noise) is time and context variant [15].

Thermal distribution across the die depends on the application and workload. Hotspots put more demand on power distribution networks and result in dynamic supply voltage variations [2]. Both the device and interconnect performance have temperature dependence, with hot temperature causing interconnect performance degradation and cold temperature causing device performance degradation [16]. Additionally, temperature variation across communicating blocks on the same chip may cause performance mismatches, which may lead to logic or functional failures.

Variations in switching activity across the die and diversity of the type of logic, result in uneven power dissipation across the die. This variation results in uneven supply voltage distribution and temperature hot spots, across a die, causing transistor subthreshold leakage variation across the die. Packaging and platform technologies do not follow the scaling trends of CMOS process. Therefore, power delivery impedance does not scale with supply voltage and voltage variation has become a significant percentage of the supply voltage [1].

Infrequent worst-case voltage droop could cause timing path failing and impact reliability and stability [17].

1.1.4 Aging Degradation

As technology scales and device dimension shrink, the trend in the V_t variability at both time zero and after NBTI (Negative Bias Temperature Instability) aging shows an increase [18]. Random defects in the gate oxide can cause aging-induced device delta V_t variability [19]. The electric fields and current have increased continuously as a result of scaling and have reached the maximum values that can be allowed for reliable CMOS operation. Increasing power density leads to higher chip temperature and consequently an even faster acceleration of chip degradation mechanisms [20]. Due to accelerated aging in future technology nodes, early wear-out effects could occur during the product life cycle and jeopardize the success of the functionality or safety of the system.

1.1.5 Reliability and Power Trade-Off

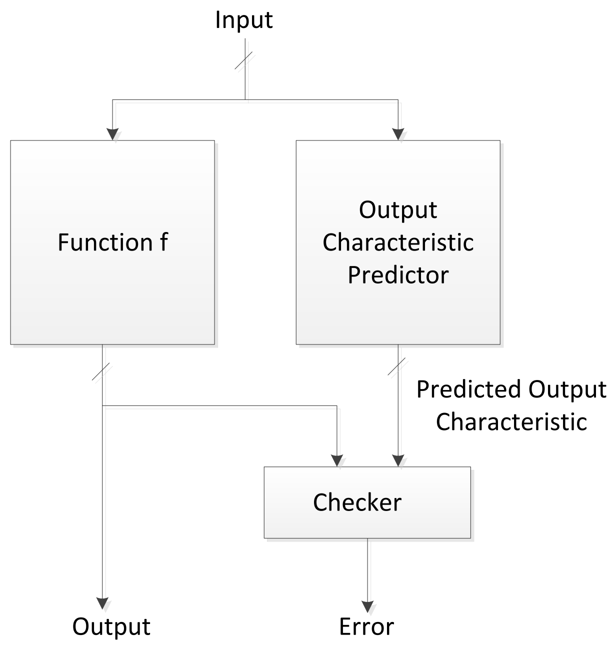
Energy-efficient VLSI design is of great interest given the proliferation of mobile and pervasive computing devices. Reduced power consumption not only leads to longer battery life but also results in reduced packaging cost, increased reliability, lower cost of ownership and longer lifetime for VLSI devices. However, there is a fundamental trade-off between energy-efficiency and reliability. In traditional design practices, the DSM noise, variation, and aging degradation issues are handled by adding a safety margin or so-called “guard band" to the supply voltage or timing requirement [21]. Although the impact can be alleviated by raising the supply voltage at the expense of maximum product frequency and yield, supply voltage increase has a direct impact on power consumption. The necessity of ensuring correct operation even under infrequent worst-case conditions results in clock frequency or supply voltage guard bands that degrade performance and increase energy consumption. Low power design technique like voltage over-scaling (VOS) reduces power by scaling the supply voltage just before data-dependent timing errors start to appear [22]. Design for reliability also becomes mandatory for reducing power dissipation. Voltage reduction strongly affects reliability by reducing noise margins and thus the sensitivity to soft errors, and by increasing circuit delays and thus the severity of timing faults.

1.2 Overview of Concurrent Error Detection

Concurrent error detection (CED) is the detection of errors or faults in a circuit or data path concurrent with normal operation of the circuit [23].

Suppose the system under test realizes a function f and produces output $f(i)$ in response to an input sequence i. A CED scheme generally contains another unit which independently predicts some special characteristic of the system output $f(i)$ (checker symbol) for every input sequence i. A checker unit checks whether the special characteristics of the output actually produced by the system in response to input sequence i is the same as the one predicted. An error signal is generated when a mismatch occurs. The CED scheme aims to preserve the data integrity which means that the system either produces correct outputs or indicates erroneous situations when incorrect outputs are produced. This property is also referred to as the fault-secure property [24]. CED can be deployed in digital systems to tackle the reliability challenges encountered in technology scaling and provide a solution to achieve desired reliability requirements. Figure 1.1 shows the general architecture of a general CED scheme.

Figure 1.1: General architecture of a concurrent error detection scheme


Input
Function f
Output Characteristic Predictor
Predicted Output Characteristic
Checker
Output
Error

1.2.1 Redundancy

Fault tolerance is an exercise in exploiting and managing redundancy. As the failures happen, redundancy is used to mask or work around these failures, thus maintaining the desired level of functionality. Hardware faults are usually dealt with by using hardware, time and information redundancy [25]. Software redundancy is used mainly against software failures which are outside the scope of the dissertation.

1.2.1.1 Hardware Redundancy

Hardware redundancy is provided by incorporating extra hardware into the design to either detect or override the effects of a failed component. The simple form of the hardware redundancy is dual modular redundancy (DMR) for error detection and triple modular redundancy (TMR) for error correction. DMR uses a second copy of the circuit under operation as output characteristic prediction and compares the results of the two circuits. TMR adds a third copy of the circuit under operation to achieve error correction function through voting [26].

1.2.1.2 Time Redundancy

Time redundancy attempts to reduce the amount of extra hardware required for fault tolerance at the expense of additional time. Transient faults can be detected by repeating the computation several times. Alternate-data-retry [27] and Re-computation with shifted operands [28] are examples of time redundancy. Although the hardware cost of time redundancy is generally less than hardware redundancy, it also has a direct impact on system performance.

1.2.1.3 Information Redundancy

Information redundancy is provided by adding information (error detecting codes and error correction codes) to data to tolerant faults. Information redundancy is widely used in memory and communication application, for example, parity codes and Hamming codes [13]. The arithmetic code can be used to protect arithmetic functions. For checking arithmetic operations, data is encoded before the operation. Code words are checked after the operation. Two common types of arithmetic codes are AN codes and residue codes [29]. AN code is formed by multiplying each data work N by some constant A. AN codes are invariant to addition and subtraction. If no error occurred, the output can be evenly divisible by A. Residue codes are created by computing a residue for data and appending it to the data. The residue is generated by dividing a data by an integer, called a modulus. Decoding is done by simply removing the residue. Residue codes are invariants with respect to addition.

1.2.2 Fault Tolerance at Different Design Level

Redundancy can be inserted at different design levels (circuit, architecture, and algorithm) for fault tolerance and error detection.

1.2.2.1 Circuit Level

Razor [30] [31] employs error-tolerant dynamic voltage scaling (DVS) technology which dynamically detects and corrects circuit timing errors to permit design optimizations in typical circuit operational levels instead of worst case corner. Razor eliminates the need for the voltage margins required for "always correct" circuit operations. At the circuit level, a shadow latch controlled by a delayed version of the clock augments each delay-critical flip-flop. An error is detected whenever there is a mismatch between the main flip-flop and shadow latch. Since the shadow latch always contains the correct value, it can re-execute an instruction failure in one pipeline stage through the following stage, while incurring a one-cycle penalty.

However, the error detection is done at the circuit level which requires significant area and power overhead. In large designs, the Razor shadow latch cannot be used if complex control signals are on the critical path. The Razor flip-flop's hold time is much larger than that of a conventional flip-flop. It consumes area and energy to lengthen the short timing paths. How to choose representative timing paths within area budget to be monitored is another open challenge due to timing paths redistribution in post-silicon. Depending on the process corner, voltage, temperature variations, and also workload, different timing paths which are not shown as critical in pre-silicon timing analysis might become critical [32]. Razor leverages the existing timing path skew within the design. If the timing paths within the design are well balanced, the Razor method cannot be applied. Also, Razor checks for exact data equivalence while the requirement can be relaxed for most digital signal processing application as long as the signal-to-noise ratio (SNR) is above the threshold. Razor generally requires an additional pipeline stage to enable recovery after error detection, which will require a modification of the architecture.

1.2.2.2 Architecture Level

To take the advantage of chip multiprocessor (CMP) for architecture level fault tolerance, the redundant execution approach executes copies of the same program on two independent threads and compares the results [33]. Unused processor cores can be used to run redundant threads as full utilization of cores is not usually feasible. Dynamic core coupling (DCC) [34] allows arbitrary processors cores to verify each other in a DMR setup to avoid static binding overhead. Chip-level redundant threading (CRT) [35] extends redundant multi-threading techniques for single simultaneous multithreading (SMT) to CMP. EDDI (Error Detection by Duplicate Instruction) [36] and SWIFT (Software implemented fault tolerance) [37] are software-based redundant techniques that provide low-cost alternatives to hardware-based redundant methods. Dynamic verification [38] is an error detection technique that operates at runtime and uses dedicated hardware checkers to verify the validity of specific invariants assumed to be true in error-free operation. The key point in a dynamic verification approach is to define a comprehensive set of invariants.

1.2.2.3 Algorithm Level

To reduce the overhead incurred by redundancy, deploying the redundancy at algorithm level is an attractive approach. However, algorithm level fault tolerance is often application specific and needs to develop different protection scheme for the different algorithms.

Algorithm-based fault tolerance (ABFT) is an effective error detection and correction technique for matrix operation [39]. ABFT achieves fault tolerance at the algorithm level rather than providing hardware level protection.

ANT (Algorithmic Noise Tolerance) [40] was one of the earliest proposals to leverage the inherent error resilience of algorithms to achieve energy efficiency and tolerance to deep submicro noise. Subsequent efforts [41] [42] [43] [44] proposed variants of the ANT approach and demonstrated significant energy benefits for DSP algorithms. Different algorithm may need different type of the ANT. However, the area overhead of ANT is significant in some cases.

Significance Driven Computation is introduced in Roy's work [45] [46] [47] [48]. The concept of significance driven computation is to identify the important computation and make sure these computations are executed correctly across all process corners in the early design phase. It allows the non-important computation to fail. The method involves no error detection or error correction. The algorithm and micro-architecture techniques are employed to avoid the timing errors on critical computation. The method has been deployed on color interpolation filtering, FIR filter, discrete cosine transform (DCT), and motion estimator applications. The limitation of this approach is that each different DSP application requires a different algorithm to identify and isolate the important computation, and some applications do not have a known solution. Also, the method can only handle the timing errors but not soft errors.

1.2.3 Fault Tolerance of Approximate Computing

Approximate computing [49] [50] is an emerging design paradigm and broadly refers to a class of design techniques that leverage intrinsic application resilience to design more efficient (better power/performance/area) computing platforms. Today, approximate computing is predominantly proposed for multimedia and signal processing applications that have a certain degree of inherent error tolerance. However, a gap exists in extending the technique to other compute-intensive tasks in science and engineering. To close the gap, it requires that the allowed error or the required minimum precision of the application is either known beforehand or reliably determined online to deliver trustworthy and useful results. Errors outside the allowed range have to be reliably detected and tackled by appropriate fault tolerance measurements.

Although approximate computing looks promising, there exist challenges to deploying the technique in the actual product. There is a need to develop fault tolerance techniques of approximate compute algorithms [51]. Approximate circuits need to be protected by online testing and concurrent error detection to cover the entire lifecycle of the application. The error detection scheme must be developed with appropriate metrics and characterization procedures to access whether the circuit is either critical, marginal or non-critical. The quality aware error detection scheme is essential to explore the benefit of the approximate computing. To reduce the area and performance overhead, an algorithm level quality aware error detection scheme is attractive. However, the technique is naturally algorithm-specific.

1.3 Research Motivation

Putting it all together, the current and future VLSI chip will have tens of billions of transistors but many of them might be unusable because of extreme static variation. In addition, circuits will encounter dynamic variations of supply voltage and temperature, frequent and intermittent soft errors and slow performance degradation over time due to aging. As technology scales into the deep submicron regime, reliability is becoming a metric of comparable importance to power, performance and area (PPA) for the analysis and design of VLSI digital systems.

Since 2001, the International Technology Roadmap for Semiconductor (ITRS) [52] has stated the reliability and energy-efficiency as two of the critical design technology challenges. These technology trends show the strong demand for techniques to design reliable and energy-efficient digital systems. Designing energy-efficient VLSI systems in the presence of above variation and error sources is a challenging research problem since it calls the need to tackle the issues of energy reduction and reliable operation in a unified manner.

The challenge can be attained only through a new design paradigm to achieve system reliability as opposed to component reliability. Despite these difficulties introduced by technology scaling, the chips cannot be retested at the factory after product shipment, thus users expect the system to remain reliable and to continue to deliver the required performance. The behavior of reliability failure mechanisms is becoming more stochastic/random, voltage/temperature/workload dependent and widely distributed in time. The need for concurrent error detection and on-line testing becomes more and more important. Low cost and power efficient fault tolerant schemes going beyond double/triple redundancy are needed. This research will focus on algorithm level concurrent error detection to improve reliability and energy-efficiency in digital system design.

1.4 Overview

The dissertation is organized as follows. In this chapter 1, an overview of the reliability challenge in deep sub-micro technology is provided and the source of errors and variations in emerging digital VLSI systems is summarized. The concurrent error detection overview is presented. The concept of redundancy (hardware, time, and information) and fault tolerance at different design levels (circuit, architecture, and algorithm) are introduced.

The trade-off between reliability and power is discussed. Power consumption and system stability often have contradictory requirements. Jointly considering reliability and energy efficiency in VLSI design, calls for the need to develop low cost, concurrent error detection scheme.

Chapter 2 presents the generic 2-D separable linear transform error detection algorithm. Mathematical proof is provided to demonstrate the proposed algorithm. Several common 2-D linear transforms are studied as examples and the checksum weighting vector associated with them are derived. The error detection overhead and effectiveness are studied and compared with prior work. Multiple faults and common mode failures of the proposed algorithm is analyzed and showed the effectiveness of the proposed method.

Chapter 3 presents the novel concept of error modeling. The errors at functional level instead of boolean level are defined and a generic quality aware error detection in the 2-D separable linear transform is proposed. The quality-aware error detection technique is deployed on a low power implementation of 2-D Discrete Wavelet Transform (DWT) application via voltage over-scaling (VOS). An SNR-aware error detection scheme built on top of the proposed algorithm is used as the quality sensor to guide adaptive voltage scaling based on output image quality.

Chapter 4 summarizes and concludes the dissertation with discussions on future work in this research direction.

Chapter 2

Concurrent Error Detection Algorithm in 2-D Separable Linear Transform

This chapter describes the proposed concurrent error detection algorithm for 2-D separable linear transform in details. Mathematical proof of the algorithm is provided. The error detection capability is validated via Matlab simulation.

2.1 Background

2-D separable linear transformation is widely used in multimedia and digital signal processing. 2-D signals such as images are usually partitioned into square blocks of $N \times N$ samples where N is the width of the square block. In this work, 2-D linear transforms are considered, which maps a 2-D input vector X, into a 2-D output vector Y. The attention is restricted to invertible transforms in this work.

Definition 1.0: A 1-D unitary transform is defined by matrix T which maps N samples, possibly complex-valued, into N transform coefficient [53].

$$y = Tx \tag{2.1}$$

where y and x are $N \times 1$ data vectors and T is $N \times N$ matrix. The coefficients in y are often visualized as being in a frequency domain. Any normalizing factors are incorporated in the transform matrix T and its inverse T^{-1}. An important characteristic of a unitary matrix connects the inverse T^{-1} to the Hermitian transpose of T, written as T^*.

$$T^{-1} = T^* \tag{2.2}$$

$$x = T^* y \tag{2.3}$$

□

Definition 1.1: Let an $N \times N$ 2-D input vector X and an $N \times N$ 2-D output vector Y be denoted by

$$X = [x(m,n)], 0 \leq m, n \leq N-1 \tag{2.4}$$

$$Y = [y(k,l)], 0 \leq k, l \leq N-1 \tag{2.5}$$

The forward and inverse 2-D linear transforms are defined as

$$y(k,l) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} w(k,l;m,n) x(m,n) \tag{2.6}$$

$$x(m,n) = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} v(m,n;k,l) y(k,l) \tag{2.7}$$

where $0 \leqq k, l, m, n \leqq N-1$. And $w(.)$ and $v(.)$ are the forward and inverse transform kernels.

□

In most practical multimedia applications, the 2-D transform kernels are separable and symmetric. Therefore, the 2-D transform kernel can be expressed as the product of two 1-D unitary, orthogonal or bi-orthogonal basis functions. If the 1-D transform operator is denoted by M , the forward and inverse transformations can be expressed in matrix form as

$$Y = M^* X M^T \tag{2.8}$$

$$X = M^T Y M^* \tag{2.9}$$

The above formulations show that the image transformation can be done in two stages: by taking the transformation M of each row of the 2-D input vector, and then by applying transformation M^* to each column of the intermediate result.

Discrete Fourier Transform (DFT), Discrete Hartley Transform (DHT), Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT) are few examples of 2-D linear separable transforms.

2.2 Related Work

2.2.1 Algorithm-Based Fault Tolerance

Algorithm-based Fault Tolerance (ABFT) [39] [54] is a system level method to provide fault detection and diagnosis through data redundancy. The data redundancy is implemented at algorithm level. ABFT techniques have been developed for matrix-based and signal processing applications such as matrix multiplication, matrix inversion, LU decomposition and the Fast

Fourier Transform [55]. Data redundancy in matrix operations is implemented using a checksum code. Given an $m \times n$ matrix X, the column checksum matrix X_C is defined as

$$X_C = \begin{bmatrix} X \\ eX \end{bmatrix} \tag{2.10}$$

where $e = [1\ 1 \cdots 1]$ is a row vector containing m 1s. The elements in the last row of X_C are the checksums of the corresponding columns of X. Similarly, the row checksum matrix X_R is defined as

$$X_R = \begin{bmatrix} X & Xf \end{bmatrix} \tag{2.11}$$

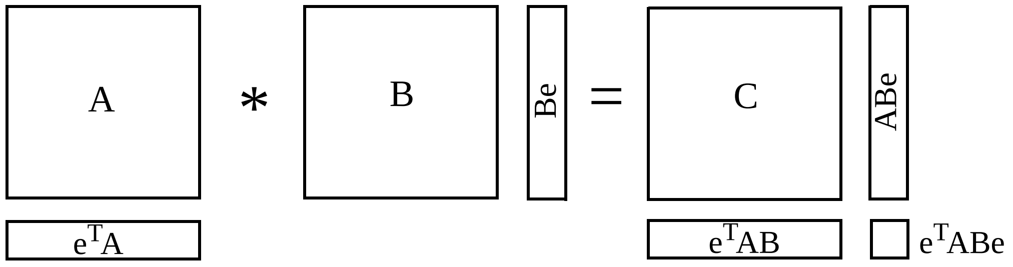
where $f = [1\,1 \cdots 1]^T$ is a column vector contains n 1s. Finally, the full $(m+1) \times (n+1)$ checksum matrix X_F is defined as

$$X_F = \begin{bmatrix} X & Xf \\ eX & eXf \end{bmatrix} \tag{2.12}$$

The column or row checksum matrix can be used to detect a single fault in any row or column of X, respectively, whereas the full checksum matrix can be used to locate a single erroneous element of X. If the computed checksums are accurate, locating the erroneous element allows error correction as well. The above column, row, and full checksums can be used to detect or correct errors in various matrix operations. For example, the matrix addition $A+B = C$ can be replaced by $A_C + B_C = C_C$ or $A_R + B_R = C_R$ or $A_F + B_F = C_F$.

Similarly, instead of calculating $AB = C$, the arithmetic can be computed as $AB_R = C_R$ or $A_C B = C_C$ or $A_C B_R = C_F$. Figure 2.1 shows the graphical representation of ABFT encoding for matrix multiplication protection.

Figure 2.1: Graphical representation of ABFT encoding for matrix multiplication protection



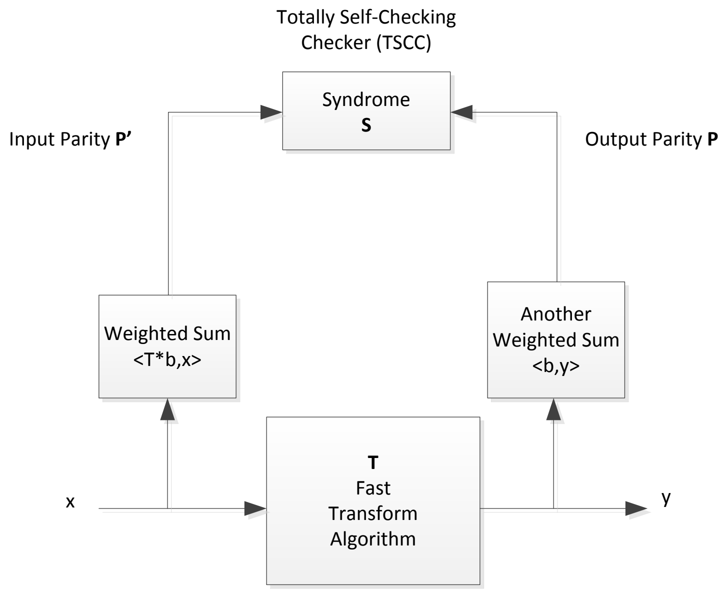
Weighted checksum code based ABFT [54] extends the error correction capability to matrix-vector multiplication, LU-decomposition, and matrix inversion. Another row/column weighted checksum is generated with weighting factor $e_w = [1\ 2 \cdots 2^{m-1}]$ and $f_w = [1\ 2 \cdots 2^{n-1}]$.

2.2.2 Parity Check Error Detection

A basic approach to fault tolerance in transform algorithms employs error detection based on comparing two parity values, one computed by forming a weighted sum of the transform coefficients and the other one form a comparable weighted sum over the input data. By comparing these related parity values, errors are detected and the complete transform can be recomputed. Figure 2.2 shows the error detection scheme overview.

The input parity P' can be described using a weighting vector d and

Figure 2.2: Protection of Fast Unitary Transform Implementations


Totally Self-Checking Checker (TSCC)
Syndrome S
Input Parity P'
Output Parity P
Weighted Sum <T*b,x>
Another Weighted Sum <b,y>
T Fast Transform Algorithm
x
y

the inner product applied to the input vector x

$$P' =< d, x >= d * x \tag{2.13}$$

The output parity P can be described using a weighting vector b and the inner product applied to the output transform vector y

$$P =< b, y >= b * y \tag{2.14}$$

$$P =< b, Tx >=< T * b, x >= b * Tx \tag{2.15}$$

In a fault-free situation, the input parity equals output parity $P = P'$, therefore

$$d = T * b \tag{2.16}$$

The error detection is achieved by comparing P and P' in a totally self-checking checker (TSCC) [56]. The checker forms the syndrome S, the difference between the two parties, and determines if its magnitude is small which below a chosen threshold indicating acceptable round-off tolerance.

Fault tolerance technique for FFT algorithms has been discussed in many papers. Jou [57] first proposed a concurrent error detection method for FFT networks which utilizes a coding relationship in the FFT computations to achieve fault-secure results and to distinguish round-off errors and functional errors. Fault location is accomplished using a time-redundancy method. The method can be used to detect a single error in either the multiplier or input/output set of lines. Tao's work [58] enhanced fault coverage of FFT networks error detection and provided a detailed round-off error analysis. Oh's work [59] [60] proposed linear weighting factors at FFT network level by leveraging FFT algorithm property. The scheme supports one-dimensional and multidimensional FFT. Reddy [61] proposed SOS (Sum of Square) system-level checking scheme based on Parseval's theorem and deployed the checks on FFT. Parseval's theorem can be expressed as $N\sum_{i=0}^{N-1}X^2(i) = \sum_{i=0}^{N-1}Y^2(i)$. Although the method has less hardware complexity than previous work, however, the fault model is weaker and thus the fault coverage can be lower than others. Wang [62] implemented weighted checksum scheme for FFT network based on algorithm based fault tolerant. The design is similar to Oh's work [59] with a more simplified output checksum calculation. Above work can be summarized at the theoretical form of the parity check scheme depicted in Figure 2.2. Although there are many researches utilizing weighted checksum code for FFT error detection, the works have focused on protecting against a single high-level error appearing on a single line between stages. The error detection scheme and weighting factor are often tied to specific fast algorithm architecture. The prior error detection technique does not appear to be transferable to different transforms.

2.2.3 Fault Tolerance for Generic Linear Transformation

Redinbo [63] proposed a generic concurrent error detection method for fast unitary transform based on parity weighted sum which utilizes iterative code design methodology (Figure 2.3) to come up specific parity for each transform application.

For each error ε to be detected, based on Equation 2.13 and 2.15, syndrome S need to be not equal to zero for each individual error in each stage and line.

$$S = P - P' = b * (y + \varepsilon) - b * Tx = b * \varepsilon \neq 0 \tag{2.17}$$

The method requires the computer program to create error gain matrix and output error patterns for errors in each matrix factorization stage and line. The error detection linear functional is formed based on error patterns matrix. The algorithm starts with initial weighting vector and computes error gain vector iteratively to check if all gains are non-zero and weightings are non-zero. If the condition is not meet, it will modify the weightings and retry until the weighting vector meets the requirement. Although the method seems generic, the code design process is heuristic and iterative. A poor choice for a parity weighting vector can lead to poor detection capability.

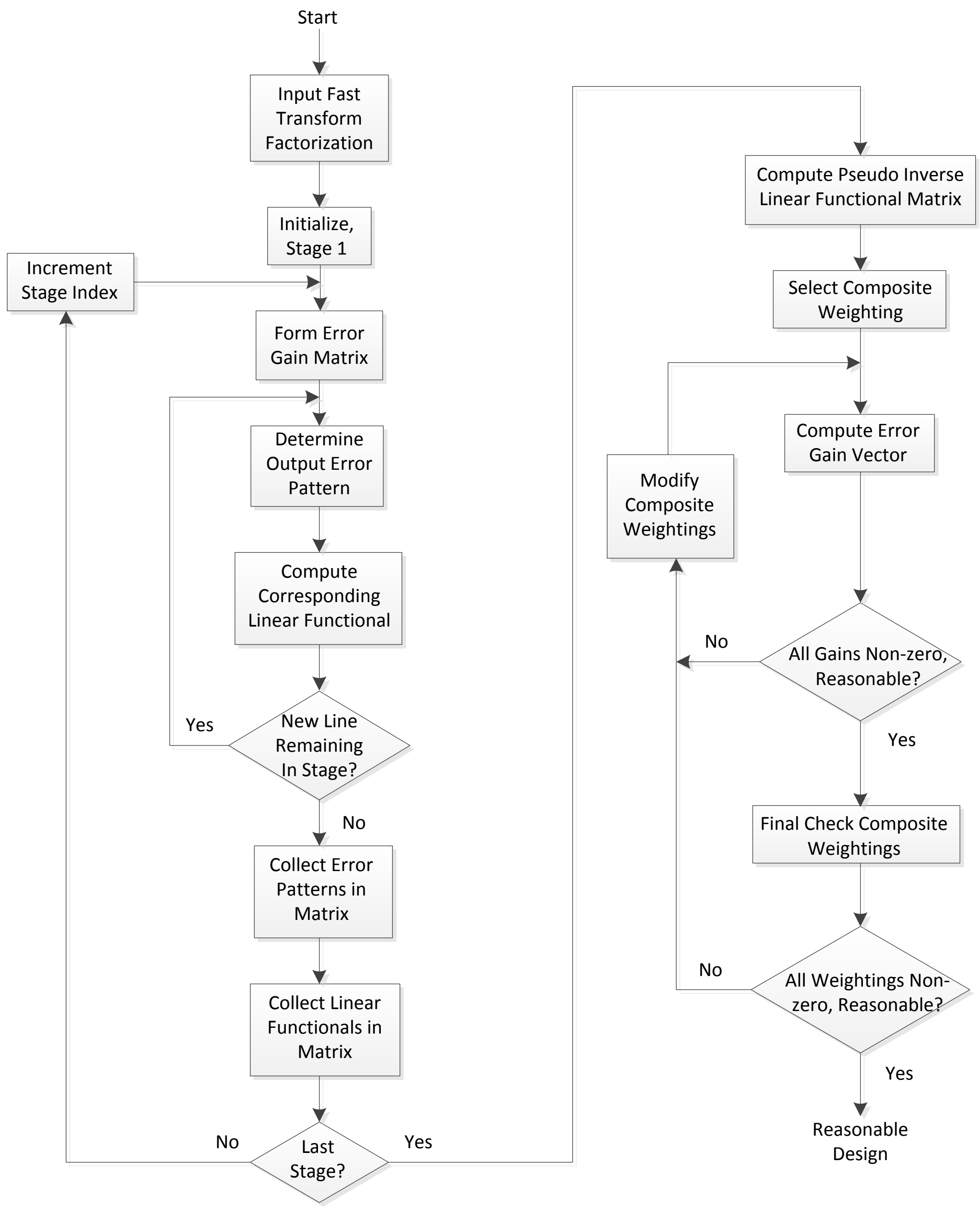
Redinbo and Nguyen's work [64] [65] extended parity weighted sums for discrete wavelet lifting transforms (DWT) error detection. The technique can detect errors introduced at a lifting section. However, it still relies on an iterative design process to determine the parity weighting vector and can lead to poor detection capability if weighting vector is not carefully chosen. No error correction capability can be achieved based on this technique. Since this method can be used only in 1-D DWT, in 2-D DWT applications, the parity weighted sum generation and detection needs to be implemented for each 1-D row/column-wise DWT stage. The resulting overhead is significant.

2.3 Proposed Algorithm

The proposed 2-D linear transform error detection method does not depend on the particular hardware or software structure of the transform being targeted. The foundation of the proposed method is Algorithm-based fault tolerance (ABFT) [39]. However, the naive implementation of ABFT cannot be directly applied in 2-D linear transform due to the following reasons.

1. ABFT can be used to detect and correct errors in matrix multiplication. A column checksum matrix multiplying a row checksum matrix will produce a full checksum matrix. However, multiplying a full check-

Figure 2.3: Code Design Methodology


Start
Input Fast Transform Factorization
Initialize, Stage 1
Increment Stage Index
Form Error Gain Matrix
Determine Output Error Pattern
Compute Corresponding Linear Functional
New Line Remaining In Stage?
Yes
No
Collect Error Patterns in Matrix
Collect Linear Functionals in Matrix
Last Stage?
No
Yes
Compute Pseudo Inverse Linear Functional Matrix
Select Composite Weighting
Compute Error Gain Vector
Modify Composite Weightings
All Gains Non-zero, Reasonable?
No
Yes
Final Check Composite Weightings
All Weightings Non-zero, Reasonable?
No
Yes
Reasonable Design

sum matrix by row, column or full checksum matrices will not create a full checksum matrix. If ABFT technique is applied directly on 2-D linear transform, the error detection needs to be done at each 1-D transform, which incurs a lot of overhead.

2. Linear transform operations may include row swapping/rearrangement at inputs or outputs. For example, In the wavelet lifting transform, the row data need to be rearranged to separate odd/even indices or to reposition the high/low-frequency subband. The encoding technique needs to be able to work even when the row data has been rearranged during the transformation.

3. Some linear transforms have the sparse coefficient matrix structure. To reduce computation cost, the actual implementation of the transformation may not keep the matrix format. For example, wave lifting structure is the popular method to implement discrete wavelet transform (DWT). Also, many popular linear transforms have fast HW architecture implementation which does not keep matrix structure either. For example, there are many different FFT algorithms/architectures to compute DFT sequences. The data encoding technique needs to be able to be integrated into the existing non-matrix form structure without significant architecture modifications.

To resolve the issues above, a generic weighted checksum code based fault tolerance technique (2DLTC) targeted to 2-D separable linear transformation is developed. It has proven that gate-level single stuck-at fault models are not satisfactory as any physical defect which affects a given area on a chip will affect a great amount of the circuitry and will cause a large block of logic to become faulty. The general module level fault model is assumed which allows a computation module to produce arbitrary erroneous outputs under failure condition.

The details are described in the following section. First, a few extensions of ABFT are introduced.

Definition 1.2: $f_r()$ is defined as the function takes a 2-D array as input to do a row-wise operation to reposition the rows in array in specific order defined by the transform equation.$\square$

For instance, in DWT, the following row reordering function $f_{r_{oe}}(X)$ and $f_{r_{hl}}(X)$ are being implemented.

- $X^{oe} = f_{r_{oe}}(X), X = [x_0 \cdots x_{2n-1}]^{\mathrm{T}}$, where $x_0, \cdots, x_{2n-1}$ are $1 * N$ array. Then $X^{oe} = [x_1 x_3 \cdots x_{2n-1} x_0 x_2 \cdots x_{2n-2}]^{\mathrm{T}}$

  $f_{r_{oe}}()$ function takes a 2-D array as input to do a row-wise operation to move an even index row to the upper region of the array in order and move an odd index row to the lower region of the array, in order.

- $X^{hl} = f_{r_{hl}}(X), X = [x_0 \cdots x_{2n-1}]^{\mathrm{T}}$, where $x_1, \cdots, x_{2n-1}$ are $1 * N$ array. Then $X^{hl} = [x_n \cdots x_{2n-1} x_0 \cdots x_{n-1}]^{\mathrm{T}}$

  $f_{r_{hl}}()$ function takes a 2-D array as input to do a row-wise operation to move the lower region of the array to the upper region and move the upper region of the array to the lower region.

The 2-D linear separable transform can be generalized in the following matrix form. Given a 2-D $N \times N$ input array X,

$$X_c = f_{r2}(M * f_{r1}(X)) \tag{2.18}$$

$$Y_{cr} = (f_{r2}(M * f_{r1}(X_c^{\mathrm{T}})))^{\mathrm{T}} \tag{2.19}$$

X_c is the result of 1-D column-wise transform

Y_{cr} is the result of 2-D transform which applies a 1-D column-wise transform first and then row-wise transform later.

M is the consolidated transform coefficient matrix.

f_{r1} and f_{r2} are matrix row reordering function. These functions are defined to generalize the matrix form to support transform like DWT. For linear transformation which does not need row reordering, these functions will be defined as No-Ops.

Corollary 1.1: When applying function $f_r()$ on a full checksum matrix C_f, the column checksum of C and the check of the checksum can be preserved. The function will be applied to C and the row checksum of C. Column checksum will not be reordered by $f_r()$ as $f_r()$ is meant to apply on the transform matrix only.

$$C = A * B \tag{2.20}$$

$$C_f = \left[\begin{array}{c|c} \dfrac{AB}{e^{\mathrm{T}}AB} & \dfrac{ABe}{e^{\mathrm{T}}ABe} \end{array} \right] \tag{2.21}$$

$$f_r(C_f) = \left[\begin{array}{c|c} \dfrac{f_r(AB)}{e^{\mathrm{T}}AB} & \dfrac{f_r(ABe)}{e^{\mathrm{T}}ABe} \end{array} \right] \tag{2.22}$$

Theorem 1.1: For 2-D separable linear transform, the result of applying the 1-D linear transform column-wise first and then row-wise, is the same as the result of applying the 1-D linear transform row-wise first and then column-wise.

Proof: This is possible because the 2-D separable transform linear functions can be expressed as separable functions which are the product of two 1-D linear transform functions [66].

In the case of column first, X_c is the intermediate result,

$$X_c = M^*X;\ Y = (M(X_c)^T)^T = M^*XM^T \tag{2.23}$$

In the case of row first, X_r is the intermediate result,

$$X_r = (MX^T)^T = XM^T;\ Y = M^*X_r = M^*XM^T \tag{2.24}$$

The theorem holds true even the row reordering operation $f_r()$ has been applied to the matrix.

$$X_c = f_{r_2}(M * f_{r_1}(X)) \tag{2.25}$$

$$X_r = (f_{r_2}(M * f_{r_1}(X^{\mathrm{T}})))^{\mathrm{T}} \tag{2.26}$$

$$Y_{rc} = f_{r_2}(M * f_{r_1}(X_r)) \tag{2.27}$$

$$Y = Y_{cr} = Y_{rc} \tag{2.28}$$

X_r: the result of 1-D row-wise transform.

Y_{rc}: the result of 2-D transform which applies 1-D row-wise first and then column-wise later.

Y: the result of 2-D transform. □

Theorem 1.2: For 2-D linear separable transform, there exists an input array column checksum weighting vector, so that the result array column checksum is the input array column weighted checksum applied to the 1-D transform.

Proof:

Based on Equation 2.21, 2.22 and 2.25, the full checksum of X_c can be derived as

$$\begin{aligned} X_{cf} &= f_{r_2}\left(\left[\frac{M}{e^{\mathrm{T}}M}\right] * \left[\ f_{r_1}(X) \mid f_{r_1}(X)e\ \right]\right) \\ &= f_{r_2}\left(\left[\ \frac{Mf_{r_1}(X)}{e^{\mathrm{T}}Mf_{r_1}(X)} \;\middle|\; \frac{Mf_{r_1}(X)e}{e^{\mathrm{T}}Mf_{r_1}(X)e}\ \right]\right) \\ &= \left[\ \frac{f_{r_2}(Mf_{r_1}(X))}{e^{\mathrm{T}}Mf_{r_1}(X)} \;\middle|\; \frac{f_{r_2}(Mf_{r_1}(X)e)}{e^{\mathrm{T}}Mf_{r_1}(X)e}\ \right] \\ &= \left[\ \frac{X_c}{e^{\mathrm{T}}Mf_{r_1}(X)} \;\middle|\; \frac{f_{r_2}(Mf_{r_1}(X)e)}{e^{\mathrm{T}}Mf_{r_1}(X)e}\ \right] \end{aligned} \tag{2.29}$$

And the row checksum of X_c^{T} can be represented as

$$X_c^{\mathrm{T}}e = (e^{\mathrm{T}}Mf_{r1}(X))^{\mathrm{T}} \tag{2.30}$$

$$f_{r1}(X_c^{\mathrm{T}})e = f_{r1}((e^{\mathrm{T}}Mf_{r1}(X))^{\mathrm{T}}) \tag{2.31}$$

and based on Equation 2.19, 2.21, 2.22 and 2.31 the full checksum of Y can be derived as

$$\begin{aligned} Y_f &= \left(f_{r_2}\left(\left[\ \frac{M}{e^{\mathrm{T}}M}\ \right] * \left[\ f_{r_1}(X_c^{\mathrm{T}}) \mid f_{r_1}(X_c^{\mathrm{T}})e\ \right]\right)\right)^{\mathrm{T}} \qquad (2.32) \\ &= \left(f_{r_1}\left(\left[\ \frac{M}{e^{\mathrm{T}}M}\ \right] * \left[\ f_{r_1}(X_c^{\mathrm{T}}) \mid f_{r_1}((e^{\mathrm{T}}Mf_{r_1}(X))^{\mathrm{T}})\ \right]\right)\right)^{\mathrm{T}} \\ &= \left(f_{r_2}\left(\left[\ \frac{Mf_{r_1}(X_c^{\mathrm{T}})}{e^{\mathrm{T}}Mf_{r_1}(X_c^{\mathrm{T}})} \;\middle|\; \frac{Mf_{r_1}((e^{\mathrm{T}}Mf_{r_1}(X))^{\mathrm{T}})}{e^{\mathrm{T}}Mf_{r_1}((e^{\mathrm{T}}Mf_{r_1}(X))^{\mathrm{T}})}\ \right]\right)\right)^{\mathrm{T}} \\ &= \left(\left[\ \frac{f_{r_2}(Mf_{r_1}(X_c^{\mathrm{T}}))}{e^{\mathrm{T}}Mf_{r_1}(X_c^{\mathrm{T}})} \;\middle|\; \frac{f_{r_2}(Mf_{r_1}((e^{\mathrm{T}}Mf_{r_1}(X))^{\mathrm{T}}))}{e^{\mathrm{T}}Mf_{r_1}((e^{\mathrm{T}}Mf_{r_1}(X))^{\mathrm{T}})}\ \right]\right)^{\mathrm{T}} \\ &= \left[\ \frac{Y}{(f_{r_2}(Mf_{r_1}((e^{\mathrm{T}}Mf_{r_1}(X))^{\mathrm{T}})))^{\mathrm{T}}} \;\middle|\; \frac{(e^{\mathrm{T}}Mf_{r_1}(X_c^{\mathrm{T}}))^{\mathrm{T}}}{e^{\mathrm{T}}Mf_{r_1}((e^{\mathrm{T}}Mf_{r_1}(X))^{\mathrm{T}})}\ \right] \end{aligned}$$

Therefore, the result array column checksum is

$$(f_{r_2}(M f_{r_1}((e^{\mathrm{T}} M f_{r_1}(X))^{\mathrm{T}})))^{\mathrm{T}} = (f_{r_2}(M f_{r_1}((W_C X)^{\mathrm{T}})))^{\mathrm{T}} \tag{2.33}$$

where W_C is the input array column checksum weighting vector. Since $e^{\mathrm{T}}M$ is known as the coefficient matrix is pre-determined, W_C can be obtained simply by reordering vector $e^{\mathrm{T}}M$ based on the reverse of function f_{r_1}.

$$W_C = f_{r1}^{-1}(e^{\mathrm{T}}M) \tag{2.34}$$

□

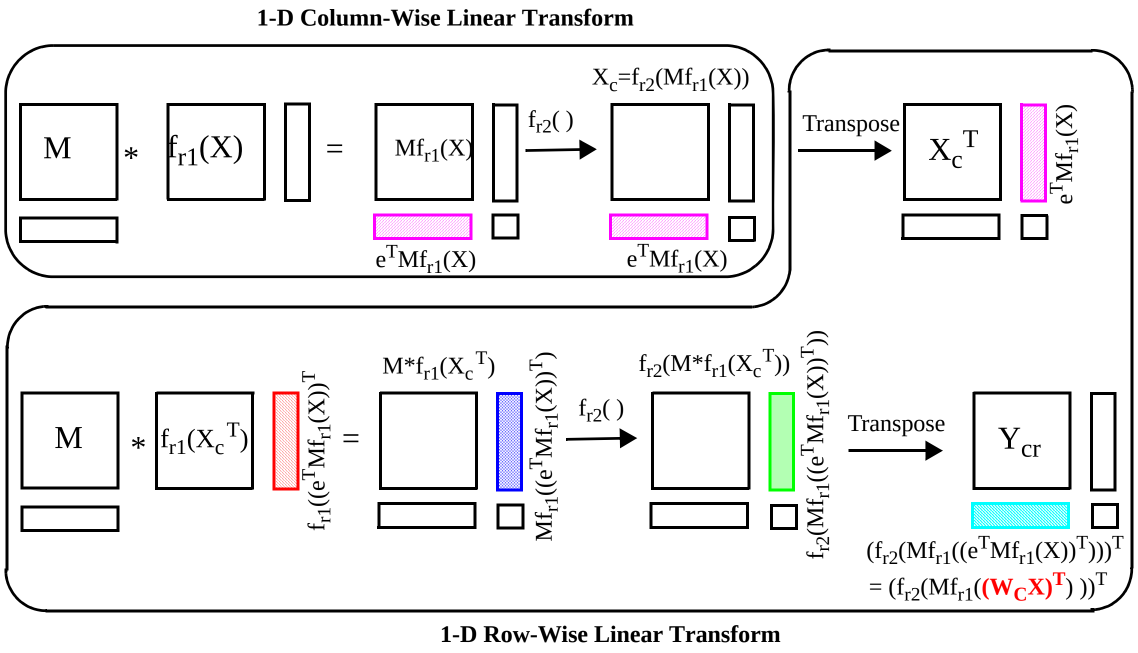
Figure 2.4 illustrates the graphical representation of Theorem 1.2.

Theorem 1.3: For 2-D linear separable transform, there exists an input array row checksum weighting vector, so that the result array row checksum is the input array row weighted checksum being applied the 1-D transform.

Proof:

Based on Equation 2.21, Equation 2.22, and Equation 2.26, the full checksum of X_r can be derived as

Figure 2.4: Graphical representation of Theorem 1.2



$$
\begin{aligned}
X_{rf} &= \left(f_{r_2}\left(\left[\begin{array}{c} M \\ \hline e^{\mathrm{T}}M \end{array} \right] * \left[\; f_{r_1}(X^{\mathrm{T}}) \mid f_{r_1}(X^{\mathrm{T}})e \; \right] \right) \right)^{\mathrm{T}} \\
&= \left(f_{r_2}\left(\left[\begin{array}{c|c} \dfrac{Mf_{r_1}(X^{\mathrm{T}})}{e^{\mathrm{T}}Mf_{r_1}(X^{\mathrm{T}})} & \dfrac{Mf_{r_1}(X^{\mathrm{T}})e}{e^{\mathrm{T}}Mf_{r_1}(X^{\mathrm{T}})e} \end{array} \right] \right) \right)^{\mathrm{T}} \\
&= \left(\left[\begin{array}{c|c} \dfrac{f_{r_2}(Mf_{r_1}(X^{\mathrm{T}}))}{e^{\mathrm{T}}Mf_{r_1}(X^{\mathrm{T}})} & \dfrac{f_{r_2}(Mf_{r_1}(X^{\mathrm{T}})e)}{e^{\mathrm{T}}Mf_{r_1}(X^{\mathrm{T}})e} \end{array} \right] \right)^{\mathrm{T}} \\
&= \left[\begin{array}{c|c} \dfrac{X_r}{(f_{r_2}(Mf_{r_1}(X^{\mathrm{T}})e))^{\mathrm{T}}} & \dfrac{(e^{\mathrm{T}}Mf_{r_1}(X^{\mathrm{T}}))^{\mathrm{T}}}{e^{\mathrm{T}}Mf_{r_1}(X^{\mathrm{T}})e} \end{array} \right] \qquad (2.35)
\end{aligned}
$$

And the row checksum of X_r can be represented as

$$X_r e = (e^{\mathrm{T}}Mf_{r1}(X^{\mathrm{T}}))^{\mathrm{T}} \qquad (2.36)$$

$$f_{r1}(X_r)e = f_{r1}((e^{\mathrm{T}}Mf_{r1}(X^{\mathrm{T}}))^{\mathrm{T}}) \qquad (2.37)$$

and the full checksum of Y as

$$\begin{aligned}
Y_f &= f_{r_2}\left(\left[\begin{array}{c} M \\ \hline e^{\mathrm{T}}M \end{array}\right] * \left[\begin{array}{c|c} f_{r_1}(X_r) & f_{r_1}(X_r)e \end{array}\right]\right) \\
&= f_{r_2}\left(\left[\begin{array}{c} M \\ \hline e^{\mathrm{T}}M \end{array}\right] * \left[\begin{array}{c|c} f_{r_1}(X_r) & f_{r_1}((e^{\mathrm{T}}Mf_{r_1}(X^{\mathrm{T}}))^{\mathrm{T}}) \end{array}\right]\right) \\
&= f_{r_2}\left(\left[\begin{array}{c|c} \dfrac{Mf_{r_1}(X_r)}{e^{\mathrm{T}}Mf_{r_1}(X_r)} & \dfrac{Mf_{r_1}((e^{\mathrm{T}}Mf_{r_1}(X^{\mathrm{T}}))^{\mathrm{T}})}{e^{\mathrm{T}}Mf_{r_1}((e^{\mathrm{T}}Mf_{r_1}(X^{\mathrm{T}}))^{\mathrm{T}})} \end{array}\right]\right) \\
&= \left[\begin{array}{c|c} \dfrac{f_{r_2}(Mf_{r_1}(X_r))}{e^{\mathrm{T}}Mf_{r_1}(X_r)} & \dfrac{f_{r_2}(Mf_{r_1}((e^{\mathrm{T}}Mf_{r_1}(X^{\mathrm{T}}))^{\mathrm{T}}))}{e^{\mathrm{T}}Mf_{r_1}((e^{\mathrm{T}}Mf_{r_1}(X^{\mathrm{T}}))^{\mathrm{T}})} \end{array}\right] \\
&= \left[\begin{array}{c|c} \dfrac{Y}{e^{\mathrm{T}}Mf_{r_1}(X_r)} & \dfrac{f_{r_2}(Mf_{r_1}((e^{\mathrm{T}}Mf_{r_1}(X^{\mathrm{T}}))^{\mathrm{T}}))}{e^{\mathrm{T}}Mf_{r_1}((e^{\mathrm{T}}Mf_{r_1}(X^{\mathrm{T}}))^{\mathrm{T}})} \end{array}\right] \qquad (2.38)
\end{aligned}$$

Therefore, the result array row checksum is

$$f_{r_2}(Mf_{r_1}((e^{\mathrm{T}}Mf_{r_1}(X^{\mathrm{T}}))^{\mathrm{T}})) = f_{r_2}(Mf_{r_1}((W_R X^{\mathrm{T}})^{\mathrm{T}})) \qquad (2.39)$$

where W_R is the input array column checksum weighting vector. Since $e^{\mathrm{T}}M$ is known, W_R can be obtained simply by reordering vector $e^{\mathrm{T}}M$ based on the reverse of function f_{r_1}.

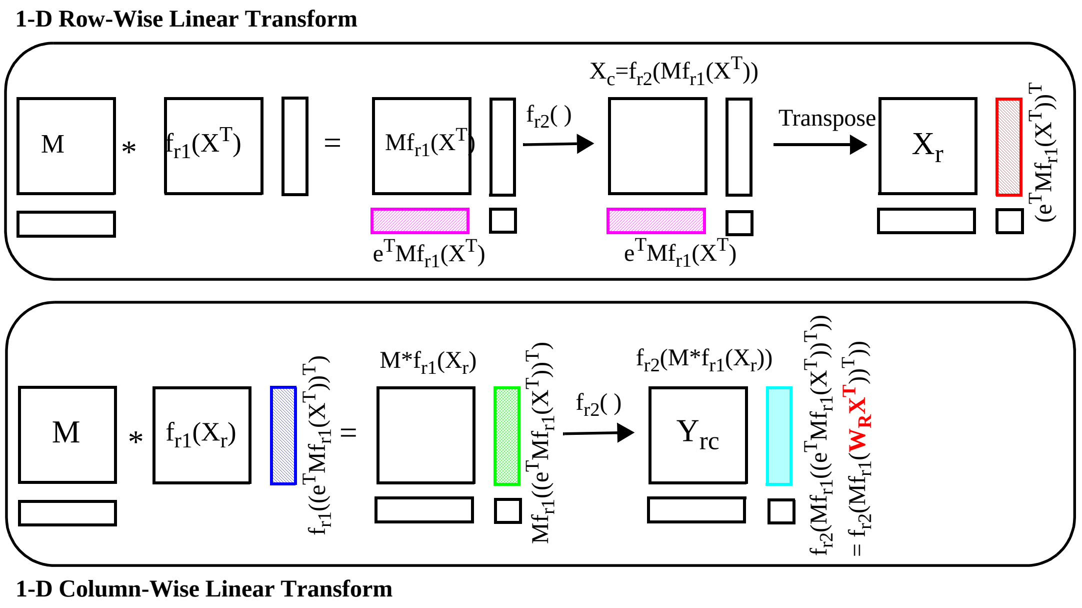
$$W_R = f_{r1}^{-1}(e^{\mathrm{T}}M) \qquad (2.40)$$

$\square$

Figure 2.5 illustrates the graphical representation of Theorem 1.3.

From Equation 2.33 and Equation 2.39 shown before, the following equations can be derived

Figure 2.5: Graphical representation of Theorem 1.3



$$W_C x = e^{\mathrm{T}} M f_{r_1}(x) \tag{2.41}$$

$$W_R x^{\mathrm{T}} = e^{\mathrm{T}} M f_{r_1}(x^{\mathrm{T}}) \tag{2.42}$$

$$W_C = W_R = f_{r_1}^{-1}(e^{\mathrm{T}} M) \tag{2.43}$$

Base on the coefficient matrix of the transform, the checksum weighting vector for given transform can be derived using Equation 2.43. The checksum weighting vector is essentially column checksum of the coefficient matrix $e^T M$. Some transforms like DWT have complicated form, and Matlab symbolic computation can be used to find out the consolidated coefficient matrix. In case the transform involves row re-ordering, the reverse ordering can be applied to derive the checksum weighting factor.

2.4 Error Detection Scheme

The proposed error detection scheme utilized both data redundancy and time redundancy at algorithm level to lower the protection overhead. The error detection scheme is described below.

1. Encode input array using transformation specific checksum weighting vector as described in Equation 2.43.

2. Perform 1-D transform on encoded input vector to generate golden column and row checksum.

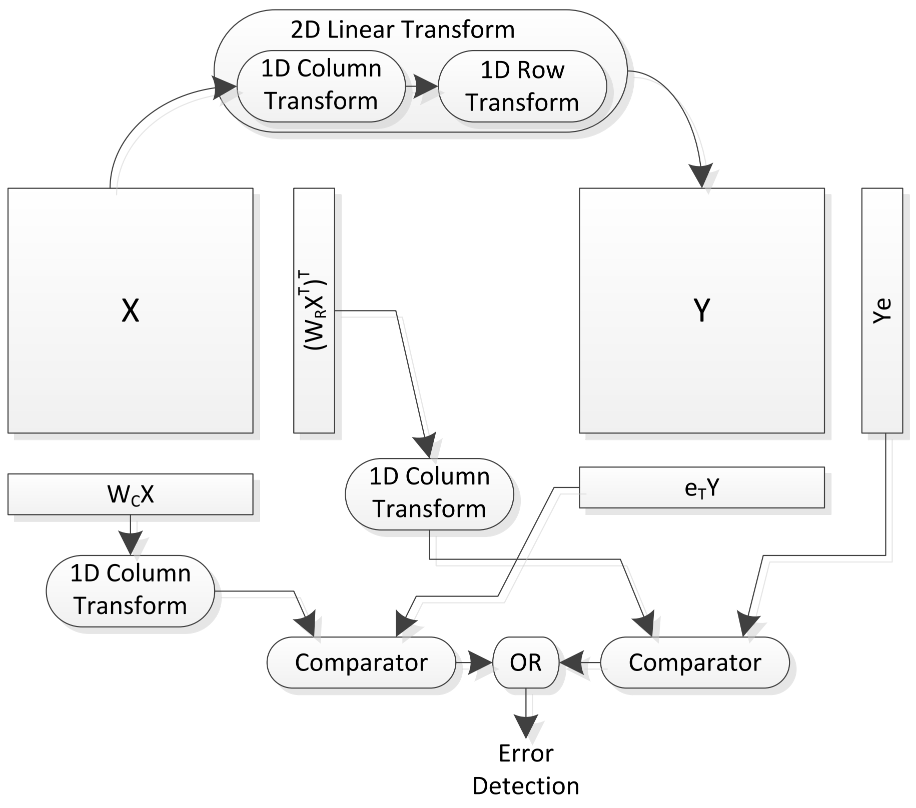
3. Compute the sum of output array elements in each row and column to form revised column and row checksum.

4. Compare each computed sum with the corresponding checksum vector entry. Due to potential round-off errors, a small tolerance should be allowed for this comparison.

5. An inconsistent row or column is detected when there is a mismatch in comparison. If any such case happens, an error is detected in the 2-D linear transform application.

The block diagram of proposed error detection scheme is shown in Figure 2.6.

Figure 2.6: 2-D Linear Transform Error Detection



2.5 2-D Discrete Fourier Transform Error Detection

2.5.1 Background

The discrete Fourier transform (DFT) converts a finite sequence of equally-spaced samples of a function into an equivalent-length sequence of equally-spaced samples of the discrete-time Fourier transform. The DFT is a frequency domain representation of the original input sequence. The DFT is the critical discrete transform used to perform Fourier analysis in many practical applications. In digital signal processing, the samples can be any physical measurement over a finite time interval. In image processing, the samples can be the values of pixels along a row or column of the raw image. The DFT is also used to conveniently solve partial differential equations and to perform other arithmetic such as convolutions. The 2-D DFT is a direct extension of the 1-D case and is given by

$$y(k,l) = \frac{1}{N^2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} x(m,n) \mathrm{e}^{-j2\pi(\frac{km+ln}{N})} \tag{2.44}$$

for $k, l = 0, 1, 2, \ldots, N-1$.

2.5.2 Checksum Weighting Vector

For 2-D $N \times N$ DFT, the checksum weighting vector can be derived as,

$$W_{DFT}[0] = N; \tag{2.45}$$

$$W_{DFT}[i] = 1 + \sum_{k=1}^{N-1} \mathrm{e}^{(\frac{-2\pi j}{N})^{ik}} \tag{2.46}$$

where $i = 1, \cdots, N-1$

2.6 2-D Discrete Cosine Transform Error Detection

2.6.1 Background

Discrete Cosine Transform (DCT) has emerged as the effective image transformation in most visual systems. DCT has been widely deployed by present video coding standards, for example, MPEG, JPEG, etc. The 2-D DCT is a direct extension of the 1-D case and is given by

$$y(k,l) = \alpha(k)\alpha(l)\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} x(m,n)cos(\frac{\pi(2m+1)k}{2N})cos(\frac{\pi(2n+1)l}{2N}) \qquad (2.47)$$

for $k, l = 0, 1, 2, \ldots, N-1$. $\alpha(k)$ and $\alpha(l)$ are defined as

$$\alpha(k) = \begin{cases} \sqrt{\frac{1}{N}}, & \text{for } k = 0. \\ \sqrt{\frac{2}{N}}, & \text{for } k \neq 0. \end{cases} \qquad (2.48)$$

2.6.2 Checksum Weighting Vector

For 2-D $N \times N$ DCT, the checksum weighting vector can be derived as,

$$W_{DCT}[i] = \sqrt{\frac{1}{N}} + \sqrt{\frac{2}{N}}\sum_{j=1}^{N-1} cos((\frac{\pi j}{2N})(2i+1)) \qquad (2.49)$$

where $i = 0, \cdots, N-1$

2.7 2-D Discrete Hartley Transform Error Detection

2.7.1 Background

A discrete Hartley transform (DHT) is a Fourier-related transform of discrete, periodic data similar to the discrete Fourier transform (DFT), with similar applications in signal processing and related fields. Its main difference from the DFT is that the transform converts real inputs to real outputs, with no natural involvement of complex numbers. Same as the DFT is the discrete analogue of the continuous Fourier transform, the DHT is the discrete analogue of the continuous Hartley transform, introduced by R. V. L. Hartley in 1942 [67].

Since there are fast algorithms for the DHT analogous to the fast Fourier transform (FFT), the DHT was originally proposed by R. N. Bracewell in 1983 as a more efficient computational tool in the common case where the data are purely real [68].

$$y(k,l) = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} x(m,n) cas(\frac{2\pi km}{N}) cas(\frac{2\pi ln}{N}) \tag{2.50}$$

for $k, l = 0, 1, 2, \ldots, N-1$.

$$cas(\frac{2\pi km}{N}) = cos(\frac{2\pi km}{N}) + sin(\frac{2\pi km}{N}) \tag{2.51}$$

2.7.2 Checksum Weighting Vector

For 2-D $N \times N$ DHT, the checksum weighting vector can be derived as,

$$W_{DHT}[0] = N; \tag{2.52}$$

$$W_{DHT}[i] = 1 + \sum_{k=1}^{N-1}(sin(\frac{2\pi}{N}ik) + cos(\frac{2\pi}{N}ik)) = 0 \tag{2.53}$$

where $i = 1, \cdots, N-1$.

In 2-D DHT, as there is only one non-zero element in checksum weighting vector, it will take only N multiplications to derive input checksum vector. This application-specific property for 2-D DHT checksum weighting vector can greatly reduce the error detection computation overhead.

2.8 2-D Discrete Wavelet Transform Error Detection

2.8.1 Background

The Discrete Wavelet Transform (DWT) has become a widely used signal processing tool over the last decade. It has been effectively used in signal and image processing applications since its multi-resolution analysis capability. 2-D DWT is also at the center of JPEG 2000 image compression standard. DWT is ultimately implemented in either hardware or software, and it is susceptible to transient failure caused by either radiation, noise, or timing errors as well. Due to its pipelined structure and multi-rate processing requirements, a single numerical error in one stage can easily affect multiple outputs in the final result. Most importantly, the influence of these failure mechanisms increases with technology scaling trends. It is desirable to develop a low-cost error detection method for 2-D DWT.

A discrete wavelet transform (DWT) is any wavelet transform for which the input signal and wavelet parameters are discretely sampled. DWT has traditionally been implemented using convolution or FIR filter bank structures. This kind of implementation leads to a very large number of arithmetic computations and requires a large amount of storage. Wave lifting [69] is a mathematical formulation for wavelet transformation based on the spatial construction of the wavelets and a very versatile scheme for its factorization. Wave lifting implementation breaks up the high-pass and low-pass wavelet filters into a sequence of upper and lower triangular matrices and converts the filter implementation into banded matrix multiplications. Wave lifting implementation requires far fewer computations compared to convolution based DWT. The lifting factorization can be represented in the following forms.

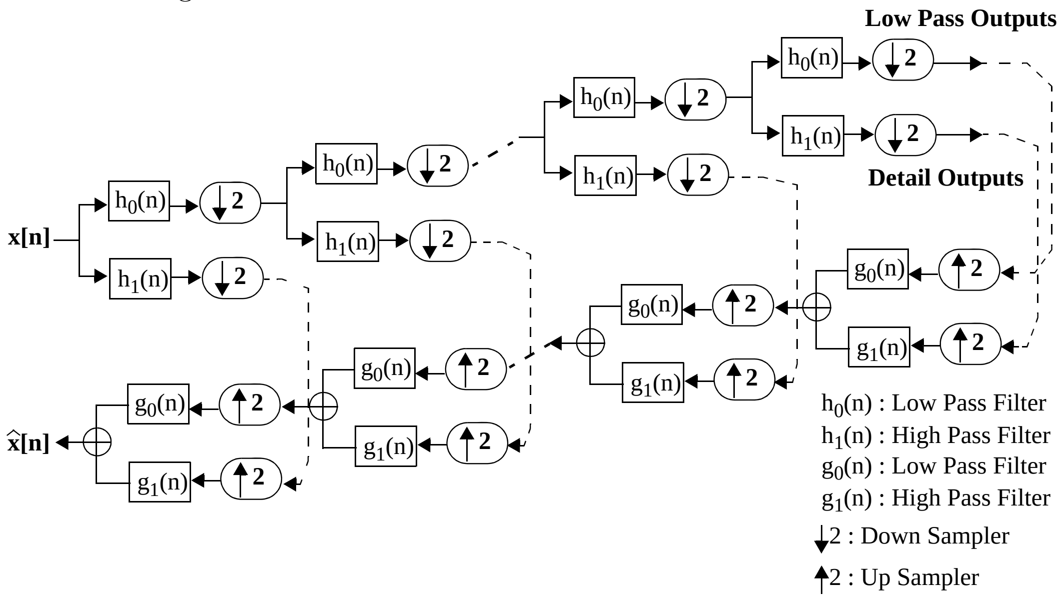
$$\widetilde{P}(z) = \left(\prod_{i=1}^{m} \begin{bmatrix} 1 & \widetilde{s_i}(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \widetilde{t_i}(z) & 1 \end{bmatrix} \right) \begin{bmatrix} K & 0 \\ 0 & \frac{1}{K} \end{bmatrix} \tag{2.54}$$

where $\widetilde{s_i}(z)$ and $\widetilde{t_i}(z)$ are Laurent polynomials, and K is a constant act as a scaling factor.

The lifting based forward wavelet transform essentially first splits the input stream into even and odd samples, then alternately executes update and predict lifting steps, and finally scales the two output streams by 1/K and K, respectively, to produce low-pass and high-pass subbands. A prediction step consists of predicting each odd sample as a linear combination of the even samples and subtracting it from the odd sample to form the prediction error. An update step consists of updating the even samples by adding them to a linear combination of the prediction errors to form the updated sequence.

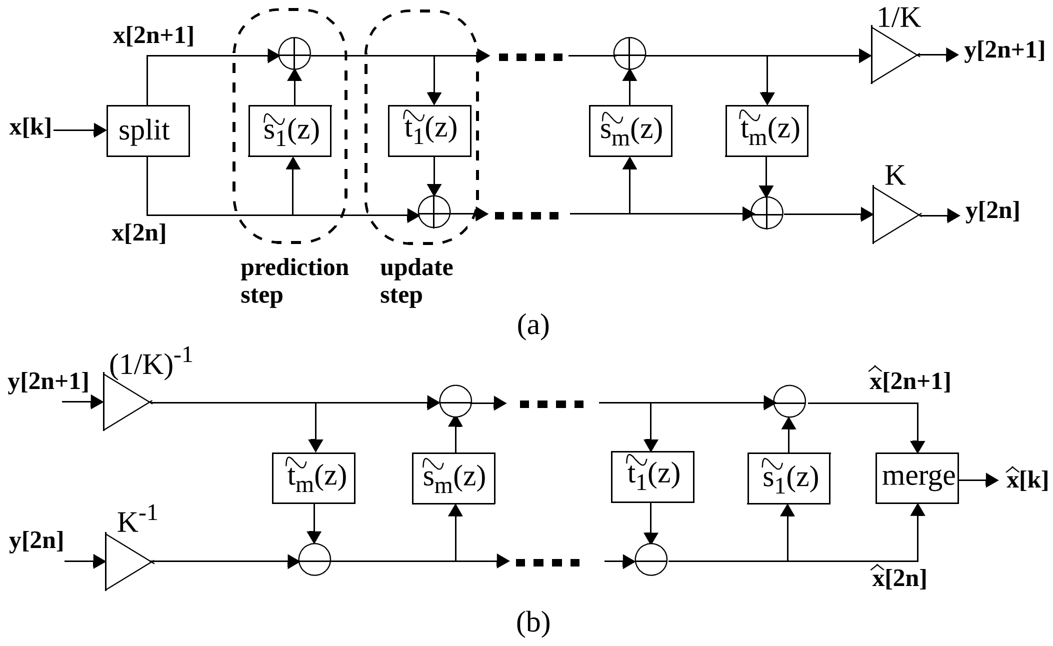
The block diagram of filter bank based forward and inverse DWT is shown in Figure 2.7. The block diagram of wave lifting is shown in Figure 2.8.

Figure 2.7: Filter bank based forward and inverse DWT



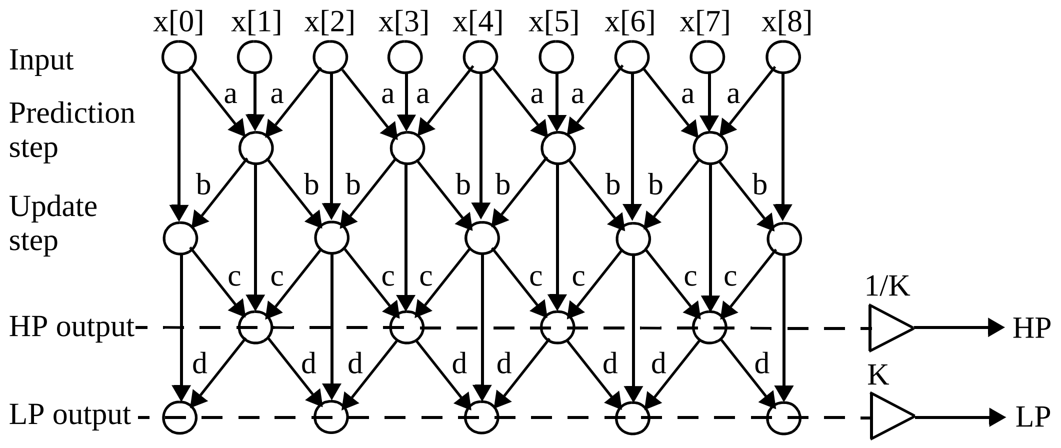
The data dependency of the wave lifting scheme can be explained via a dataflow graph as shown in Figure 2.9. For DWT filters which can be decomposed into four lifting factors, the computation is done in four stages. The value of a, b, c, d, and K depend on the selection of the DWT filters. Once the DWT filters are chosen, they are constant throughout the processing. The intermediate results generated in the first two stages for the first two lifting steps are stored temporarily and these intermediate results are subsequently processed to produce the high-pass (HP) outputs in the third stage followed

Figure 2.8: Lifting based forward DWT (Analysis) and inverse DWT (Synthesis)



by the low-pass (LP) outputs in the final stage. For the DWT filters requiring fewer factors, the intermediate stages can be simply bypassed.

Figure 2.9: Data dependency diagram of lifting-based DWT with four lifting factors [66]



The proposed error detection technique can be used with any variation of the wavelet transform. However, the 9/7 wavelet transform which is being adopted in JPEG 2000 standard will be used as an example to demonstrate the idea [70]. The application of the technique to the 5/3 wavelet transforms is also shown. For other types of wavelet transforms, the same encoding technique can still be applied, with just the weighting vector being different.

The 9/7 wavelet is implemented in four lifting steps. For 1-D DWT, the original matrix forms for wave lifting steps 1 and 2 are shown below.

Given a 1-D vector x,

$$d^{(0)} = [d_0^{(0)} \cdots d_{n-1}^{(0)}]^{\mathrm{T}} = [x_1 \cdots x_{2n-1}]^{\mathrm{T}} \tag{2.55}$$

$$s^{(0)} = [s_0^{(0)} \cdots s_{n-1}^{(0)}]^{\mathrm{T}} = [x_0 \cdots x_{2n-2}]^{\mathrm{T}} \tag{2.56}$$

$$\begin{bmatrix} d^{(1)} \\ s^{(1)} \end{bmatrix} = \begin{bmatrix} I & \alpha M_a \\ \beta M_d & \alpha\beta M_b \end{bmatrix} \begin{bmatrix} d^{(0)} \\ s^{(0)} \end{bmatrix} = M_1 * \begin{bmatrix} d^{(0)} \\ s^{(0)} \end{bmatrix} \tag{2.57}$$

$$M_a = \begin{bmatrix} 1 & 1 & & 0 \\ & \ddots & \ddots & \\ & & 1 & 1 \\ 0 & & & 2 \end{bmatrix}, M_d = \begin{bmatrix} 2 & & & 0 \\ 1 & 1 & & \\ & \ddots & \ddots & \\ 0 & & 1 & 1 \end{bmatrix} \tag{2.58}$$

$$M_b = \begin{bmatrix} 2+\frac{1}{\alpha\beta} & 2 & & & 0 \\ 1 & 2+\frac{1}{\alpha\beta} & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 2+\frac{1}{\alpha\beta} & 1 \\ 0 & & & 1 & 3+\frac{1}{\alpha\beta} \end{bmatrix} \tag{2.59}$$

where $\alpha = -1.586134342, \beta = -0.05298011854$.

Wave lifting step 3 and step 4 are similar to step 1 and step 2, except that the coefficients are changed. α is replaced by γ, and β is replaced by δ.

$$\begin{bmatrix} d^{(2)} \\ s^{(2)} \end{bmatrix} = \begin{bmatrix} I & \gamma M_a \\ \delta M_d & \gamma\delta M_c \end{bmatrix} \begin{bmatrix} d^{(1)} \\ s^{(1)} \end{bmatrix} = M_2 * \begin{bmatrix} d^{(1)} \\ s^{(1)} \end{bmatrix} \tag{2.60}$$

$$M_c = \begin{bmatrix} 2+\frac{1}{\gamma\delta} & 2 & & & 0 \\ 1 & 2+\frac{1}{\gamma\delta} & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 2+\frac{1}{\gamma\delta} & 1 \\ 0 & & & 1 & 3+\frac{1}{\gamma\delta} \end{bmatrix} \tag{2.61}$$

where $\gamma = 0.8829111, \delta = 0.4435068$.

The normalization can be written in the following matrix form.

$$\begin{bmatrix} d \\ s \end{bmatrix} = \begin{bmatrix} \frac{K}{2}I & 0 \\ 0 & \frac{1}{K}I \end{bmatrix} \begin{bmatrix} d^{(2)} \\ s^{(2)} \end{bmatrix} = M_s * \begin{bmatrix} d^{(2)} \\ s^{(2)} \end{bmatrix} \tag{2.62}$$

where $k = 1.2301741$.

The 2-D DWT can be represented in the following matrix form.

Given a 2-D $N \times N$ array X,

$$X_c = f_{hl}(M_s * M_2 * M_1 * f_{oe}(X)) \tag{2.63}$$

$$Y_{cr} = (f_{hl}(M_s * M_2 * M_1 * f_{oe}(X_c^{\mathrm{T}})))^{\mathrm{T}} \tag{2.64}$$

X_c: the result of 1-D column-wise wavelet transform

Y_{cr}: the result of 2-D wavelet transform which applies a 1-D DWT column-wise first and then row-wise later.

$f_{oe}()$: the function takes a 2-D array as input to do a row-wise operation to move an even index row to the upper region of the array in order and move an odd index row to the lower region of the array, in order.

$X_{oe} = f_{oe}(X), x = [x_0 \cdots x_{2n-1}]^{\mathrm{T}}$, where $x_0, \cdots, x_{2n-1}$ are $1 * N$ array. Then $X_{oe} = [x_1 x_3 \cdots x_{2n-1} x_0 x_2 \cdots x_{2n-2}]^{\mathrm{T}}$

$f_{hl}()$: the function takes a 2-D array as input to do a row-wise operation to move the lower region of the array to the upper region and move the upper region of the array to the lower region.

$X_{hl} = f_{hl}(X), X_{hl} = [x_0 \cdots x_{2n-1}]^{\mathrm{T}}$, where $x_1, \cdots, x_{2n-1}$ are $1 * N$ array. Then $X_{hl} = [x_n \cdots x_{2n-1} x_0 \cdots x_{n-1}]^{\mathrm{T}}$

For convenience, M_s, M_2, and M_1 can be consolidated into a single matrix M, where

$$M = M_s * M_2 * M_1 \tag{2.65}$$

Equation 2.63 and Equation 2.64 can be simplified to below which is the same as generic 2-D linear separable transformation format and therefore the proposed error detection algorithm can be applied on also discrete wavelet transform although the coefficient matrix seems complicate.

$$X_c = f_{hl}(M * f_{oe}(X)) \tag{2.66}$$

$$Y_{cr} = (f_{hl}(M * f_{oe}(X_c^{\mathrm{T}})))^{\mathrm{T}} \tag{2.67}$$

2.8.2 Checksum Weighting Vector

To come up with the checksum weighting vector of 2-D DWT, the consolidated coefficient matrix M need to be derived. The matrix M has a complicated form, but for encoding purpose, only the column checksum for this consolidated coefficient matrix need to be calculated. Matlab symbolic computation is used to find the following pattern. For coefficient matrix M with size $N * N$, the column checksum of M for 9/7 wavelet transform will be

For $n = 1$,

$e^{\mathrm{T}}M[n] = \dfrac{k}{2} + \dfrac{(3\delta)}{k} + \dfrac{(3\beta(2\gamma\delta + 1))}{k} + 2\beta\gamma k + \dfrac{(5\beta\gamma\delta)}{k} = 1.2669$

For $n = 2$,

$e^{\mathrm{T}}M[n] = \dfrac{k}{2} + \dfrac{(2\delta)}{k} + \dfrac{(2\beta(2\gamma\delta + 1))}{k} + 2\beta\gamma k + \dfrac{(5\beta\gamma\delta)}{k} = 0.9831$

For $2 < n < \dfrac{N}{2} - 1$,

$e^{\mathrm{T}}M[n] = 1$

For $n = \dfrac{N}{2} - 1$,

$e^{\mathrm{T}}M[n] = \dfrac{k}{2} + \dfrac{(2\delta)}{k} + \dfrac{(\beta(2\gamma\delta + 1))}{k} + \dfrac{\beta(3\gamma\delta + 1)}{k} + \dfrac{(5\beta\gamma k)}{2} + \dfrac{3\beta\gamma\delta}{k} =$ 0.9712

For $n = \frac{N}{2}$,

$e^{\mathrm{T}}M[n] = \frac{k}{2} + \frac{(\delta)}{k} + \frac{(\beta(3\gamma\delta+1))}{k} + \frac{(3\beta\gamma k)}{2} + \frac{\beta\gamma\delta}{k} = 0.7788$

For $n = \frac{N}{2} + 1$,

$e^{\mathrm{T}}M[n] = \frac{\alpha k}{2} + \frac{(2\alpha\beta+1)(2\gamma\delta+1)}{k} + \frac{\gamma k(2\alpha\beta+1)}{2} + \frac{3\alpha\delta}{k} + \frac{\alpha\beta(2\gamma\delta+1)}{k} + \frac{\gamma\delta(2\alpha\beta+1)}{k} + \alpha\beta\gamma k + \frac{3\alpha\beta\gamma\delta}{k} = 0.3015$

For $n = \frac{N}{2} + 2$,

$e^{\mathrm{T}}M[n] = \alpha k + \frac{(2\alpha\beta+1)(2\gamma\delta+1)}{k} + \gamma k(2\alpha\beta+1) + \frac{5\alpha\delta}{k} + \frac{3\alpha\beta(2\gamma\delta+1)}{k} + \frac{3\gamma\delta(2\alpha\beta+1)}{k} + 2\alpha\beta\gamma k + \frac{4\alpha\beta\gamma\delta}{k} = -0.0782$

For $n = \frac{N}{2} + 3$,

$e^{\mathrm{T}}M[n] = \alpha k + \frac{(2\alpha\beta+1)(2\gamma\delta+1)}{k} + \gamma k(2\alpha\beta+1) + \frac{4\alpha\delta}{k} + \frac{2\alpha\beta(2\gamma\delta+1)}{k} + \frac{2\gamma\delta(2\alpha\beta+1)}{k} + 2\alpha\beta\gamma k + \frac{5\alpha\beta\gamma\delta}{k} = 0.0267$

For $\frac{N}{2} + 3 < n < N - 1$,

$e^{\mathrm{T}}M[n] = 0$

For $n = N - 1$,

$e^{\mathrm{T}}M[n] = \alpha k + \frac{(2\alpha\beta+1)(2\gamma\delta+1)}{k} + \gamma k(2\alpha\beta+1) + \frac{4\alpha\delta}{k} + \frac{\alpha\beta(2\gamma\delta+1)}{k} + \frac{2\gamma\delta(2\alpha\beta+1)}{k} + \frac{\alpha\beta(3\gamma\delta+1)}{k} + \frac{5\alpha\beta\gamma k}{2} + \frac{3\alpha\beta\gamma\delta}{k} = 0.0456$

For $n = N$,

$e^{\mathrm{T}}M[n] = \frac{3\alpha k}{2} + \frac{(3\alpha\beta+1)(3\gamma\delta+1)}{k} + \frac{3\gamma k(3\alpha\beta+1)}{2} + \frac{4\alpha\delta}{k} + \frac{\alpha\beta(2\gamma\delta+1)}{k} + \frac{\gamma\delta(3\alpha\beta+1)}{k} + \alpha\beta\gamma k + \frac{2\alpha\beta\gamma\delta}{k} = -0.2956$

N is an even number since symmetric extension of the input array is assumed in this work. It can be observed that no matter how big the matrix is, there are only 9 non-zero or one column checksum for this coefficient matrix.

From equations shown before,

$$W_C x = e^{\mathrm{T}} M f_{oe}(x) \tag{2.68}$$

$$W_R x^{\mathrm{T}} = e^{\mathrm{T}} M f_{oe}(x^{\mathrm{T}}) \tag{2.69}$$

$$W_C = W_R \tag{2.70}$$

Since the pattern of $e^{\mathrm{T}}M$ is already known, W can be obtained simply by reordering vector $e^{\mathrm{T}}M$ based on the reverse of function $f_{oe}()$.

$$\begin{aligned} W_{9/7DWT} =& [0.3015, 1.2669, -0.0782, 0.9831, 0.0267, \\ & 1, 0, 1 \cdots, 0, 1, 0.0456, 0.9712, -0.2956, 0.7788] \end{aligned}$$

Although these results are only for the 9/7 wavelet transform, the derivation of other wavelet transform coefficient matrix column checksums can be obtained in the same manner. For example, the checksum weighting vector for 5/3 wavelet transform can be represented below.

$$W_{5/3DWT} = [0.375, 1.25, -0.125, 1, 0, 1 \cdots, 0, 1, -0.25, 0.75]$$

Although only the encoding and checksum generation of forward DWT is shown, the encoding and checksum generation of reverse DWT can be obtained via the same method. Also, the weighted checksum code scheme proposed here is totally different than Jou's study [71]. In proposed method, only the single checksum is used, and the weighting vector is different.

2.8.3 Integration with existing wave lifting VLSI architecture

From Equation 2.33 and 2.39, it can be observed that the output array column checksum is essentially the input array column checksum $W_C X$ being applied to 1-D DWT, and the output array row checksum is essentially the input array row checksum $(W_R X^{\mathrm{T}})^{\mathrm{T}}$ being applied to 1-D DWT. Once the input array checksum is calculated, the result can be feed into the existing wave lifting VLSI architecture pipeline and just be treated as an additional row or column. Therefore, this encoding scheme can work with existing wave lifting VLSI architectures without significant modification.

2.9 Simulation and Analysis

2.9.1 Error Detection Capability

Error detection capability is the most important metric to be evaluated for the proposed scheme. The proposed scheme need to ensure that there exists an error threshold which can be used to detect a small error while the false alarm will not be triggered due to checksum rounding. Matlab simulation is performed to verify the metric. The main purpose of the simulation is to investigate the proper value of error detection threshold. In the simulation, the input array with size $N \times N$ is randomly generated. Each element of the array is assigned integer range from 0 to 255. Different 2-D linear transforms (DCT, DWT and DHT) are performed on an array without error and with single error injection. The simulation program randomly selects the stage/node on which a random magnitude error with zero mean and various variance is superimposed. The input array is encoded with proposed weighted checksum scheme and compared with output array golden (from error free computation) and revised (from error injected computation) checksum. By comparing encoded checksum and golden output checksum, maximum checksum rounding error can be derived. Maximum and minimal checksum error can be obtained by comparing encoded checksum and revised output checksum across iterations. For each error variance, the simulation runs 100000 iterations and logs maximum/minimal checksum error and maximum rounding error ever observed.

The simulation results are summarized in Table 2.1, Table 2.3 and Table 2.2. As shown in simulation result, across different 2-D linear transform test cases, minimal checksum difference due to error injection is several magnitudes larger than maximum rounding error ever observed. It demonstrates that there exists an error threshold setting which can avoid false alarm triggering due to checksum rounding and detect all injected error in simulation.

Table 2.1: 2-D DCT Transform Single Error Injection and Detection Matlab Simulation Result

Array Size N	Err Variance	Max Checksum Difference	Min Checksum Difference	Max Rounding Error
100	1	28.4017	$8.2206 * 10^{-6}$	$9.8225 * 10^{-11}$
100	5	182.5088	$1.5561 * 10^{-5}$	$9.7856 * 10^{-11}$
1000	1	66.8016	$3.2522 * 10^{-6}$	$8.5565 * 10^{-9}$
1000	5	302.7623	$6.9888 * 10^{-6}$	$8.4983 * 10^{-9}$

Table 2.2: 2-D DHT Transform Single Error Injection and Detection Matlab Simulation Result

Array Size N	Err Variance	Max Checksum Difference	Min Checksum Difference	Max Rounding Error
100	1	320.6826	$2.0031 * 10^{-5}$	$4.9593 * 10^{-8}$
100	5	1568.6	$4.5884 * 10^{-5}$	$4.936 * 10^{-8}$
1000	1	2303.9	$6.786 * 10^{-5}$	$2.2456 * 10^{-5}$
1000	5	9564.5	$6.8337 * 10^{-5}$	$2.2158 * 10^{-5}$

Table 2.3: 2-D 9/7 DWT Transform Single Error injection and Detection Matlab Simulation Result

Array Size N	Err Variance	Max Checksum Difference	Min Checksum Difference	Max Rounding Error
100	1	5.8504	$1.6058 * 10^{-5}$	$3.7018 * 10^{-8}$
100	5	28.1763	$2.6628 * 10^{-5}$	$3.7828 * 10^{-8}$
1000	1	5.7566	$1.5747 * 10^{-5}$	$1.3148 * 10^{-7}$
1000	5	29.4913	$1.6634 * 10^{-5}$	$1.3523 * 10^{-7}$

2.9.2 Error Detection Overhead Analysis

In this section, the computation overhead of proposed generic 2-D linear transform error detection scheme is compared to prior work which target general transform application. The weighted check sum technique in Jou's work [71] is excluded from the comparison since its cost is obviously higher than ABFT. The cost to encode and compute an $N \times N$ array is used as an example.

1. 2-D Linear Transform Error Check (2DLTC): The proposed algorithm is referred as 2-D linear transform error check (2DLTC). $W_C X = W_R X$ has been shown therefore either row or column checksum can be used for error detection. For 9/7 2-D DWT, there is only 9 non zero-or-one element in weighting vector and half of the weighting vector element are zero regardless the array size. For 2-D DHT, there is only one non zero element in weighting vector regardless the array size. The computation cost to calculate input checksum in each row/column can be further reduced due to these specific transform properties. The encoded checksum then needs to go through the 1-D transformation in order to get the output predictive row/column checksum. Finally, the elements in each row/column of the output array are sum up to create output checksum.

2. Algorithm Based Fault Tolerant (ABFT): Since 2-D linear transform can be represented in matrix form, ABFT [39] can be implemented at

each matrix multiplication for protection. However, the computation overhead is big as the encoding is done at each matrix multiplication level rather than 2-D linear transform level. For 2-D linear transform, encoding and detection need to be performed in each column-wise or row-wise 1-D transformation. The naive ABFT implementation cannot take advantage of more efficient transform algorithm. For example, in DWT, matrix implementation can create huge overhead compared to lifting implementation due to its sparse coefficient matrix. Here, the overhead which matrix multiplication incurs is ignored, and only the encoding and checksum creation cost are compared.

3. Parity Weighted Sum (PW): Parity weight sum [63] utilizes iterative code design methodology to design specific weighting vector for each transform application For 2-D linear transform, encoding and detection need to be performed every time when column-wise or row-wise 1-D transformation is calculated. To consider the computation overhead, the weighted parity for output is also calculated.

Table 2.4 summarizes the encoding cost, computation overhead, and total cost of the above calculation. The proposed 2DLTC method is clearly shown to be better than the Parity Weighted Sum (PW) method and ABFT.

For 2DLTC method, it takes N^2 multiplications and $N^2 - N$ additions to compute input array weighted checksum (encoding cost). The computation overhead can be divided into output array checksum calculation ($N^2 - N$ additions) and 1-D transform of input array weighted checksum (N^2 multiplications and $N^2 - N$ additions). There will be hardware overhead to compute input array weighted checksum and output array checksum (N^2 multiplications and $2N^2 - 2N$ additions). The 1-D transform of input array weighed checksum can reuse the existing transformation hardware. The hardware overhead can be traded off with minor performance overhead ($\frac{1}{2N}$ of the total computation latency).

If 16-bit multiplication in the application is considered, the cost of doing one multiplication is roughly equal to doing 15 additions. For a better comparison, the cost of different methods are translated into an equivalent number of additions required. The comparison result is shown in Figure 2.10. It can be seen that the 2DLTC method has the least amount of cost.

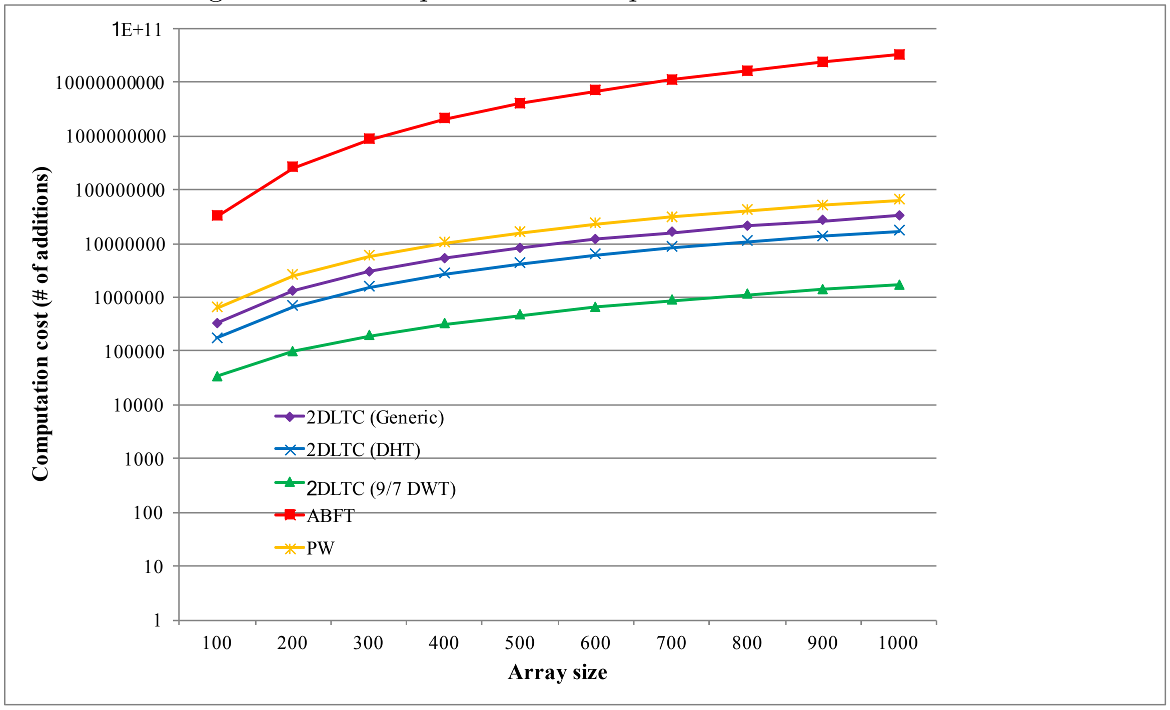
Table 2.4: Comparison of Computation Overhead

Algorithm	Encoding Cost	Computation Overhead	Total Cost
2DLTC (Generic)	N^2 mult, $N^2 - N$ add	N^2 mult, $2N^2 - 2N$ add	$2N^2$ mult, $3N^2 - 3N$ add
2DLTC (DHT)	N mult	N^2 mult, $2N^2 - 2N$ add	$N^2 + N$ mult, $2N^2 - 2N$ add
2DLTC (9/7 DWT)	$9N$ mult, $\frac{N^2}{2} + 4N$ add	$3N$ mult, $N^2 + 3N$ add	$12N$ mult, $\frac{3N^2}{2} + 7N$ add
ABFT	$4N^2 - 4N$ add	$2N^3 - 2N^2$ mult, $2N^3 - 6N^2 - 4N$ add	$2N^3 - 2N^2$ mult, $2N^3 - 2N^2 - 8N$ add
PW	$2N^2$ mult, $2N^2 - 2N$ add	$2N^2$ mult, $2N^2 - 2N$ add	$4N^2$ mult, $4N^2 - 4N$ add

Table 2.5: Comparison of Computation Overhead with different algorithm and array size N. Normalized to number of additions

N	100	200	300	400	500	600	700	800	900	1000
2DLTC (Generic)	3.30E+05	1.32E+06	2.97E+06	5.28E+06	8.25E+06	1.19E+07	1.62E+07	2.11E+07	2.67E+07	3.30E+07
2DLTC (DHT)	1.71E+05	6.83E+05	1.53E+06	2.73E+06	4.26E+06	6.13E+06	8.34E+06	1.09E+07	1.38E+07	1.70E+07
2DLTC (9/7 DWT)	3.37E+04	9.74E+04	1.91E+05	3.15E+05	4.69E+05	6.52E+05	8.66E+05	1.11E+06	1.38E+06	1.69E+06
ABFT	3.17E+07	2.55E+08	8.61E+08	2.04E+09	3.99E+09	6.90E+09	1.10E+10	1.64E+10	2.33E+10	3.20E+10
PW	6.40E+05	2.56E+06	5.76E+06	1.02E+07	1.60E+07	2.30E+07	3.14E+07	4.10E+07	5.18E+07	6.40E+07

Figure 2.10: Comparison of Computation Overhead



2.9.3 Multiple Faults and Common Mode Failures Analysis

Common-mode failures are a special and very important cause of multiple faults. Common-mode failures (CMFs) produce multiple faults, generally occurring due to a single cause [72]. In the presence of CMFs, the system data integrity may not be guaranteed. These include design mistakes and operational failures that may be due to external (such as EMI, power-supply disturbances and radiation) or internal causes.

Redundant systems are subject to common-mode failures (CMFs). Design diversity has been proposed in the past to protect redundant systems against common-mode failures [73]. Unlike systems with duplication, concurrent error detection techniques based on error detection codes introduces inherent diversity in the system. Thus, these systems are well-protected against CMFs The section analyses the CMF vulnerability of the proposed algorithms.

In ABFT [39], it is shown that if an error is located in a row or a column of an output array which does not contain any other error, the error will cause an inconsistent row or column which can be detected. But when the errors are connected to form a loop, the errors may mask each other and cannot be detected.

To analyze the multiple faults detection capability of the proposed algorithm, a given noise source causing an error in a d bits $N \times N$ output array element is assumed. The error is described as correct d bits data having d_e bit flip in the location specified by error vector $[e_{d-1}, \cdots, e_0]$ where $e_i = \{0, 1\}$

and $e_i = 1$ represents bit flip in ith bit location and these bits are generated randomly. The probability that the error in one element masks the sum of the errors in the other elements is less than 2^{-d_e} since the errors are generated randomly. The probability of the undetectable errors in a column or a row $P_{undetect}$ is less than 2^{-d_e} and the probability of the undetectable errors of the whole output array is less than $(P_{undetect})^N = 2^{-d_e \times N}$. The probability of error detection $(P_{detect})_{min}$ is not less than $1 - P_{undetect}$.

The calculation of the minimal error detection probability in multiple faults scenarios is summarized in Table 2.6. Even with error pattern with single bit flip, when array size N is larger than 15, the minimal error detection probability is 99.99695%. As the N increases, the minimal error detection probability converges to 100%. Even with just 2 bits flip in error pattern, the minimal error detection probability converges to 100% even with smaller array size.

Table 2.6: The Minimal Error Detection Probability in Multiple Faults Scenarios Assuming Errors are Randomly Generated

N	$(P_{detect})_{min}$ (d_e=1)	$(P_{detect})_{min}$ (d_e=2)	$(P_{detect})_{min}$ (d_e=3)	$(P_{detect})_{min}$ (d_e=4)	$(P_{detect})_{min}$ (d_e=5)
5	96.87500%	99.90234%	99.99695%	99.99990%	100.00000%
10	99.90234%	99.99990%	100.00000%	100.00000%	100.00000%
15	99.99695%	100.00000%	100.00000%	100.00000%	100.00000%
50	100.00000%	100.00000%	100.00000%	100.00000%	100.00000%
100	100.00000%	100.00000%	100.00000%	100.00000%	100.00000%
1000	100.00000%	100.00000%	100.00000%	100.00000%	100.00000%

Chapter 3

Quality Aware Error Detection Algorithm in 2-D Separable Linear Transform

3.1 Background

Besides reliability, power consumption is also one important design constraint in digital hardware design. However, reliability and low-power design often have conflicting design requirement. Conventional digital design strategies guarantee timing correctness of all corners/conditions. Voltage guard band is often allocated in VLSI systems to tackle different kinds of variation and uncertainty to ensure function correctness. Reducing the supply voltage could introduce timing errors into the design.

One way to achieve low power design is to design circuit at typical corner instead of worst case corner and reduce the voltage margin. Voltage over-scaling (VOS) technique even reduces power by scaling the supply voltage until data-dependent timing errors start to appear. Technology trends show the need for a more efficient error detection technique for VLSI systems. Many DSP applications, like 2-D linear transformation used in the multimedia compression system, do not require exactly correct results, but rather require that the Signal-to-Noise Ratio (SNR) is above a certain threshold. In this chapter, this property is explored and the proposed generic error detection method in the 2-D separable linear transformation is extended to be quality aware to solve the reliability and energy efficiency problems.

3.2 Inherit Error Tolerance in Applications

A wrong output signal produced by a defective system is called an error. Traditionally, an error of the digital systems is defined at the boolean function level. In the classical Von Neumann fault model [74], an unreliable gate is modeled as a gate that with probability $1-p$ $(0 \leq p < \frac{1}{2})$ computes the correct output on its inputs and with probability p, it produces an incorrect output (its binary value is flipped). However, the boolean level error definition may be too pessimistic for some applications which do not require exact correct output.

Multimedia applications are inherent error tolerant. Many computations addressed in these areas focus on good or bounded but not necessary exactly correct results. There are several interesting aspects to the computational requirements for such applications [75].

- The result of computation is not measured regarding being right or wrong, but rather at the perceptual quality. As the output is consumed by a human user, the perceptual quality is defined to determine if the output acceptable to the human user.
- Many such applications are by design lossy, in the sense that the out-

puts deviate from perfection due to the sampling of input signals, conversion to digital, quantization, lossy encoding, decoding and conversion to analog signals. The multimedia information has undergone sampling, quantization, lossy compression, lossy transmission and A/D and D/A conversion. The data conversion may have Boolean values that differ from the ideal. There may be room for still additional "noise" to be added to these signals induced via noisy or unreliable circuitry.

- Many such applications require parallel computation architectures as they are computationally intensive and have real-time performance constraints. Even if some arithmetic unit occasionally produces errors, this unit only processes a small portion of the results and hence its results may not be too detrimental to the overall results from the system.

Breuer proposed the methodology for analysis of error tolerance [76] to increase the effective yield for a given design and domain.

3.3 Related Work

Several efforts in the past have explored the possibility of trading off DSP system quality for lower energy.

Algorithmic Noise Tolerance (ANT) is first introduced in Shanbhag's work [40] [77] to compensate for degradation in the system output due to timing errors introduced by voltage over-scaling. There are many different variations of ANT and each has its application.

In prediction-based ANT [40] [77], a low complexity linear forward predictor is employed to get an estimate of the current sample of the filter output based on its past samples. Error cancellation-based ANT [41] requires a separate filter called the error canceller that generates an estimate of the noise. The error canceller needs to be trained first to learn the correlation structure between noise and signal input. During normal operation, the noisy output is improved by subtracting the estimated noise. In reduced precision redundancy (RPR) based ANT [41], the low precision replica of the main DSP module computes only the MSB of the error free output. If the difference between noisy output and replica is above a predefined threshold, the system will choose the low precision replica as the final output. Input subsampled replica (ISR) ANT [78] is proposed to resolve the situation where the main DSP block has smaller number bit range. In such a case, RPR will lead to an inaccurate result. ISR ANT subsamples the input data using the same precision as the main DSP block, but power consumption is reduced due to operation at a divided frequency. However, ISR ANT estimator is the same size of the main DSP block.

The main disadvantage of ANT is its area cost. All versions of ANT require replicas in different flavors of complexity. The key underlying assumption of ANT is that the error control block is error-free, though this assumption is no longer valid in the presence of soft errors. Algorithmic Soft Error-Tolerance (ASET) [79] based on ANT is also proposed to solve the soft error issue. However, it even incurs a higher cost than ANT.

Chippa and Roy proposed the concept of Dynamic Effort Scaling (DES) [80] which a feedback control framework regulates the effort scaling to maintain output quality at or above a specific limit. DES leverages the time-varying resilience in the application and dynamically navigates the trade-off between output quality and efficiency. The challenge in DES scheme is accurate quality estimation to guide effort scaling. Sensors for quality estimation at circuit, architecture and algorithm levels are reviewed. Circuit-level quality sensors are generic and can be applied to different algorithms. However, critical path based error detection may not correlate well with output quality and could leave a margin for further optimization. Architecture-level sensors can be implemented as datapath reduced precision replica. The area overhead is mitigated with reduced precision replica. Algorithm-level sensors are attractive given its low overhead. Internal variables produced during the computation was proposed to serve as the quality estimator at algorithm level. However, the output quality of the application is input pattern and algorithm dependent. The selection of internal variable is often empirical and not mathematically proven.

ERSA (Error Resilient System Architecture) [81] is a programmable multi-core architecture which combines a few reliable cores with many small unreliable cores to reliably execute probabilistic applications. It uses asymmetric reliability, software optimization, and light-weight checks to overcome the reliability issues. ERSA utilizes algorithmic convergence damping and filtering to control the quality of the algorithm output.

He [82] proposed low energy 2-D IDCT design by controlling timing error induced by voltage over-scaling. Architecture and micro-architecture level implementation techniques like dynamic adder width reduction, dynamic accumulation reordering, and algorithm steps rescheduling are deployed to avoid/reduce the timing error. Although the proposed techniques are general, algorithm classification and study is required to convert the hardware to robust implementation.

3.4 Proposed Algorithm

Applications like multimedia have inherent error resilient. Few errors introduced in an application may not cause noticeable impact if output array meet the quality requirement. The output array quality is often measured by SNR (Signal-to-Noise Ratio). In this work, SNR is defined the same as the definition in Andra's study [83]. Here each data element in golden/revised output array data is one pixel.

$$SNR = 20log_{10}(\frac{\sum |Y_{golden}[i,j]|}{\sum |Y_{golden}[i,j] - Y_{revised}[i,j]|}) \tag{3.1}$$

Let SNR_T be the output array SNR targeted after 2-D linear transform; The upper bound of the sum of the column checksum difference to meet the output array quality requirement can be derived. The equation can be rewritten since the summation of the absolute value of golden/revised array element delta will be greater or equal than the summation of the absolute value of golden/revised array column checksum difference.

$$\frac{\sum |Y_{golden}[i,j]|}{10^{\frac{SNR_T}{20}}} = \sum |Y_{golden}[i,j] - Y_{revised}[i,j]|$$
$$\geq \sum_{j=0}^{N-1} |\sum_{i=0}^{N-1} Y_{golden}[i,j] - \sum_{i=0}^{N-1} Y_{revised}[i,j]| \tag{3.2}$$

Assuming 1-D column-wise linear transform is performed first and followed up 1-D row-wise linear transform, it is easier to calculate the predicted column checksum and the output image column checksum. Following the proof presented in Chapter 2, the equation can be rewritten to be based on the column checksum. Since the creation of column checksum does not take an absolute value, the equation above can be rewritten to,

$$10^{\frac{-SNR_T}{20}} \times ((\sum_{j=0}^{N-1} |f_{r2}(Mf_{r1}((W_C X)^{\mathrm{T}})))^{\mathrm{T}}[j]|) + \xi) \geq$$
$$\sum_{j=0}^{N-1} |f_{r2}(Mf_{r1}((W_C X)^{\mathrm{T}})))^{\mathrm{T}}[j] - \sum_{i=0}^{N-1} Y[i,j]| \tag{3.3}$$

where ξ is the margin to account for mismatch due to the absolute value. This shows if the sum of the column checksum difference between golden and revised output array is less than a fraction of the sum of the golden column checksum, the SNR of the output array can be guaranteed to be within an acceptable range.

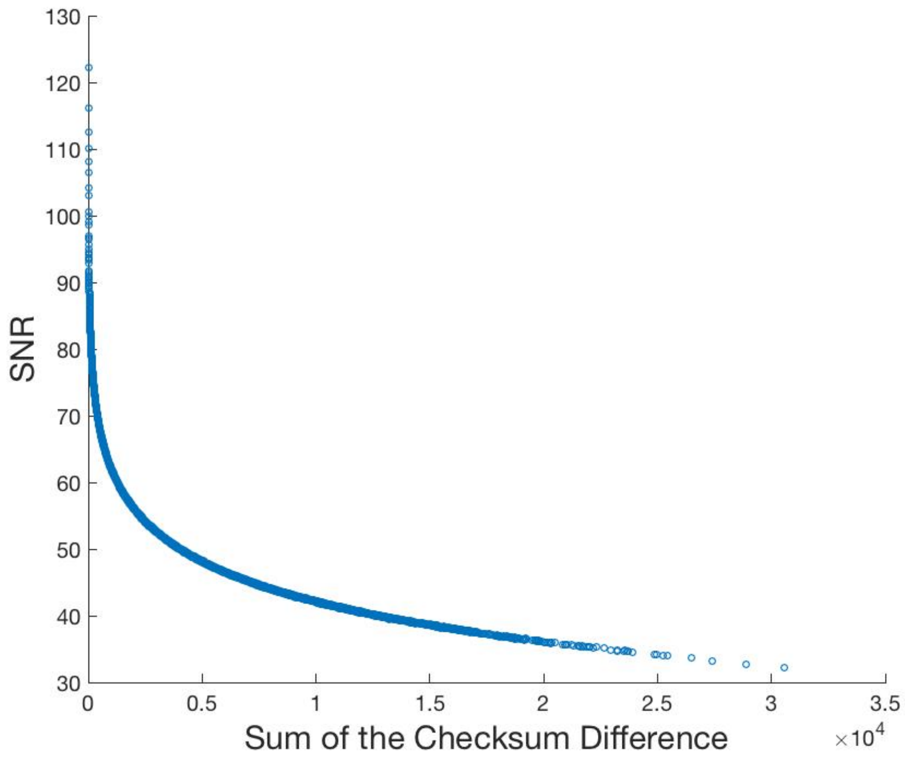
3.5 Correlation between Sum of Checksum Difference and SNR

To verify the proposed algorithm, Matlab simulation is performed to evaluate the correlation between the sum of checksum difference and SNR. A 100×100 array is randomly generated which each array element is ranged from 0 to 255 to emulate the pixel data range. 10000 iterations are run with error injection on random magnitude, location and occurrence on the targeted array. The result is shown in Figure 3.1. As the sum of the checksum difference increases, the SNR reduces exponentially. By setting the required threshold of the sum of the checksum difference in error detection, SNR of the output can be maintained within the acceptable range.

3.6 Quality Aware Error Detection for Low-Power Discrete Wavelet Lifting Transform in JPEG 2000

The JPEG 2000 standard adopted the 5/3 and 9/7 wavelet lifting transforms for implementation. The 9/7 wavelet transform got its name from the fact that the low and high-pass analysis filters have 9 and 7 taps respectively. This transform has been found to yield optimal or near optimal performance in image compression application and has enjoyed widespread popularity in the image compression community. Although the dissertation only discusses the 9/7 wavelet transforms, the same encoding technique can still be applied to other types of wavelet transforms with just the weighting vector being different.

Figure 3.1: Sum of checksum difference v.s SNR



A reliable low-power implementation of the discrete wavelet transform (DWT) is desired for portable and battery operated multimedia devices. In this research, low power DWT is achieved by over-scaling the supply voltage. However, reliability and low-power have conflicting design requirements. Reducing the supply voltage introduces timing errors into the design. Many DSP applications, including DWT, do not require exactly correct results, but rather require that the Signal-to-Noise Ratio (SNR) is below a certain threshold. It is desire to find the optimal operation voltage which achieves low power while maintaining acceptable image quality. Normally, SNR values cannot be obtained on-line, which limits exploration of dynamic voltage scaling for image applications. The weighted checksum code based error detection developed earlier is extended to estimate image SNR at runtime and detect any image quality degradation due to timing errors. This information can then be used to choose the optimal voltage setting in dynamic voltage scaling. Since power reduction is achieved by reducing the supply voltage, and the error detection scheme is independent of the underlying DWT architecture, this technique can be applied to existing low power DWT architectures to save additional power.

3.7 SNR-Aware DWT Architecture

The weighted checksum error detection technique is independent of the underlying DWT architecture. It can be used in any existing DWT architecture to enhance design reliability to overcome process variations and provides extra power saving via voltage-over-scaling. To demonstrate the idea, the low-cost detector is implemented on the general purpose 2-D DWT hardware based on the architecture proposed by Andra [83] with minor modifications to simplify the implementation. The architecture calculates the DWT in row-column fashion on a block of data of size $N * N$. To perform the DWT, the hardware reads in the block of data, carries out the transform, and outputs the LH/HL/HH sub-band data at each level of decomposition. The LL sub-band is used for the next level of decomposition. Figure 3.2 shows the block diagram of SNR-aware DWT architecture. RP/CP represent row/column processors, MEM1/2 are on-chip SRAMs and REG1/2 are register files. The checker processor is a new module introduced in addition to the original DWT architecture. The row processor and column processor have the same micro-architecture structure. Some modifications are done in row and column processors so that the same hardware can be reused in the weighting multiplication phase to reduce the checker hardware overhead. Figure 3.3 shows the block diagram of the row/column processor. Figure 3.4 shows the block diagram of the checker processor.

Without checking, the original computation takes 267296 cycles to finish transforming one 512x512 grayscale image, which translates to 165.54M samples/sec. (design cycle time is 5650ps). To parallelize the checking task and reduce the hardware overhead, the overall computation is divided into 5 operation phases. The cycle numbers shown below are based on the time to transform one $512 * 512$ gray scale image.

(1) *Weighting Multiplication:* Each row/column processor is config-

ured to be weighting multiplication mode to multiply 9 none-zero or none-one weighting coefficients with the corresponding input data and accumulate the result to form a partial checksum. Each row/column processor is responsible for calculating a fourth of the columns input data. The temporary column checksum is stored into MEM1. It takes 2560 cycles to complete this step. In Figure 3.3, the shaded component indicates that it is additional to the original architecture. To further reduce the hardware overhead, the low-cost checksum is implemented by dividing the input data by 16 for checksum calculation (shifter1 right shifts the data by 4-bit).

(2) *Column-Wise 1-D DWT Phase:* Row/column processors are configured to operate in functional mode. The row/column processors take input data and perform 1-D DWT. The checker processor takes even samples of input data, divided by 16 (right shifts by 4-bits) and continues to calculate the input data column checksum by adding the partial checksum read from MEM1. Besides the one multiplied by the weighting factor in the previous phase, odd samples do not need to be considered in checksum calculation since their weighting factors are zero. After completion, the input data checksum is stored back into MEM1. It takes 133648 cycles to complete this step. The checker processor is over-designed to make sure it can operate in a voltage-over-scaling condition without timing errors.

(3) *Output Checksum Prediction:* The input weighted checksum stored in MEM1 is fed into the wavelet transform pipeline to be treated as an additional row to calculate the predicted output checksum. After completion, the

predicted output checksum is stored in MEM2. It takes 520 cycles to complete this step.

(4) *Row-Wise 1-D DWT Phase:* Row/column processors are in functional mode. The processors take input data and perform 1-D row-wise DWT. The checker processor takes high/low pass output data divided by 16 to calculate the output row checksum (after a transpose, it becomes the final result column checksum). It takes 133648 cycles to complete this step.

(5) *Final Check Phase:* The checker processor calculates the sum of the predicted column checksum and the sum of column checksum differences. Then the sum of the column checksum difference is compared with the divided version of the sum of the predicted column checksum to check the image quality. It takes 1034 cycles to complete this step.

To enforce the correctness of the checking process in weighting multiplication, output checksum prediction, and final check phases, the clock frequency is divided by 2 to avoid any timing error. The cycle number shown above already considers this latency overhead. Overall, the checker takes an additional 4114 cycles to compute, this impacts the overall latency by about 1.54%. This is achieved by only adding checker processor hardware and some control logic overhead. For color images, each plane Y/Cb/Cr will take the same amount of computation time as one gray scale image. Additionally, one extra step is taken to add up the predicted checksum and checksum difference from all 3 planes.

Figure 3.2: SNR-Aware DWT Architecture Top Level Block Diagram

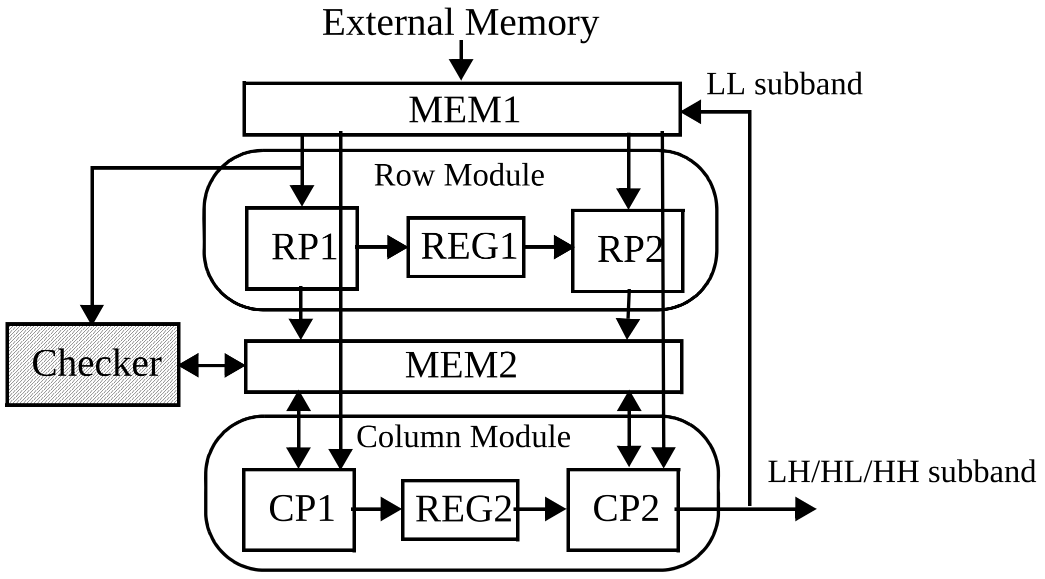


Figure 3.3: SNR-Aware DWT Architecture Row/Column Processor Block Diagram

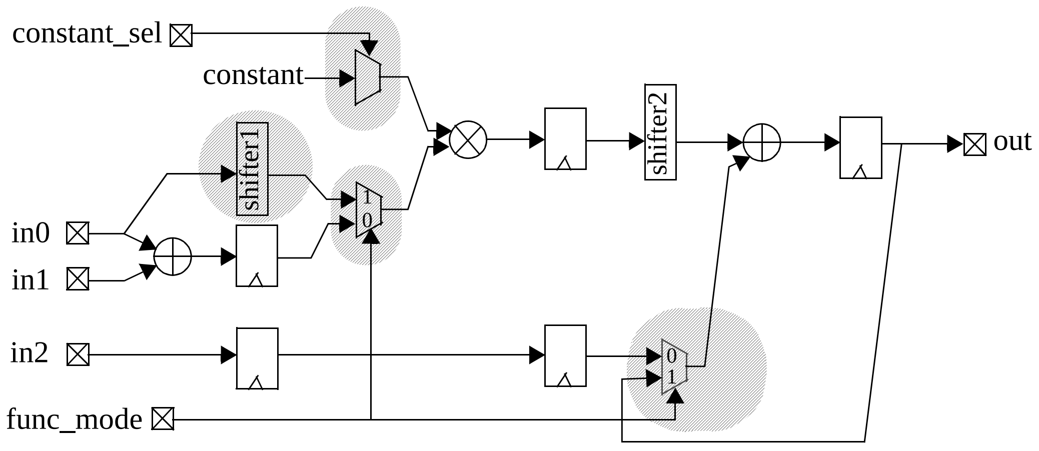
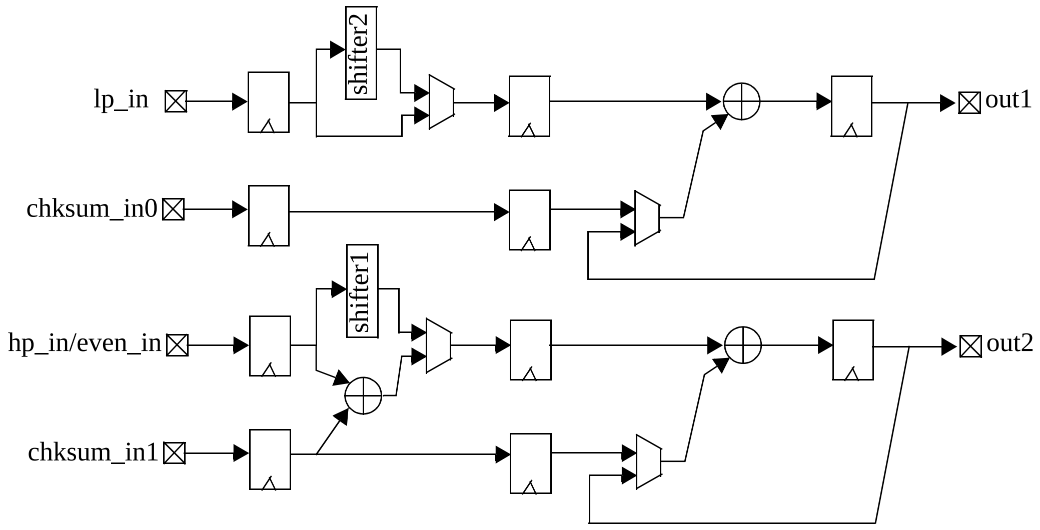


Figure 3.4: SNR-Aware DWT Architecture Checker Processor Block Diagram



3.8 Simulation Framework

To demonstrate the idea, a floating-point Matlab model of the DWT architecture is developed to verify the correctness of the algorithm. A fixed-point Matlab model is developed as a golden reference for the Verilog model and is also used to study the accuracy of the fixed-point implementation.

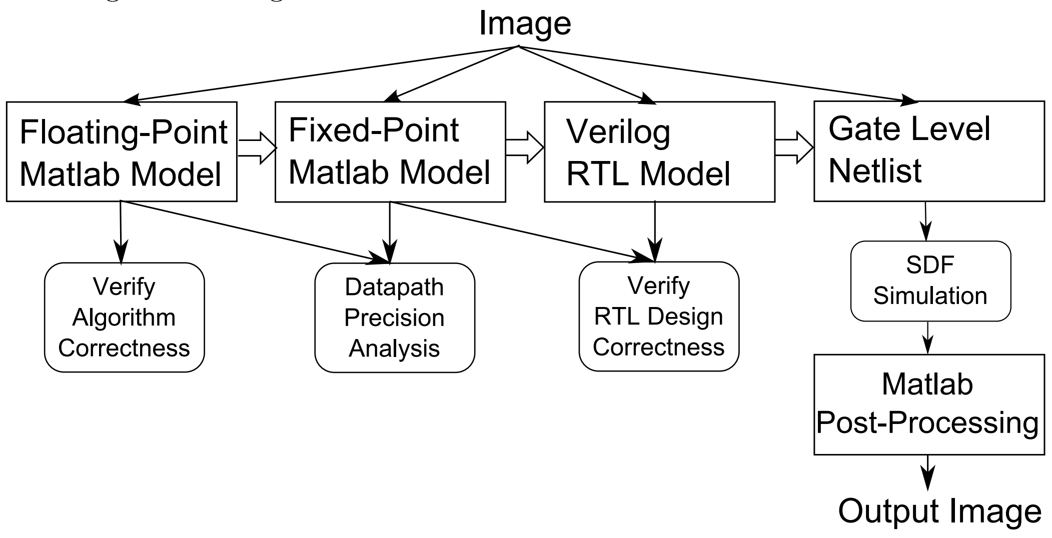
The precision analysis method in Andra's work [83] is followed in this study. The pixels in RGB format are ranged from 0 to 255. In JPEG, the pixel data in RGB format is converted to YCbCr format where Y ranges from 16 to 235, Cb ranges from 16 to 240, and Cr ranges from 16 to 240. To get the desire image quality in the fixed-point implementation, the input pixel data value is scaled by 32. The filter coefficients are multiplied by 512 and

$a = -812$, $b = -27$, $c = 452$, $d = 227$, $k1 = 416$ and $k2 = 314$ can be derived.

The signal dynamic range in the intermediate DWT pipelines is calculated at each stage to determine the optimal datapath width. A 20-bit (1-bit sign and 19-bit value) datapath is chosen and implemented in the design. The Verilog model is then developed and compared against the result from the fixed-point Matlab model to validate model correctness.

The design was synthesized by Synopsys Design Compiler using the Faraday cell library in UMC $0.13\mu m$ technology. The area of the design is 228520 μm^2. Standard delay format (SDF) files, which contain actual delay information for worst/typical/best corners, are exported from the timing tool and back-annotated into the gate-level Verilog model for dynamic timing analysis. To model the effect of voltage-over-scaling on signal propagation delay, the delay in the SDF file is scaled by a certain ratio according to the scaled voltage. The ratio is characterized by running Spectre simulation on a ring oscillator with different supply voltages in different process corners compared to the delay in $1.2v$ supply voltage. The dt/dv ratio is approximately 1.6 while the ring oscillator is in the typical corner. The characteristic result is shown in Figure 3.6. The SDF Verilog simulation output is then fed back to Matlab post-processing simulation to display result image and calculate actual SNR. The algorithm to gate level end-to-end simulation flow is shown at Figure 3.5.

Figure 3.5: Algorithm to Gate Level End-to-End Simulation Flow



3.9 Simulation Results

Seven images from the USC-SIPI image database [84] have been simulated in both Matlab and the gate level Verilog model to validate the idea. The information of these images is shown in Table 3.1. Gate level dynamic timing analysis with different supply voltages and process corners are performed on those images. Figure 3.10(a) to 3.10(d) show the simulation result of the relationship between the supply voltage scaling and the sum of the checksum differences/SNR in typical and best case corner. More specifically, the sum of checksum differences increased significantly where the SNR of those transformed images has a drastic drop in the simulation results. Although only the result of two images is shown here, all seven images show a similar trend for the sum of checksum differences and SNR. The SNR for a decompressed image is higher than the SNR for a compressed image, showing that the image compression application is inherently error tolerant. This gives a little more room to further push the supply voltage down. Figure 3.11(a) to 3.12(d) shows the decompressed image compressed by hardware running at different supply voltage in the typical corner.

Figure 3.7 shows a strong negative correlation between the sum of checksum differences and the SNR of the transformed images. This demonstrates that the sum of checksum differences is an efficient metric to predict the SNR of the transformed images. In the Verilog implementation, the sum of the checksum differences in error-free conditions is not zero due to the floating-point to fixed-point conversion. The effective checksum mismatch in each pixel is about 0.8%, which is acceptable. The reason for the high slope around $SNR = 30$ is due to this fixed offset.

Due to design tool limitation and large size of input samples, direct power consumption simulation is not feasible. For a 512×512 image input, there are 262144 samples which is not feasible to simulate via SPICE. Also, the standard cell timing libraries only contains the timing information for support voltage supply. To solve the above limitations, the scaling method is used to estimate the power consumption in different supply voltage condition. The power consumption of the proposed design in each voltage setting is estimated by scaling the typical corner power simulated in Synopsys Design Compiler. The total dynamic power of the proposed design is $16.34mW$ in the typical corner. The dynamic power scaling is performed by scaling the voltage using the equation $P = \alpha CV^2$. The equivalent voltage in each simulation corner is scaled back to the nominal voltage. The result shows the supply voltage can be scaled down to 75% of the nominal voltage in the typical corner without significant image quality degradation, which translates to $9.15mW$ power consumption (44% power saving). In the best case corner, the supply voltage can be further scaled down to 60% of the best case voltage which translates to $8.16mW$ power consumption (64% power saving). Table 3.2 shows the minimal supply voltage in the typical corner to maintain the decompressed image SNR greater than 30.

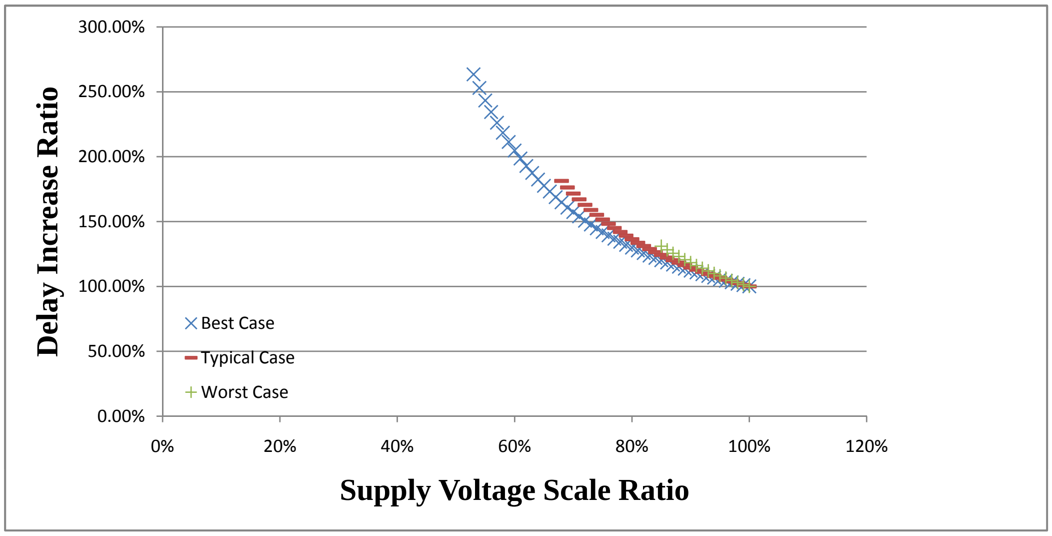
Table 3.1: Images Used in Simulation

Image	Description	Size	Type
4.2.01	Splash	512	Color
4.2.02	Girl(Tiffany)	512	Color
4.2.03	Mandrill	512	Color
4.2.04	Girl (Lena)	512	Color
4.2.05	Airplane (F-16)	512	Color
4.2.06	Sailboat on lake	512	Color
4.2.07	Peppers	512	Color

Table 3.2: Min. Voltage in Typ. Corner for Decompressed Image SNR > 30

Image	Min. Voltage	Image SNR (decompressed)	Image SNR (compressed)	Sum of Checksum Difference
4.2.01	0.89	32.824	20.2491	183424
4.2.02	0.86	34.6052	15.1434	287879
4.2.03	0.86	34.9199	22.4807	170027
4.2.04	0.86	30.8851	14.376	301766
4.2.05	0.86	30.5887	8.7766	522223
4.2.06	0.86	32.9012	17.4348	228068
4.2.07	0.86	32.0065	17.1362	225877

Figure 3.6: Delay variation due to scaled voltage



3.10 Conclusions

A generic quality aware 2-D linear transformation error detection algorithm is presented. Matlab simulation shows the correlation between the sum of checksum difference and SNR. The inherent error tolerance in the application is discussed and related works are reviewed. The key element needed to leverage inherent error for low power operation is the online quality estimator. The proposed method can be used as low-cost online quality estimator to determine the optimal supply voltage to enable precise dynamic voltage scaling.

2-D DWT architecture in the JPEG 2000 standard is selected as an example to demonstrate the SNR-aware error detection and voltage-over-scaling capability. A low-cost weighted checksum is used to estimate image SNR. If

Figure 3.7: Correlation between SNR and Sum of the checksum difference

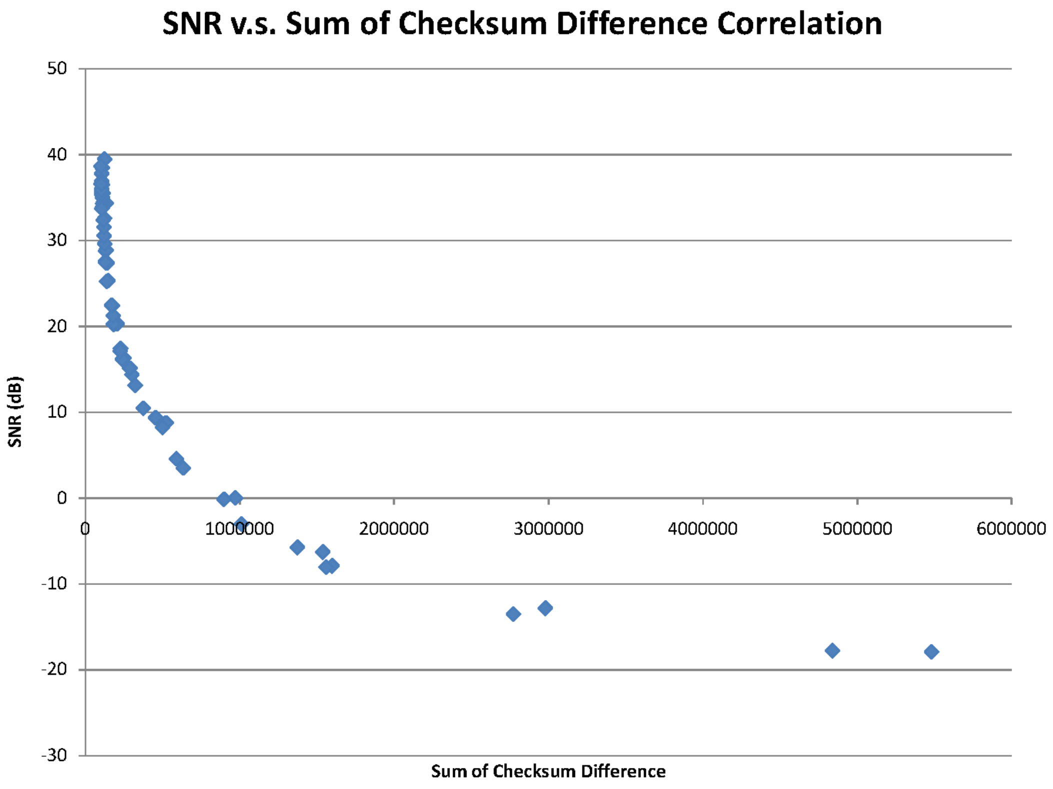
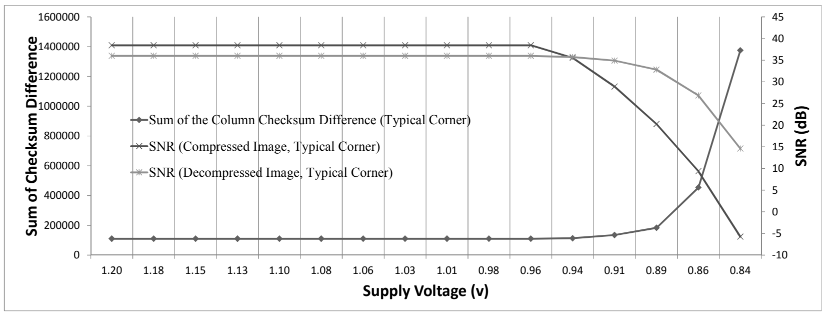
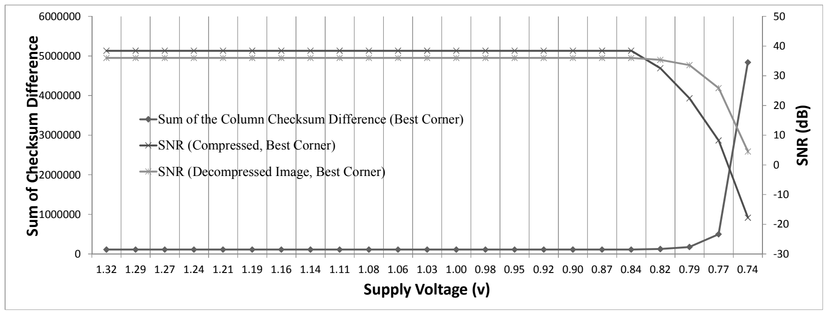


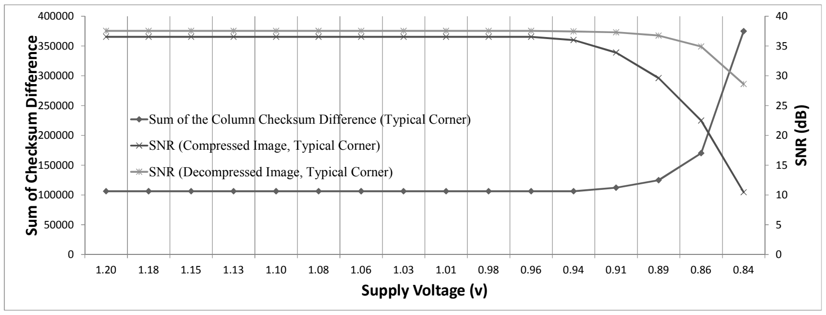
Figure 3.8: Supply voltage scaling v.s. SNR/Sum of checksum difference for image 4.2.01/4.2.03



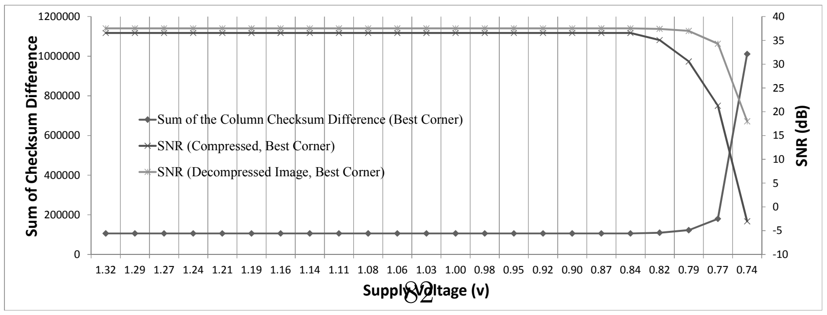
(a) 4.2.01 Typ. case process corner



(b) 4.2.01 Best case process corner

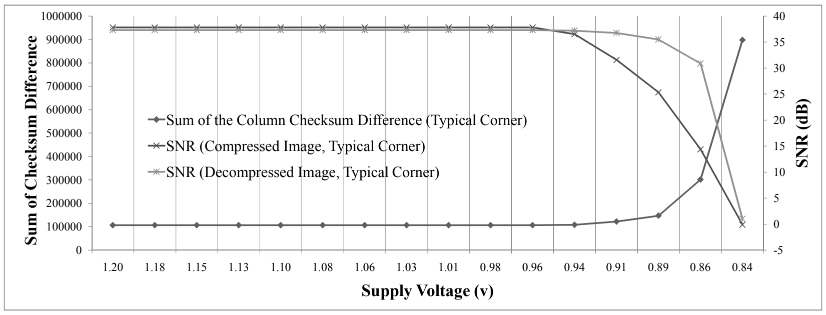


(c) 4.2.03 Typ. case process corner

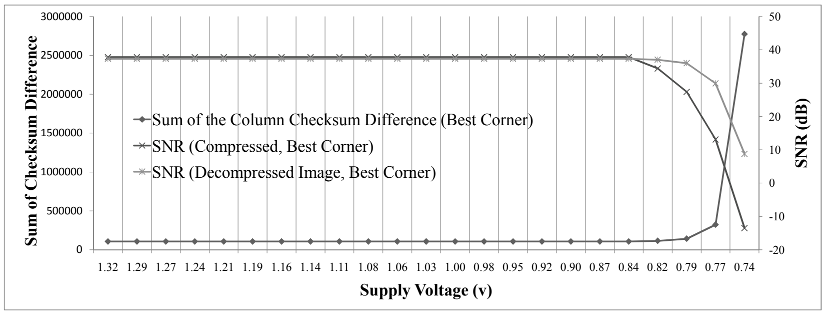


(d) 4.2.03 Best case process corner

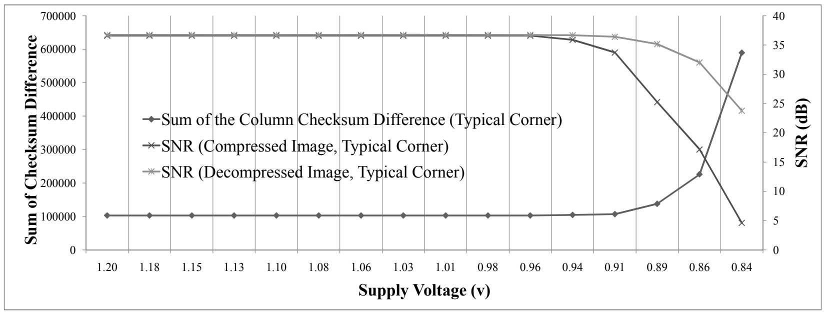
Figure 3.9: Supply voltage scaling v.s. SNR/Sum of checksum difference for image 4.2.04/4.2.07



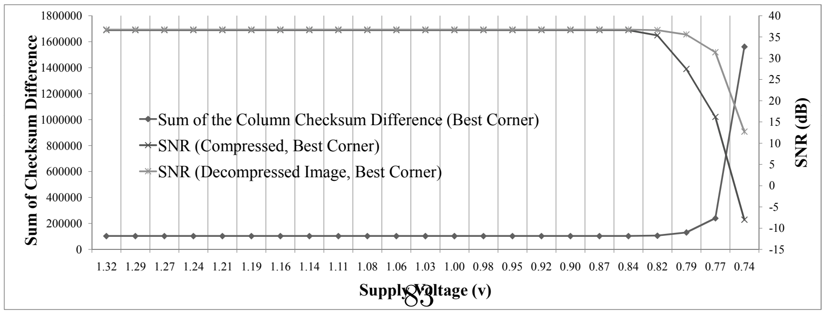
(a) 4.2.04 Typ. case process corner



(b) 4.2.04 Best case process corner

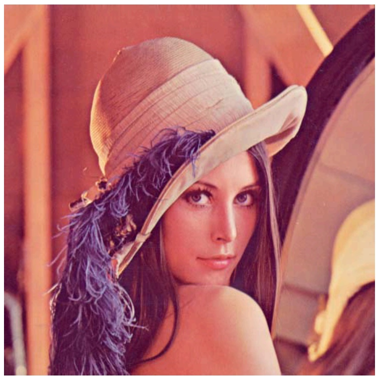


(c) 4.2.07 Typ. case process corner

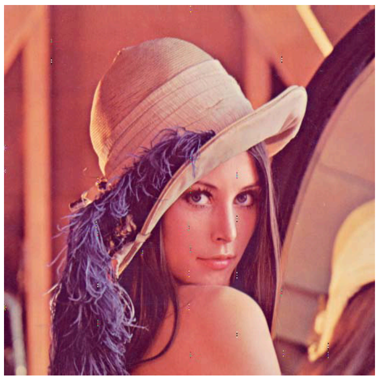


(d) 4.2.07 Best case process corner

Figure 3.10: Voltage scaling impact to image quality in typical corner for image 4.2.04


(a) 1.2V, SNR=37.3

(b) 0.96V, SNR=37.3


(c) 0.912V, SNR=36.757

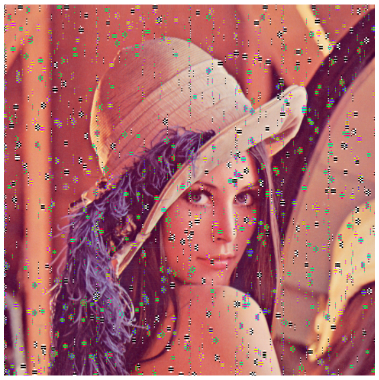
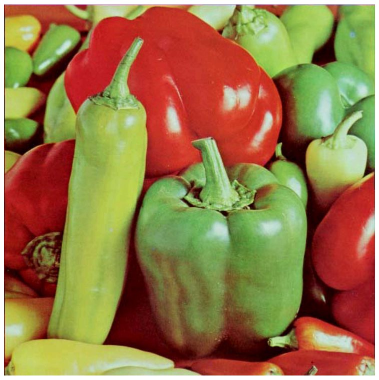
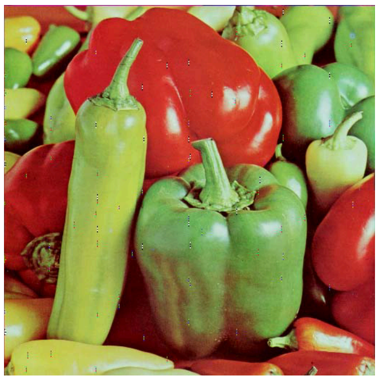
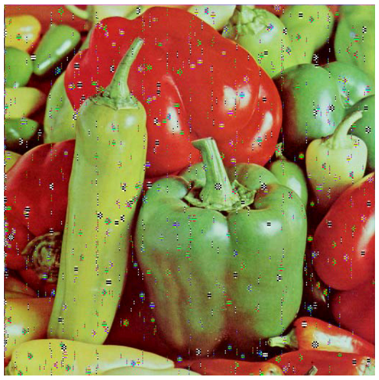

(d) 0.84V, SNR=1.0134

Figure 3.11: Voltage scaling impact to image quality in typical corner for image 4.2.07


(a) 1.2V, SNR=36.7444

(b) 0.96V, SNR=36.7444


(c) 0.888V, SNR=35.153


(d) 0.84V, SNR=23.7729

the checksum difference is below a certain threshold, the decompressed image quality can be guaranteed. The existing wavelift HW architecture is revised to add checksum generation and checker logic with small overhead.

The simulation result showed that the supply voltage can be scaled down to 75% of the nominal voltage in typical process corner without significant image quality degradation, which translates to $9.15mW$ power consumption (44% power saving).

Chapter 4

Conclusions and Future Work

4.1 Conclusion

As the process technology continuously shrinks the transistor geometry and improves device performance and power, reliability becomes the major challenge for current and future VLSI design. Static variations cause a wide distribution of transistor threshold voltages could result to timing errors in the case of test escape. The time and context varying dynamic variations such as temperature variation and supply voltage variation affect circuit performance based on workload. Infrequent worst-case voltage droop could cause timing errors and impact stability. Aging degradation throughout the product life cycle jeopardizes the functionality of the system over time. The reduction in critical charge of logic circuits makes the circuit more susceptible to soft error. Concurrent error detection is essential for digital VLSI system to tackle these instability factors in the field.

On the other hand, reliability and power have contradicted design requirement. Voltage margin (guardband) which is often allocated to account for variation and uncertainty causes significant power impacts. A low power design technique like voltage over scaling (VOS) scales down the voltage to reduce power consumption until data-dependent timing error occurs in systems. Concurrent error detection at different design levels (circuit, architecture and algorithm) can be used as quality estimator to guide adaptive voltage scaling and achieve optimal performance/power/quality operation point.

In this work, a low-cost weighted checksum code based error detection algorithm (2DLTC) for generic 2-D linear separable transform has been proposed. The technique encodes the input array at 2-D linear transformation level, and algorithms are designed to operate on encoded data and produce encoded output data. The proposed error detection technique is a system-level method. The scheme primarily leverages the interchangeable property of column/row-wise linear transform and special patterns in the transform coefficient matrix. The mathematics proof of the coding technique is presented. It shows the proposed 2DLTC technique can detect the errors at 2-D linear separable transforms system level. The hardware implementation has very little overhead and can perfectly fit into the existing 2-D linear separable transform VLSI implementation given the system level encoding/decoding which is independent of the hardware architecture/implementation selection. The generation of weighted checksum code is discussed. The weighted checksum code for different 2-D linear transforms is presented in mathematics form. Cost analysis shows this technique provides error detection capability with the least hardware overhead compared to prior work. In the 2-D DHT, there is only one non-zero element in weighted checksum vector regardless the array size. In the 2-D DWT, there are only nine non-zero or one element in weighted checksum vector regardless the array size. The above transformation specific property can dramatically reduce the encoding cost and make the proposed error detection algorithm even more attractive. Multiple faults and common mode failures are studied. The proposed technique can detect multiple faults at the minimal error detection probability 99.99% if the array size is larger than 15 or there is more than 2 bits flip in error pattern.

For multimedia and RMS (Recognition, Mining and Synthesis) application, the exact precise output is often not needed. The result of the computation is measured at perceptual quality and many of them are by design lossy. The traditional boolean level error definition is too pessimistic for this type of the application. In this work, a quality aware error detection extension of the checksum check is presented. It is shown as long as the sum of the column checksum delta can be constrained within a certain threshold, output array quality can be controlled within an acceptable range.

To demonstrate the concept, the quality-aware error detection hardware on the low power 2-D DWT architecture used in JPEG2000 standard has been implemented. The low-cost weighted checksum is extended to estimate image SNR. If the checksum difference is below a certain threshold, the decompressed image quality can be guaranteed. Based on the quality estimator output from checker unit, the supply voltage can be adjusted to optimal point considering quality/power balance. Unlike traditional DVFS techniques utilize a delay chain or a lookup table to determine the minimum voltage necessary to guarantee error-free operation at a particular frequency [85]. With algorithm level quality estimator, system controller can optimize voltage and enable more aggressive power saving. The proposed method shows that the supply voltage of the 2-D DWT wavelift hardware can scale down to 75% of the nominal voltage in typical process corner without significant image quality degradation The design consumes $9.15mW$ power only which translates to 44% power saving.

4.2 Future Work

Future work is to develop more comprehensive algorithm or architecture level techniques and design methodologies for reliable and energy efficient digital systems design to tackle the challenge posted by process scaling.

4.2.1 Online Quality Estimator for RMS application

It is well known that the RMS application does not need exact result correctness. However, lacking a trustworthy quality estimator for the computation prevents the exploration of potential power savings for an application. It is still possible that the errors introduced in critical location will cause application level errors. The massive amount of computation in RMS applications makes naive error protection extremely cost ineffective. Prior work [86] in this area often uses an empirical approach to find the internal variables to monitor instead of the mathematical approach in this dissertation. The empirical approach is not a solid proof and has the major drawback. The quality of the application output depends on not only the computation itself but also the input vectors. There could be a possibility that input/computation combinations are not being considered and this could lead to incorrect outputs.

4.2.2 Fault Tolerance Algorithm for Approximate Computing

Approximate computing which trades off output accuracy for significant gains in energy efficiency is an emerging research filed. However, the challenge remains for adoption of approximate computing in industry products. To extend the scope of approximate computing beyond multimedia and signal processing, it is necessary that the allowed error or the required minimum precision of the application is either known beforehand or reliability determined online to deliver trustworthy and useful results. Errors outside the allowed range have to be reliably detected and tackled by appropriate fault tolerance measures. To have reliable result in approximate computing, it is necessary to either prove that the approximation is acceptable offline, or a concurrent error detection technique needs to be implemented to enforce the quality online. In the case of proving the technique offline, the performance and accuracy of the application must be confirmed at design time, and be fully tested in post-silicon to guarantee the capability. In online testing case, the effort spent on each iteration or instance of the computation can be adjusted based on the quality metric indicator. Fault tolerance algorithm for approximate computing [51] is the key to make the design methodology to be widely used.

4.2.3 Robust Energy Efficient DSP Architecture and Design Methodology

The objective for this research is to develop robust energy efficient DSP architecture and design methodology for multimedia and RMS (Recognition, Mining, and Synthesis) application. These application have inherent error tolerant property. Previous research consider only low power, means only do voltage scaling, given a throughput requirement, both voltage and frequency scaling can be considered to minimize the energy consumption.

The area overhead for ANT approach is big. It is desire to develop arithmetic code based end-to-end solution which leverage the existing computation in the algorithm for concurrent error detection. Given the DSP algorithm specification (quality and throughput requirement), the goal is to develop a methodology to achieve the lowest possible energy consumption.

4.2.4 Control and Data Flow Graph Level Optimization for Energy Efficient and Resilience Computation

Application can be described as control and data flow graph (CDFG). A CDFG exhibits data dependencies inside basic blocks and captures the control flow between those basic blocks. The high level information embedded in CDFG may be analyzed to explore the cross layer optimization for reliability and energy efficiency enhancement. The information can be used for better design partitioning between control and data flow functions. Different error detection techniques can be deployed on control and data flow functions separately. The control function can be also over-designed to guarantee the function correctness. Different techniques can be employed to relax the data flow function requirement as long as the performance requirement is met.

Bibliography

[1] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter variations and impact on circuits and microarchitecture," in *Proceedings of the 40th Annual Design Automation Conference*, pp. 338–342, ACM, 2003.

[2] S. Borkar, "Designing reliable systems from unreliable components: the challenges of transistor variability and degradation," *IEEE Micro*, vol. 25, no. 6, pp. 10–16, 2005.

[3] K. L. Shepard and V. Narayanan, "Noise in deep submicron digital design," in *International Conference on Computer Aided Design*, pp. 524–531, 1996.

[4] K. L. Shepard and V. Narayanan, "Conquering noise in deep-submicron digital ics," *IEEE Design & Test of Computers*, vol. 15, no. 1, pp. 51–62, 1998.

[5] J. A. Davis, R. Venkatesan, A. Kaloyeros, M. Beylansky, S. J. Souri, K. Banerjee, K. C. Saraswat, A. Rahman, R. Reif, and J. D. Meindl, "Interconnect limits on gigascale integration (gsi) in the 21st century," *Proceedings of the IEEE*, vol. 89, no. 3, pp. 305–324, 2001.

[6] A. H. Ajami, K. Banerjee, A. Mehrotra, and M. Pedram, "Analysis of ir-drop scaling with implications for deep submicron p/g network designs," in *Proceedings. Fourth International Symposium on Quality Electronic Design*, pp. 35–40, IEEE, 2003.

[7] Y.-S. Chang, S. K. Gupta, and M. A. Breuer, "Analysis of ground bounce in deep sub-micron circuits," in *15th IEEE VLSI Test Symposium*, pp. 110–116, IEEE, 1997.

[8] O. Amusan, A. Sternberg, A. Witulski, B. Bhuva, J. Black, M. Baze, and L. Massengill, "Single event upsets in a 130 nm hardened latch design due to charge sharing," in *45th Annual IEEE International Reliability Physics Symposium Proceedings*, pp. 306–311, IEEE, 2007.

[9] Y. Lu, L. Shang, H. Zhou, H. Zhu, F. Yang, and X. Zeng, "Statistical reliability analysis under process variation and aging effects," in *Proceedings of the 46th Annual Design Automation Conference*, pp. 514–519, ACM, 2009.

[10] A. H. Johnston, "Scaling and technology issues for soft error rates," in *Proceeding of the 4th Annual Conference on Reliability*, Citeseer, 2000.

[11] A. V. Mezhiba and E. G. Friedman, "Scaling trends of on-chip power distribution noise," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 4, pp. 386–394, 2004.

[12] S. S. Mukherjee, J. Emer, and S. K. Reinhardt, "The soft error problem: An architectural perspective," in *11th International Symposium on High-Performance Computer Architecture*, pp. 243–247, IEEE, 2005.

[13] C.-L. Chen and M. Hsiao, "Error-correcting codes for semiconductor memory applications: A state-of-the-art review," *IBM Journal of Research and Development*, vol. 28, no. 2, pp. 124–134, 1984.

[14] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, and L. Alvisi, "Modeling the effect of technology trends on the soft error rate of combinational logic," in *DSN*, pp. 389–398, IEEE Computer Society, 2002.

[15] K. Bowman, J. Tschanz, C. Wilkerson, S.-L. Lu, T. Karnik, V. De, and S. Borkar, "Circuit techniques for dynamic variation tolerance," in *Proceedings of the 46th Annual Design Automation Conference*, pp. 4–7, ACM, 2009.

[16] Y. Zu, W. Huang, I. Paul, and V. J. Reddi, "Ti-states: Processor power management in the temperature inversion region," in *The 49th Annual IEEE/ACM International Symposium on Microarchitecture*, IEEE, 2016.

[17] V. J. Reddi, D. Z. Pan, S. R. Nassif, and K. A. Bowman, "Robust and resilient designs from the bottom-up: Technology, cad, circuit, and system issues," in *17th Asia and South Pacific Design Automation Conference*, pp. 7–16, IEEE, 2012.

[18] S. Hamdioui, M. Nicolaidis, D. Gizopoulos, A. Grasset, G. Guido, and P. Bonnot, "Reliability challenges of real-time systems in forthcoming technology nodes," in *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 129–134, EDA Consortium, 2013.

[19] S. Pae, J. Maiz, C. Prasad, and B. Woolery, "Effect of bti degradation on transistor variability in advanced semiconductor technologies," *IEEE Transactions on Device and Materials Reliability*, vol. 8, no. 3, pp. 519–525, 2008.

[20] G. Groeseneken, R. Degraeve, B. Kaczer, and P. Roussel, "Recent trends in reliability assessment of advanced cmos technologies," in *Proceedings of the 2005 International Conference on Microelectronic Test Structures, 2005. ICMTS 2005.*, pp. 81–88, IEEE, 2005.

[21] C. R. Lefurgy, A. J. Drake, M. S. Floyd, M. S. Allen-Ware, B. Brock, J. A. Tierno, and J. B. Carter, "Active management of timing guardband to save energy in power7," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 1–11, ACM, 2011.

[22] R. Hegde and N. R. Shanbhag, "A voltage overscaled low-power digital filter ic," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 2, pp. 388–391, 2004.

[23] F. Sellers, M.-Y. Hsiao, and L. W. Bearnson, *Error Detection Logic for Digital Computers*. McGraw-Hill Book Company, 1968.

[24] J. E. Smith and G. Metze, “Strongly fault secure logic networks,” *IEEE Transactions on Computers*, vol. 100, no. 6, pp. 491–499, 1978.

[25] I. Koren and C. M. Kirshna, *Fault-Tolerant Systems*. Morgan Kaufmann, 2007.

[26] R. E. Lyons and W. Vanderkulk, “The use of triple-modular redundancy to improve computer reliability,” *IBM Journal of Research and Development*, vol. 6, no. 2, pp. 200–209, 1962.

[27] J. J. Shedletsky *et al.*, “Error correction by alternate-data retry,” *IEEE Transactions on Computers*, vol. 27, no. 2, pp. 106–112, 1978.

[28] J. H. Patel and L. Y. Fung, “Concurrent error detection in alu’s by recomputing with shifted operands,” *IEEE Transactions on Computers*, vol. 100, no. 7, pp. 589–595, 1982.

[29] A. Avizienis, “Arithmetic error codes: Cost and effectiveness studies for application in digital system design,” *IEEE Transactions on Computers*, vol. 100, no. 11, pp. 1322–1331, 1971.

[30] D. Ernst, N. S. Kim, S. Das, S. Pant, R. R. Rao, T. Pham, C. H. Ziesler, D. Blaauw, T. M. Austin, K. Flautner, and T. N. Mudge, “Razor: A low-power pipeline based on circuit-level timing speculation,” in *MICRO*, pp. 7–18, ACM/IEEE, 2003.

[31] T. M. Austin, D. Blaauw, T. N. Mudge, and K. Flautner, "Making typical silicon matter with razor," *IEEE Computer*, vol. 37, no. 3, pp. 57–65, 2004.

[32] M. Zandrahimi, Z. Al-Ars, P. Debaud, and A. Castillejo, "Challenges of using on-chip performance monitors for process and environmental variation compensation," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1018–1019, IEEE, 2016.

[33] E. Rotenberg, "Ar-smt: A microarchitectural approach to fault tolerance in microprocessors," in *Fault-Tolerant Computing, 1999. Digest of Papers. Twenty-Ninth Annual International Symposium on*, pp. 84–91, IEEE, 1999.

[34] C. LaFrieda, E. Ipek, J. F. Martinez, and R. Manohar, "Utilizing dynamically coupled cores to form a resilient chip multiprocessor," in *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*, pp. 317–326, IEEE, 2007.

[35] S. S. Mukherjee, M. Kontz, and S. K. Reinhardt, "Detailed design and evaluation of redundant multi-threading alternatives," in *Proceedings of the 29th Annual International Symposium on Computer Architecture*, pp. 99–110, IEEE, 2002.

[36] N. Oh, P. P. Shirvani, and E. J. McCluskey, "Error detection by duplicated instructions in super-scalar processors," *IEEE Transactions on Reliability*, vol. 51, no. 1, pp. 63–75, 2002.

[37] G. A. Reis, J. Chang, N. Vachharajani, R. Rangan, and D. I. August, "Swift: Software implemented fault tolerance," in *Proceedings of the International symposium on Code generation and optimization*, pp. 243–254, IEEE Computer Society, 2005.

[38] T. M. Austin, "Diva: A reliable substrate for deep submicron microarchitecture design," in *Microarchitecture, 1999. MICRO-32. Proceedings. 32nd Annual International Symposium on*, pp. 196–207, IEEE, 1999.

[39] K.-H. Huang and J. A. Abraham, "Algorithm-based fault tolerance for matrix operations," *IEEE Transactions on Computers*, vol. 33, no. 6, pp. 518–528, 1984.

[40] R. Hegde and N. R. Shanbhag, "Energy-efficient signal processing via algorithmic noise-tolerance," in *International symposium on Low Power Electronics and Design* (F. N. Najm, J. Cong, and D. Blaauw, eds.), pp. 30–35, ACM, 1999.

[41] N. R. Shanbhag, "Reliable and energy-efficient digital signal processing," in *Design Automation Conference*, pp. 830–835, ACM, 2002.

[42] R. Hegde and N. R. Shanbhag, "A low-power digital filter ic via soft DSP," in *IEEE Conference on Custom Integrated Circuits*, pp. 309–312, IEEE, 2001.

[43] N. R. Shanbhag, R. A. Abdallah, R. Kumar, and D. L. Jones, "Stochastic computation," in *Proceedings of the 47th Design Automation Conference*,

pp. 859–864, ACM, 2010.

[44] G. V. Varatkar and N. R. Shanbhag, “Error-resilient motion estimation architecture,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 10, pp. 1399–1412, 2008.

[45] G. Karakonstantis, N. Banerjee, K. Roy, and C. Chakrabarti, “Design methodology to trade off power, output quality and error resiliency: application to color interpolation filtering,” in *International Conference on Computer Aided Design* (G. G. E. Gielen, ed.), pp. 199–204, IEEE, 2007.

[46] N. Banerjee, G. Karakonstantis, and K. Roy, “Process variation tolerant low power dct architecture,” in *Design, Automation and Test in Europe* (R. Lauwereins and J. Madsen, eds.), pp. 630–635, ACM, 2007.

[47] N. Banerjee, J. H. Choi, and K. Roy, “A process variation aware low power synthesis methodology for fixed-point fir filters,” in *International Symposium on Low Power Electronics and Design* (D. Marculescu, A. Raghunathan, A. Keshavarzi, and V. Narayanan, eds.), pp. 147–152, ACM, 2007.

[48] D. Mohapatra, G. Karakonstantis, and K. Roy, “Significance driven computation: a voltage-scalable, variation-aware, quality-tuning motion estimator,” in *International Symposium on Low Power Electronics and Design* (J. Henkel, A. Keshavarzi, N. Chang, and T. Ghani, eds.), pp. 195–200, ACM, 2009.

[49] S. Venkataramani, K. Roy, and A. Raghunathan, “Approximate computing,” in *2016 29th International Conference on VLSI Design (VLSID)*, pp. 3–4, IEEE, 2016.

[50] J. Han, “Introduction to approximate computing,” in *2016 IEEE 34th VLSI Test Symposium (VTS)*, pp. 1–1, IEEE, 2016.

[51] H.-J. Wunderlich, C. Braun, A. Sch, *et al.*, “Fault tolerance of approximate compute algorithms,” in *2016 IEEE 34th VLSI Test Symposium (VTS)*, pp. 1–1, IEEE, 2016.

[52] “International technology roadmap for semiconductor http://www.itrs2.net.” `http://www.itrs2.net`.

[53] D. F. Elliot and K. Rao, *Fast Transforms Algorithm, Analyses, Applications*. New York: Academic Press, 1982.

[54] J.-Y. Jou and J. A. Abraham, “Fault-tolerant matrix operations on multiple processor systems using weighted checksums,” in *28th Annual Technical Symposium*, pp. 94–101, International Society for Optics and Photonics, 1984.

[55] S.-J. Wang and N. K. Jha, “Algorithm-based fault tolerance for fft networks,” *IEEE Transactions on Computers*, vol. 43, no. 7, pp. 849–854, 1994.

[56] D. A. Anderson and G. Metze, “Design of totally self-checking check circuits for m-out-of-n codes,” *IEEE Transactions on Computers*, vol. 100, no. 3, pp. 263–269, 1973.

[57] J. Jou and J. A. Abraham, “Fault-tolerant FFT networks,” *IEEE Transactions on Computers*, vol. 37, no. 5, pp. 548–561, 1988.

[58] D. 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, 1993.

[59] C. G. Oh and H. Y. Youn, “On concurrent error detection, location, and correction of fft networks,” in *The Twenty-Third International Symposium on Fault-Tolerant Computing, 1993. FTCS-23. Digest of Papers.*, pp. 596–605, IEEE, 1993.

[60] C. G. Oh, H. Y. Youn, and V. K. Raj, “An efficient algorithm-based concurrent error detection for FFT networks,” *IEEE Transactions on Computers*, vol. 44, no. 9, pp. 1157–1162, 1995.

[61] A. L. N. Reddy and P. Banerjee, “Algorithms-based fault detection for signal processing applications,” *IEEE Transactions on Computers*, vol. 39, no. 10, pp. 1304–1308, 1990.

[62] S. Wang and N. K. Jha, “Algorithm-based fault tolerance for FFT networks,” *IEEE Transactions on Computers*, vol. 43, no. 7, pp. 849–854, 1994.

[63] G. R. Redinbo, "Concurrent error detection in fast unitary transform algorithms," in *2001 International Conference on Dependable Systems and Networks (DSN 2001) (formerly: FTCS), 1-4 July 2001, Göteborg, Sweden, Proceedings*, pp. 37–46, 2001.

[64] G. R. Redinbo and C. Nguyen, "Concurrent error detection in wavelet lifting transforms," *IEEE Transactions on Computers*, vol. 53, no. 10, pp. 1291–1302, 2004.

[65] C. Nguyen and G. R. Redinbo, "Fault tolerance design in jpeg 2000 image compression system," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 1, pp. 57–75, 2005.

[66] T. Acharya and P. Tsai, *JPEG2000 Standard for Image Compression: Concepts, Algorithms and VLSI.* Wiley, 2004.

[67] R. V. L. Hartley, "A more symmetrical fourier analysis applied to transmission problems," *Proceedings of the IEEE*, vol. 30, pp. 144–150, 1942.

[68] R. N. Bracewell, "Discrete hartley transform," *Journal of the Optical Society of America*, vol. 73, no. 12, pp. 1832–1835, 1983.

[69] W. Sweldens, "The lifting scheme: A custom-design construction of biorthogonal wavelets," in *Applied and Computational Harmonic Analysis*, vol. 3, 1996.

[70] D. S. Taubman and M. W. Marcellin, *JPEG2000 Image Compression Fundamentals, Standards and Practice.* Kluwer Academic Publishers, 2002.

[71] J.-Y. Jou and J. Abraham, "Fault-tolerant matrix arithmetic and signal processing on highly concurrent computing structures," *Proceedings of the IEEE*, vol. 74, pp. 732–741, May 1986.

[72] A. Avizienis and J. P. Kelly, "Fault tolerance by design diversity: Concepts and experiments," *Computer*, vol. 17, no. 8, pp. 67–80, 1984.

[73] S. Mitra and E. J. McCluskey, "Which concurrent error detection scheme to choose?," in *Proceedings of International Test Conference, 2000.*, pp. 985–994, IEEE, 2000.

[74] J. Von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," *Automata Studies*, vol. 34, pp. 43–98, 1956.

[75] M. Breuer, "Hardware that produces bounded rather than exact results," in *Proceedings of the 47th Design Automation Conference*, pp. 871–876, ACM, 2010.

[76] M. A. Breuer and H. Zhu, "An illustrated methodology for analysis of error tolerance," *IEEE Design & Test of Computers*, vol. 25, no. 2, pp. 168–177, 2008.

[77] R. Hegde and N. R. Shanbhag, "Soft digital signal processing," *IEEE Transactions on VLSI System*, vol. 9, no. 6, pp. 813–823, 2001.

[78] G. Varatkar and N. R. Shanbhag, "Energy-efficient motion estimation using error-tolerance," in *International Symposium on Low Power Electronics and Design* (W. Nebel, M. R. Stan, A. Raghunathan, J. Henkel, and D. Marculescu, eds.), pp. 113–118, ACM, 2006.

[79] B. Shim and N. R. Shanbhag, "Energy-efficient soft error-tolerant digital signal processing," *IEEE Transactions on VLSI System*, vol. 14, no. 4, pp. 336–348, 2006.

[80] V. K. Chippa, K. Roy, S. T. Chakradhar, and A. Raghunathan, "Managing the quality vs. efficiency trade-off using dynamic effort scaling," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 12, no. 2s, p. 90, 2013.

[81] L. Leem, H. Cho, J. Bau, Q. A. Jacobson, and S. Mitra, "Ersa: Error resilient system architecture for probabilistic applications," in *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*, pp. 1560–1565, IEEE, 2010.

[82] K. He, A. Gerstlauer, and M. Orshansky, "Controlled timing-error acceptance for low energy idct design," in *2011 Design, Automation & Test in Europe*, pp. 1–6, IEEE, 2011.

[83] K. Andra, C. Chakrabarti, and T. Acharya, "A VLSI architecture for lifting-based forward and inverse wavelet transform," in *IEEE Transactions on Signal Processing*, vol. 50, 2002.

[84] "USC-SIPI image database http://sipi.usc/edu/database." `http://sipi.usc.edu/database`.

[85] T. Burd, T. Pering, A. Stratakos, and R. Brodersen, "A dynamic voltage scaled microprocessor system," in *2000 IEEE International Solid-State Circuits Conference. Digest of Technical Papers. ISSCC*, pp. 294–295, 466, 2000.

[86] Q. Zhang, F. Yuan, R. Ye, and Q. Xu, "Approxit: An approximate computing framework for iterative methods," in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2014.

Vita

Shih-Hsin Hu is currently Director of Engineering at Qualcomm Technologies Inc. He manages low power design and methodology team and leads IP and technology development for adaptive voltage scaling (AVS), adaptive clock distribution (ACD), thermal/limit management, and on-chip sensor.

Before joining Qualcomm, he was Member of Technical Staff and SoC design lead at Freescale Semiconductor from 2004 to 2012, where he had managed multi-core network processor SoC design and microarchitecture definition. Before Freescale, He was Member of Technical Staff at Sun Microsystems from 2001 to 2004 where he had contributed to UltraSPARC processor design.

Mr. Hu received the MS degree in Electrical and Computer Engineering from Purdue University in 2000 and the BS degree in Mechanical Engineering from National Taiwan University in 1997.

Permanent e-mail address: shihhsin.hu@gmail.com

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.