Algorithms and Architectures for Secure Embedded Multimedia Systems

Amit Pande
Iowa State University
Algorithms and architectures for secure embedded multimedia systems

by

Amit Pande

A dissertation submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Engineering

Program of Study Committee:
Joseph Zambreno, Major Professor
Akhilesh Tyagi
Philip Jones
Zhao Zhang
Zhengdao Wang

Iowa State University
Ames, Iowa
2010

Copyright © Amit Pande, 2010. All rights reserved.
DEDICATION

Dedicated to my teacher, Dr. P. V. Krishnan - his life and precepts

who has taught me the meaning of education

and given me the inspiration to dedicate my life ...

... for the cause of education
TABLE OF CONTENTS

LIST OF TABLES ................................................................. vii
LIST OF FIGURES ............................................................... ix
ACKNOWLEDGEMENTS ......................................................... xiii
ABSTRACT ............................................................................. xv

CHAPTER 1. INTRODUCTION .................................................. 1
  1.1 Motivation for present research ......................................... 1
  1.2 Multimedia Compression Basics ....................................... 2
  1.3 Multimedia Encryption Basics ......................................... 3
  1.4 Comparison to Existing State-of-the-Art. ........................... 4
  1.5 Research Problem statement ........................................... 6
  1.6 Thesis Organization ...................................................... 7

CHAPTER 2. THE POLYMORPHIC DISCRETE WAVELET TRANSFORM .... 10
  2.1 Motivation and Insight .................................................. 15
    2.1.1 Daubechies 9/7-Tap Bi-Orthogonal Filter ................. 17
    2.1.2 Le Gall’s 5/3 Filter ................................................. 18
  2.2 Background and Related Work ....................................... 18
    2.2.1 Wavelet Transform Background ................................ 19
    2.2.2 Hardware Implementation of DWT ........................... 20
  2.3 Poly-DWT Filter .......................................................... 22
    2.3.1 Parameterized Filter Design .................................. 22
    2.3.2 Numerical Study ................................................... 23
2.3.3 Candidate Filters ......................................................... 25
2.3.4 Hardware Architectures ............................................... 26
2.4 Fixed Point Implementation ............................................. 30
2.5 Hardware (Re)-Allocation ............................................... 32
   2.5.1 ‘On-the-fly’ Switching .............................................. 33
   2.5.2 ‘Bit-width’ Switching .............................................. 35
2.6 Experiments ............................................................. 36
   2.6.1 Image Reconstruction Quality ................................... 36
   2.6.2 Hardware vs Software Performance ................................. 38
   2.6.3 Hardware Comparison .............................................. 39
   2.6.4 Dynamic Bit Allocation ............................................ 41
   2.6.5 Real-World Application ............................................ 42
2.7 Conclusions and Future Work .......................................... 44

CHAPTER 3. THE SECURE WAVELET TRANSFORM ......................... 47
3.1 Preliminaries ............................................................. 50
   3.1.1 Parameterized Construction of DWT ............................... 51
   3.1.2 Subband Re-orientation .......................................... 55
3.2 Security ................................................................. 60
3.3 Hardware Implementation ............................................... 61
   3.3.1 Reconfigurable Constant Multiplier (RCM) ....................... 64
   3.3.2 Implementation Results ............................................ 68
3.4 Conclusion and Future Work ............................................ 69

CHAPTER 4. CHAOTIC FILTER BANKS ........................................... 70
4.1 Introduction ............................................................. 70
   4.1.1 Chaos and Cryptography ......................................... 70
   4.1.2 Wavelets and Chaotic Filter Banks ............................... 71
   4.1.3 Scope and Organization ............................................ 72
4.2 Wavelets ............................................................... 73
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2.1</td>
<td>Commonly Used DWT Filters</td>
<td>73</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Reconfigurable Hardware Implementation</td>
<td>74</td>
</tr>
<tr>
<td>4.3</td>
<td>Chaotic Filter Bank Scheme</td>
<td>74</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Chaotic Maps</td>
<td>76</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Key Space</td>
<td>77</td>
</tr>
<tr>
<td>4.4</td>
<td>The MCFB Scheme</td>
<td>77</td>
</tr>
<tr>
<td>4.5</td>
<td>Improved Chaotic Oscillator</td>
<td>79</td>
</tr>
<tr>
<td>4.5.1</td>
<td>The Modified Logistic Map (MLM)</td>
<td>80</td>
</tr>
<tr>
<td>4.6</td>
<td>Wavelet Parameterization</td>
<td>82</td>
</tr>
<tr>
<td>4.7</td>
<td>Resistance of Chaotic Generator against Cryptanalysis</td>
<td>83</td>
</tr>
<tr>
<td>4.7.1</td>
<td>Randomness Tests</td>
<td>83</td>
</tr>
<tr>
<td>4.7.2</td>
<td>Bifurcation Map</td>
<td>85</td>
</tr>
<tr>
<td>4.7.3</td>
<td>Lyapunov Exponent</td>
<td>86</td>
</tr>
<tr>
<td>4.8</td>
<td>Security Enhancement</td>
<td>88</td>
</tr>
<tr>
<td>4.9</td>
<td>Hardware Implementation</td>
<td>89</td>
</tr>
<tr>
<td>4.9.1</td>
<td>Hardware Optimizations for ICO</td>
<td>92</td>
</tr>
<tr>
<td>4.10</td>
<td>Conclusions</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td><strong>CHAPTER 5. CHAOTIC ARITHMETIC CODING</strong></td>
<td>95</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>95</td>
</tr>
<tr>
<td>5.2</td>
<td>Piece-wise Linear Chaotic Maps</td>
<td>99</td>
</tr>
<tr>
<td>5.2.1</td>
<td>The coding procedure</td>
<td>102</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Correspondence to Arithmetic Coding</td>
<td>103</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Compression Efficiency</td>
<td>104</td>
</tr>
<tr>
<td>5.2.4</td>
<td>Application to Multimedia/ Data Encryption</td>
<td>105</td>
</tr>
<tr>
<td>5.3</td>
<td>Binary Chaotic Arithmetic Coding</td>
<td>106</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Definition</td>
<td>107</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Related works</td>
<td>107</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Implementation efficiency</td>
<td>108</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 2.1 Coefficients for the CDF 9/7 filter .......................... 17
Table 2.2 Coefficients for Le Gall 5/3 filter .......................... 18
Table 2.3 Analysis high pass filter coefficients ($H_1$) for the bi-orthogonal 9/7 tap filter . 23
Table 2.4 Analysis low pass filter ($H_0$) coefficients for the bi-orthogonal 9/7 tap filter . 24
Table 2.5 Image compression performance on SPIHT coder (PSNR values). ............ 38
Table 2.6 Hardware acceleration on a Virtex-5 XC5VLX30 FPGA (time in $\mu$s) .... 39
Table 2.7 Comparison of binary filter features and hardware resources requirements . 45
Table 2.8 Performance evaluation on 45nm standard cell libraries ....................... 46

Table 3.1 PSNR values (in db) for image reconstruction with various random keys (encoded with key0) ..................................................... 60
Table 3.2 Variations in image reconstruction quality (PSNR values) with hamming distance ........................................................ 61
Table 3.3 Hardware Utilization of DWT architecture on Xilinx Virtex XCVLX330 FPGA 63

Table 4.1 Statistical performance of Generated Sequence $b_n$ (results based on 1000 sequences of length 10000 each). ................................. 87

Table 5.1 Beginning and end Intervals for given example .............................. 101
Table 5.2 Decoding the original sequence for initial value of 0.2 ................. 102
Table 5.3 Encoding the original sequence ‘ABAC’ .............................. 103
Table 5.4 Decoding the codeword 0.2 using Arithmetic coder ........................ 103
Table 5.5 Parameter List for the eight possible choices of chaotic encoder ......... 109
Table 5.6 Compression Performance of BAC and BCAC for various length strings. The average length and standard deviation of codeword is presented for various $p$ values and various length of input string.
| Figure 1.1 | (a) A typical multimedia compression scheme, (b) Naive/ full multimedia encryption schemes, and (c) Partial or Selective Multimedia encryption schemes. |
| Figure 1.2 | The proposed scheme for efficient multimedia compression and encryption: (White) Traditional Video Compression Engine and (Red) Video Compression system augmented with different operations to ensure real-time encryption |
| Figure 2.1 | Conceptual overview of the Polymorphic Wavelet Architecture |
| Figure 2.2 | Conceptual overview of the DWT filter design constraints and desired features |
| Figure 2.3 | Basic stages of a one level 2-D wavelet transform operation |
| Figure 2.4 | Result of three level 2-D wavelet transform operation on an image |
| Figure 2.5 | Numerical analysis of quantization error for seven bit finite representation of filter coefficients |
| Figure 2.6 | Hardware architectures for bi-orthogonal 9/7 filter |
| Figure 2.7 | Architectural details of poly-DWT to facilitate ‘Reconfiguration’ |
| Figure 2.8 | Register level details to enable reconfiguration (a) Type A architecture and (b) Type B architecture |
| Figure 2.9 | (a) Results of one level of DWT and (b) Energy decomposition by respective filters |
| Figure 2.10 | Change in FPGA clock frequency(MHz) for variable word widths for various filters |
| Figure 2.11 | Plot of PSNR vs the number of bits allotted for internal registers |
| Figure 2.12 | Comparison of register usage for the binary filter implementations |
Figure 3.1 PSNR values (in db) for image reconstruction using SPIHT coder at different
bitrates (in bpp or bits per pixel) .................................................. 51

Figure 3.2 Image reconstruction with different keys. (a) show the original images which
are then encrypted with $\alpha = 2$, (b)-(e) show reconstruction with $\alpha = 1, 2, 3$
and 3.5 respectively ............................................................... 53

Figure 3.3 (a) Image decomposition with DWT (6 levels) leading to 19 subbands. 3
bits are assigned for each subband’s re-orientation information. (b) Possible
transpose relationships for sub bands. $A$ is the original matrix. The eight per-
mutations are achieved using transpose relationship (’), and reverse-ordering
of the subbands (− for reverse, + for forward read access) along both rows
and columns ................................................................. 55

Figure 3.4 MSE values for sample images with the change in no. of bits assigned to one
$\alpha$ parameter. The image was encoded with one $\alpha$ value and decoded with
adjacent alpha values for various bit-widths of $\alpha$. 1000 simulations were run
to obtain an average value. ................................................... 56

Figure 3.5 Image reconstruction with different keys. A- Aerial map image, B- San Fran-
cisco Golden gate aerial image, C- Brick wall (texture) image and D- Airplane
image. (i)- Original image encrypted with key0, (ii)- Image decrypted with
same key, (iii)-(vi)- Image decrypted with randomly generated keys......... 57

Figure 3.6 Image reconstruction with different keys. A- Aerial map image, B- San Fran-
cisco Golden gate aerial image, C- Brick wall (texture) image and D- Airplane
image. (i)- Original image encrypted with key0, (ii)- Image decrypted with
same key, (iii)-(vi) Image decrypted with hamming distance of 1,4, 6 and 8 .. 58

Figure 3.7 Image reconstruction with randomly generated keys. (a)-(d) give result of
1000 random trials on the four sample images respectively. The x-axis gives
results with different keys. The $500^{th}$ trial ( with $500^{th}$ key) refers to the
test case with decryption with same key as the encryption key. The y-axis
represents the PSNR value of the reconstructed images. ...................... 59
Figure 3.8  Hardware Architecture for the 1-D SWT Filter  

Figure 3.9  Building a (K+1)-LUT from K-LUT  

Figure 3.10  Illustration of 12-bit constant multiplication with a 8-bit input. (a) The individual bits of product are obtained as output of a 8-LUT. (b) 4-LUTs are used in the implementation with the input A divided into 2 4-bit values.  

Figure 4.1  Block Diagram representation of the Chaotic Filter Bank Scheme. (a) The encryption module and (b) The decryption module  

Figure 4.2  Histogram for 50000 samples obtained using Logistic map with initial seed 0.100010 and (a) $\lambda_{LM} = 3.61$ and (b) $\lambda_{LM} = 3.91$ (c) $\lambda_{LM} = 4$ and (d) $\lambda_{LM} = 3.83$  

Figure 4.3  Block Diagram representation of the MCFB Scheme. (a) The encryption module and (b) The decryption module  

Figure 4.4  Histogram for 50000 samples obtained using Modified Logistic map with $\alpha$ values corresponding to (a)$\lambda_{LM} = 3.61$ and (b) $\lambda_{LM} = 3.91$  

Figure 4.5  Correlation test of the pseudo-random sequence. (a) Generated using different initial values $x_0$ and (b) different initial parameter $\alpha$. The plots are measured against initial value $\alpha = 0.110000$ and $x_0 = 0.410021$  

Figure 4.6  Bifurcation Diagram for (a) Logistic Map showing the white spaces (islands of stability) and asymmetricity and (b) Modified Logistic Map with symmetric and flatter distribution  

Figure 4.7  Plot of Lyapunov Coefficient ($\Lambda$ - solid line) for (a) Logistic map as a function of parameter $\lambda_{LM}$ indicating regions of non-chaotic behavior and (b) Modified Logistic map showing higher divergence than Logistic Map and independence of $\Lambda$ from parameter $\alpha$  

Figure 4.8  Hardware architecture for the Modified Chaotic Filter Bank Scheme  

Figure 4.9  Hardware architecture for Improved Chaotic Oscillator
Figure 5.1 A sample piece-wise linear map for arithmetic coding like compression (a) The entire map is shown ($\rho$), (b) A single linear part of the map ($\rho_k$) is zoomed. It can have a positive or negative slope depending on choice.

Figure 5.2 The piece-wise chaotic map for N=4. Probability distributions for symbols A, B, C and D are given by $p(A) = 0.4$, $p(B) = 0.3$, $p(C) = 0.2$ and $p(D) = 0.1$. The mapping of maps and symbols is given by: $\rho_1(x) \equiv A$, $\rho_2(x) \equiv B$, $\rho_3(x) \equiv C$, and $\rho_4(x) \equiv D$.

Figure 5.3 An arbitrarily chosen piece-wise linear map.

Figure 5.4 (a-h) show the eight modes of the skewed binary map ($p=0.6$).

Figure A.1 (a) Resulting subbands after three levels of wavelet decomposition, and (b) Three levels of wavelet decomposition of a sample image.

Figure B.1 An overview of partial encryption scheme.
ACKNOWLEDGEMENTS

I would like to thank my major professor and advisor Dr. Joseph Zambreno for his constant support and guidance throughout the course of this study. I really appreciate his gentleness and considerate nature, and at the same time being highly professional at work. He always gave me the necessary time whenever I needed it, and has been always supportive to give the necessary freedom for pursuing my ideas, along with actively giving his own constructive inputs and suggestions. My experience with him here during this study has been extremely helpful and enriching for me. He really considers deeply about the welfare of his students.

I would like to thank my committee members Dr. Akhilesh Tyagi, Dr. Phillip Jones, Dr. Zhengdao Wang and Dr. Zhao Zhang for their support, suggestions, availability and courtesy to me.

I would like to deeply thank Dr. P.V. Krishnan, whose personal example and teachings has been pivotal for me in all aspects of my life. He has stood beside me in all the difficult times, and his love has helped me overcome all the failures and successes. He has always been with me for setting the right priorities in my work (and my life), working with a clear goal and focus, and most importantly learned the role of character in directing the proper advancement of science and technology. I wish to use this degree for its desired motive and live a life dedicated to ultimate welfare of society.

I would like to thank my undergraduate adviser Dr. Ankush Mittal, whose example was instrumental in inspiring me to pursue higher studies and present occupation.

I want to thank my parents for always supporting and encouraging me, right from childhood, in inquisitive thinking and working towards an engineering degree (at IIT Roorkee) which transformed my goals and purpose of life. I take this opportunity to thank my friends and room-mates who have made me feel at home - being always cordial, supportive, helpful, encouraging, and serving me to focus on the execution of my occupational and other responsibilities to the best. Dr. Rangan and Dr.
Siddhartha have been like my elder brothers during my stay at Ames, always helping me in difficult times. My roommates Dr. Ankit, Sparsh and Abhishek have always extended themselves to complement me with my responsibilities. I would like to thank Sidharath, Dr. Siva, Venkat, Dr. Tanay, Chetan, Vikram and Sandeep for being so nice friends and well-wishers.
ABSTRACT

Embedded multimedia systems provide real-time video support for applications in entertainment (mobile phones, Internet video websites), defense (video-surveillance and tracking) and public-domain (tele-medicine, remote and distant learning, traffic monitoring and management). With the widespread deployment of such real-time embedded systems, there has been an increasing concern over the security and authentication of concerned multimedia data.

While several (software) algorithms and hardware architectures have been proposed in the research literature to support multimedia security, these fail to address embedded applications whose performance specifications have tighter constraints on computational power and available hardware resources. The goals of this dissertation research are two fold:

1. To develop novel algorithms for joint video compression and encryption. The proposed algorithms reduce the computational requirements of multimedia encryption algorithms. We propose an approach that uses the compression parameters instead of compressed bitstream for video encryption.

2. Hardware acceleration of proposed algorithms over reconfigurable computing platforms such as FPGAs and over VLSI circuits. We use signal processing knowledge to make the algorithms suitable for hardware optimizations and reduce the critical path of circuits using hardware-specific optimizations.

The proposed algorithms ensure a considerable level of security for low-power embedded systems such as portable video players and surveillance cameras. These schemes have zero or little compression losses and preserve the desired properties of compressed data in encrypted bitstream to ensure secure and scalable transmission of videos over heterogeneous networks.
The proposed algorithms also support indexing, search and retrieval in secure multimedia digital libraries. This property is crucial not only for police and armed forces to retrieve information about a suspect from a large video database of surveillance feeds, but extremely helpful for data centers (such as those used by youtube, aol and metacafe) in reducing the computation cost in search and retrieval of desired videos.
CHAPTER 1. INTRODUCTION

1.1 Motivation for present research

With the continuing development of both computing and Internet technology, multimedia data is being used more and more widely, in applications such as video-on-demand, video conferencing, broadcasting, etc. Currently, multimedia data is closely related to many aspects of daily life, including education, commerce, and politics. In order to maintain privacy or security, sensitive data needs to be protected before transmission or distribution.

Access right control based methods are useful in controlling illegal access by the authentication of users. For example, in video-on-demand, a user name and password are used to control the browsing or downloading operations. However, in this method, the multimedia data itself is not protected, and may be stolen during the transmission process. Thus, to maintain security, multimedia data should be protected before transmission or distribution. The typical protection method is the encryption technique which transforms the data from the original form into an unintelligible form.

Computer security is an active research field, the fruits of which include the protocols (e.g. SSL [111], TLS [18]) and cryptographic ciphers (e.g. AES [31], DES [32], IDEA [50]) that drive much of the world’s electronic communications, commerce, and storage.

Communication security of multimedia data can be accomplished by use of such cryptographic ciphers over the compressed multimedia stream. In many cases, when the multimedia is textual or static data, and not a real-time streaming media, we can treat it as an ordinary binary data and use the conventional encryption techniques. Encrypting the entire multimedia stream using standard encryption methods is often referred to as the naive approach. The naive approach is usually suitable for text, and sometimes for small bitrate audio, image, and video files that are being sent over a fast dedicated channel. Secure Real-time Transport Protocol, or shortly SRTP [10], is an application of the naive
approach. In SRTP, multimedia data is packetized and each packet is individually encrypted using AES. The naive approach enables the same level of security as that of the conventionally used cryptographic cipher.

However, it is difficult to use these naive schemes directly for real-time multimedia. This is due to the fact that multimedia data are often of high redundancy, of large volumes and require real-time interactions, such as displaying, cutting, copying, bit rate conversion, etc. Besides the security issue, encrypting images or multimedia with these cryptographic ciphers is time consuming and not suitable for embedded systems which typically have constraint on device power and hardware resources. It also leads to compromises in multimedia properties such as scalability, and transcoding as the nature of a cipher text output from an encryption engine is much different from a compressed bitstream.

1.2 Multimedia Compression Basics

Video or multimedia compression refers to reducing the quantity of data used to represent digital video data. It allows transmission of multimedia over bandwidth constrained communication channels. For example- a standard video monitor displays a frame usually with the resolution of $800 \times 600$ pixels. For a color image, a pixel is represented by 3 bytes of data (one for Red, Blue and Green respectively). Thus, a one hour video at 30 frames per second will require 144 GB of space in hard disk and is impossible to transmit it over any practical communication channel. It is compressed to around 500-600 MB by the use of MPEG-2 [9] compression format. A typical compression scheme is composed of several sequential steps including transform coding, quantization, motion compensation (or temporal compression), and variable length coding. As shown in Figure 1.1(a), a sequential combination of these blocks make up an entire multimedia compression system. For example - JPEG (Joint Photographic Experts Group) still image compression standard has three main steps which are executed one after the other, in succession.

1. Discrete Cosine Transform (frequency transformation step).

2. Quantization.

3. Huffman Coding (entropy coding step).
Multimedia compression involves large computations and large amount of data-transfers thus requiring application-specific hardware such as ASICs and FPGAs to compress and deliver the media at run-time. A good summary of recent advances in multimedia compression is given in [104]. Video compression over FPGA and VLSI devices has gained increased attention because of popularity of low power embedded devices over the past two decades [92, 28, 7] Recently, the authors in [21] propose a multi-mode embedded video codec with DRAM area and external access power savings to support a real-time encoding of CIF images (having resolution of 352x288 pixels). They propose a power-aware design for video coding in embedded scenarios [21].

Thus, an efficient architectural design of multimedia compression blocks is a must to ensure real-time video delivery.

1.3 Multimedia Encryption Basics

Encryption is the process of transforming information (referred to as plaintext) using a cryptographic algorithm to make it unreadable to anyone except those possessing special knowledge (known
as a key). Multimedia encryption technology provides end-to-end security when distributing digital content over a variety of distribution systems.

AES (American Encryption Standard [31]) is the commercially used cryptographic cipher for data encryption operations. Naive multimedia encryption algorithms such as SRTP [10] encrypt the entire compressed output of a multimedia encoder as shown in Figure 1.1(b). This incurs a large computational overhead for the encoder (and consequently for the decoder also). Partial or selective encryption schemes were built to reduce the computational overhead by encrypting only an important chunk of multimedia data. This is shown in Figure 1.1(c).

On the one side, compression and encryption operations require large amount of computations and latency, while on the other side, there has been an increasing trend towards deployment of battery-driven low power embedded systems such as portable mobile devices (i-pods, mobile phones and cameras).

Apart from optimizations in hardware architectures, we also need to reduce the computation cost for secure multimedia transactions.

While the compressed multimedia files typically exhibit well defined hierarchical structure that can be exploited in several useful ways, e.g., for scalability, random access, transcoding, rate shaping; these structures are not recognizable in the cipher text, and hence, are wasted. These properties are useful to index, search and retrieve compressed multimedia from digital libraries and also for communication over heterogeneous networks. We need a paradigm where encryption doesn’t change the compressed output, yet provide access and copy control for concerned media. The partial/selective encryptions, as mentioned above, reduce the computational overhead, but generally lead to compression inefficiency, and may change the syntax of video bitstream.

Thus, we need encryption of video data without affecting the properties of compressed bitstream, or affecting the compression performance.

1.4 Comparison to Existing State-of-the-Art.

Computer security protocols (e.g. SSL, TLS) and associated cryptographic ciphers (e.g. AES, DES, RSA) drive much of the world’s electronic communications, commerce, and storage. These tech-
Techniques have been used for conventional multimedia encryption and authentication. For example, the Secure Real-time Transport Protocol, or shortly SRTP, is an application of this approach, where multimedia data is packetized and each packet is individually encrypted using AES. The HDTV encryption standard also uses a similar approach, allowing one to choose from AES or the lightweight M6 cipher. However, M6 cipher has been found prone to slide attacks and plaintext attacks. Numerous other proposals are also found in research literature where the encryption operation is performed either at some intermediate level during compression or after the final compression.

Selective encryption schemes have been proposed [61, 34] to reduce the computational requirements of these full encryption schemes. Lian et al. [56] present a scheme for encryption and watermarking of DCT coefficients. Their scheme uses Exp-Goloumb codes for the encryption operation. DWT-based partial encryption scheme have been proposed which encrypt only a part of compressed data; only $13 - 27\%$ of the output from quadtree compression algorithms is encrypted for typical images.

The naive and selective encryption schemes mostly compromise the desired features of multimedia data such as scalability, random access, rate shaping, and DC image extraction. Many usage scenarios of multimedia streams (e.g. rate adaptation for multimedia transmission in heterogeneous networks and DC-image extraction for multimedia content searching) cannot be applied directly in the bitstream encrypted by generic encryption tools or their simple variations. The processing would therefore require the delegates to hold the decryption keys to decrypt the content, process the data, and then re-encrypt the content. Besides the high computation cost of this operation, revealing the decryption keys to potentially untrustworthy delegates is often not in line with the security requirements of many applications. Unlike the ‘all-or-none’ protection for generic data, the value of multimedia and in turn its secure transmission are closely tied with the perceptual quality and the timeliness of the content.

Such limitations can be alleviated through the development of parameterized compression blocks that can achieve simultaneous encryption while preserving multimedia-specific properties. Thus, the compression operation itself uses a key to encode the input data and no external cryptographic engine is required. Recently, some schemes have been developed using this approach, but the degree of security offered is low and these modifications often lead to an added hardware implementation complexity. For
example, Grangetto et al. [36] introduce a parameterization in the arithmetic coding stage of multimedia compression. This parameterization is used to build a key scheme. However, the performance of such scheme for resource-constrained environments remain untested. Kim et al. [45] presents a variation of [36] that improves the security performance of parameterized arithmetic coding scheme but also increases the complexity of hardware implementation. Both of these schemes are found to be weak against plaintext-based attacks.

The authors in [65] present a joint signal processing and cryptographic approach to multimedia encryption. They use index mapping and constrained shuffling to achieve confidentiality protection. This ensures that the encrypted bitstream still complies with the state-of-the-art multimedia coding techniques. While the initial results are promising, the extra computational resources required are quite high. Some multimedia encryption schemes based on transform coefficients confusion alone have been proposed but they are bound to be separable and weak against any cryptanalysis. The Fast Encryption Algorithm for Multimedia (FEA-M) has been proposed for real-time multimedia encryption [121]. While FEA-M and similar algorithms can be implemented very efficiently in hardware, the security of such schemes has been tested and found wanting [75, 74, 122, 54, 53].

1.5 Research Problem statement

We find that ensuring multimedia transactions over real-time embedded systems has many issues:

1. Video compression and data encryption are both computationally expensive tasks and successive encryption after compression restricts the low-power and low-latency design of custom hardware.

2. Successive encryption after compression leads to compromises in multimedia properties such as scalability, and transcoding.

3. Hardware implementations of multimedia encryption schemes are not well optimized to provide optimal performance in constrained scenario of real-time embedded systems.

Thus, the aim of this dissertation research is to develop algorithms and architectures for secure embedded multimedia systems which promise secure multimedia content delivery and provide real-time compression and low power consumption.
In effect, we propose cryptographic schemes that encrypt the compression parameters and not the compressed output themselves. It allows us to restore the essential properties of multimedia data such as scalability, random access etc and simultaneously build a key-space for multimedia encryption.

Figure 1.2 gives the big picture overview of our thesis. By augmenting the building blocks of multimedia compression engine with simple transformations, we get an efficient key space for multimedia encryption.

There are three major components in a video encoder: temporal model, spatial model and entropy coder. In this thesis, we redesign the algorithms and architectural mappings for the Discrete Wavelet Transform (DWT) and Arithmetic Coding (AC) stages in an entropy encoder to augment security feature to it.

1.6 Thesis Organization

This thesis is organized as follows:

- Chapter 1: Introduction

  It gives a general introduction to the thesis, presenting the motivation for present research, nature of the problem, relevant research, research problem statement and thesis organization.

- Chapter 2: Polymorphic Wavelet Architecture
The second chapter presents a polymorphic hardware implementation of Discrete Wavelet Transform (DWT). The DWT is an important step for many video compression algorithms. The output of DWT operation is subband decomposition of input image/frame into several sub-images, called as sub-bands. It is traditionally calculated as convolution of input signal with two asymmetric bi-orthogonal filters. In the first part, we build a Polymorphic Wavelet Transform (which we call as Poly-DWT), which uses the signal processing expertise to cater the requirements of efficient hardware implementation. Further, it enables using DWT at different bit-width resolutions, and allows a choice of filters. The hardware architecture can be mapped to an FPGA or an ASIC, and thus dynamic trade-off can be reached between power and image quality.

- Chapter 3: Secure Wavelet Architecture

In the third chapter, we integrate security aspects to DWT. We use a parametric DWT combined with efficient subband scrambling scheme to build a DWT based encryption scheme, which we call as Secure Wavelet Transform (SWT). The parameterization combined with zero overhead subband re-orientation (or scrambling) allows us to get lightweight encryption of video data. SWT was implemented both over FPGA (Virtex-5 FPGA) and ASIC (45 nm cell library). We also present a hardware optimization using reconfigurable constant multipliers.

- Chapter 4: Chaotic Filter banks Scheme

After looking a while over DWT, we turn our focus to some other algorithms and architectures which can be used to provide real-time security. We first look at using stream ciphers for securing the multimedia bit-streams. They are commonly used for data encryption and have little computational overhead or latency, making them suitable for use with large volumes of multimedia data. We study chaotic systems which exhibit random-like behavior, are extremely simple to implement and require little hardware resources for implementations and build a stream cipher based on chaotic maps. This stream cipher was then integrated with SWT to obtain a chaotic filter bank scheme. This scheme was implemented and optimized over FPGA. This discussion forms the fourth chapter of this thesis.

- Chapter 5: Chaotic Arithmetic Coding
While studying chaos and chaotic maps, we came across an interesting observation: *Iteration over chosen chaotic maps is equivalent to arithmetic coding (AC)*. AC is one of the most efficient lossless data compression method and used in most video encoders as the last step in encoding. This observation is developed in the last part, which describes a powerful scheme for multimedia encryption using chaotic maps, called as Chaotic Arithmetic Coding. This scheme has little computational overhead over compression schemes and yields excellent compression and encryption properties.

- **Chapter 6: Conclusions**

We finally summarize our efforts in developing algorithms for efficient encryption and hardware-implementation based on multimedia compression approaches and give some direction for future work.

Considering the wide span of topics in this dissertation research - from video processing to FPGA, embedded systems, and security, we have appended a brief introductory account of terms such as video compression (DWT, AC) in Appendix A and cryptography (chaos, and video encryption) in Appendix B for the interested readers.
CHAPTER 2. THE POLYMORPHIC DISCRETE WAVELET TRANSFORM

Many modern computing applications have been enabled through the use of real-time multimedia processing. While several hardware architectures have been proposed in the research literature to support such primitives, these fail to address applications whose performance and resource requirements have a dynamic aspect. Embedded multimedia systems typically need a power and computation efficient design in addition to good compression performance. In this paper, we introduce a Polymorphic Wavelet Architecture (Poly-DWT) as a crucial building block towards the development of embedded systems to address such challenges. We illustrate how our Poly-DWT architecture can potentially make dynamic resource allocation decisions, such as the internal bit representation and the processing kernel, according to the application requirements. We introduce a filter switching architecture that allows for dynamic switching between 5/3 and 9/7 wavelet filters and leads to a more power efficient design. Further, a multiplier-free design with a low adder requirement demonstrates the potential of Poly-DWT for embedded systems. Through an FPGA prototype, we perform a quantitative analysis of our Poly-DWT architecture, and compare our filter to existing approaches to illustrate the area and performance benefits inherent in our approach. Therefore, Poly-DWT is an attempt to design an architecture that provides the desired objectives namely - good compression, large system throughput, low hardware cost and intelligent allocation of hardware resources for such applications.

The contributions of the Poly-DWT can be summarized as follows:

- The Poly-DWT architecture enables dynamic reconfiguration of hardware resources to efficiently create a dynamic response to changing external conditions.

- A family of parameterized bi-orthogonal 9/7 filters was used to derive binary coefficient filters for a hardware-efficient implementation.
• A multiplier-free binary 9/7 wavelet filter is introduced to obtain a faster and more efficient implementation.

• A ‘on-the-fly’ switching scheme to allow runtime flipping between 5/3 and 9/7 wavelet structures with hardware reuse is used in Poly-DWT.

• A ‘bit-width’ switching scheme is presented to dynamically adapt the internal hardware resources according to the dynamic requirements of application.

Multimedia services over embedded devices are becoming popular with the development of scalable encoders and rise of reconfigurable hardware to support the required high throughput. The large computational complexity and memory requirements involved in real-time image processing algorithms have been a bottleneck for such systems.

The Discrete Wavelet Transform (DWT) has emerged as a powerful tool for compression and is being used in many multimedia and signal processing applications. It constitutes a significant part of the overall computations involved in image/video compression schemes. Many image compression schemes have been derived from DWT-based structures [91, 95, 101]. The work on using Embedded Zero-tree Wavelet (EZW) structures [95] for image compression was a milestone research that introduced sub band coding to achieve high compression performance. [91] introduced a more efficient DWT-based Set Partitioning in Hierarchical Trees (SPIHT) encoding to improve the performance of Shapiro’s EZW coding. [101] proposed the DWT-based Embedded Block Coding with Optimal Truncation (EBCOT) coding algorithm which was accepted for scalable encoding in JPEG2000 [24]. JPEG2000 achieves almost twice as much compression as JPEG with the same reconstruction quality of images (in terms of PSNR or Peak Signal-to-Noise Ratio). DWT has been incorporated in recent image and video compression research such as motion JPEG2000 [85]; 3-D, 4-D sub band coding [23, 119]; and MPEG-4 SVC (Scalable Video Coding extension, released in July 2007) [93]. Of the fourteen proposals for SVC received by the MPEG committee, twelve were based on DWT, while two were extensions of the existing DCT based MPEG-4 AVC standard. Thus, DWT is increasingly becoming a popular choice for image/video compression due to high compression, scalability and other features.

We recognize that DWT serves as backbone for new generation multimedia compression schemes.
and present a polymorphic architecture for its hardware implementation in this paper. Implementations such as those using ASICs or FPGAs are capable of accelerating these computations by exploiting the inherent algorithmic parallelism. [99] discuss the performance requirements of a reconfigurable hardware architecture for a scalable wavelet-based video decoder. In [29], the authors present a complete video delivery chain including a video server, negotiation and scalable video clients using a wavelet based coding scheme at its core. Many hardware implementations have also been proposed in the research literature [4, 11, 12, 88, 49, 69, 109]. These implementations aim at reducing hardware complexity in order to improve the system throughput.

Another concern is the fact that many typical applications of DWT have dynamic resource requirements. For example, consider a distributed video surveillance setup [105, 106]. There are low-activity (idle) times and high-activity (active) times associated with the camera. During idle times, a low-power and low-bandwidth design may satisfy the requirements. However, during active times, the system would require transmission of a higher quality signal over a potentially-sparse resource network. In such cases, it would be extremely beneficial to be able to distribute the available hardware resources to allow a compression efficient implementation using a relatively large amount of power.

In this paper, we introduce a new layout and reconfiguration scheme for multimedia applications, which we call the Polymorphic Wavelet Architecture (Poly-DWT). We define polymorphism as the capacity of an architecture to adapt its hardware usage to meet the desired dynamic specifications. In the
image processing domain, these specifications would be in terms of throughput, reconstruction quality, and power consumption, among others. Our Poly-DWT architecture allows the individual processing kernels to modify their hardware resources to suit the instantaneous application requirements. At its highest level, the Poly-DWT provisions for optimal device usage under the given performance and quality requirements. A fine-grained description of poly-DWT has been provided which allows runtime reconfigurability of the design over commodity FPGA platforms and ASIC designs.

Figure 2.1 gives a general description of Poly-DWT and its interface with a larger multimedia system. Some multimedia input (such as a stream of pixels for consecutive frames of a video) is first transformed into the time-frequency domain by the wavelet transform (DWT). Depending upon the throughput required and the input available from the video device, various instances of DWT kernels can be used in the implementation. The DWT kernel can be implemented using varying lengths, leading to varying image compression properties of the DWT block. An interface is provided for the application to dynamically notify the architecture about its performance requirements in terms of the hardware requirements and the image reconstruction quality requirements. Besides the previously-mentioned video surveillance application, other real-time video streaming applications such as those used in medical image processing [52], remote laboratories [76], or educational video streaming [78] may benefit enormously from the polymorphism of DWT kernels.

We summarize the requirements of embedded multimedia system design in Figure 2.2 and they are enumerated below:
Modern embedded multimedia systems would require transmission of a high quality signal over a potentially-sparse resource network. Thus, good compression is a desired feature of an efficient implementation.

High system throughput and good perceptual quality are desired features and pose constraints on system design.

Embedded systems have hardware and power constraints because they are typically mobile, battery-driven devices.

Hardware reconfiguration of the filters is the enabling technology to realize these trade-offs. Intelligent allocation of hardware resources can achieve a run-time trade-off between hardware resources and performance constraints.

Our Poly-DWT architecture takes these explicit run-time requirements, along with an output feedback of the available hardware resources and image reconstruction quality and continually makes a reconfiguration decision. The reconfiguration mentioned in this paper refers to the ability of our hardware to reconfigure its hardware resources. The implementation of our design can be done over FPGA, and ASICs. Given an image quality constraint the architecture can self-reconfigure to maximize device performance or power consumption, and given an external resource or performance constraint it can reconfigure to maximize image quality. Initial results had been presented in [79, 80]. The proposed approach can provide a set of solutions for the dynamic requirements of system performance and power consumption without any overheads in throughput or hardware cost in comparison with existing approaches.

The contributions of this paper can be summarized as follows:

- We introduce the concept of the Polymorphic Discrete Wavelet Transform (Poly-DWT) architecture. The Poly-DWT architecture enables dynamic allocation of hardware resources to efficiently create a dynamic response to changing external conditions.

- We discuss the development of a family of parameterized bi-orthogonal 9/7 filters and the derivation of binary coefficient filters for hardware efficient implementation.
A multiplier-free binary 9/7 wavelet filter is introduced to obtain a faster and more efficient implementation.

A switching scheme to allow runtime switching between 5/3 and 9/7 wavelet structures with hardware reuse is presented.

We present a quantitative analysis of the various factors and trade-offs involved in a Poly-DWT implementation.

We present a detailed area/performance trade-off analysis for the sample Poly-DWT filters.

The remainder of this paper is organized as follows. In section 2.2, we give a working knowledge of DWT filters used in image compression. Section 2.3 provides a background study of the DWT algorithm and its hardware implementation. We also present the related research and limitations with existing hardware implementations. This motivates us for a Polymorphic design which is presented in Section 2.4. The subsections give the mathematics, numerical study and background of the design. Next, we arrive at the candidate filters and their hardware architectures and then choose the optimal filters for Poly-DWT. Section 2.5 gives an insight into hardware re-allocation by changing the internal word width representation of registers by ‘bit-width’ switching scheme. Section 2.6 introduces the ‘on-the-fly’ switching scheme for filter coefficients. Section 2.7 also gives details of other experiments both in ModelSim and Xilinx ISE for hardware performance and over MATLAB for rigorous image processing performance measurements. In Section 2.8 we conclude the paper with a look towards planned future work.

### 2.1 Motivation and Insight

Prior works in signal processing explain that the 1-D DWT can be viewed as a signal decomposition using specific low pass and high pass filters. A single stage of image decomposition can be implemented by successive horizontal row and vertical column wavelet transforms. Thus one level of DWT operation is represented by filtering with high and low pass filters across row and column successively and is illustrated in Figure 2.3. After each filtering a down sampling is done by a factor of 2 to remove the redundant information. The two most common DWT filters used in image compression are Le
Gall’s 5/3 filter and the Daubechies 9/7 filter [24]. They are accepted in the JPEG2000 standards. The Le Gall’s filter has rational coefficients and its hardware implementation requires less resources. The Daubechies 9/7 (also commonly known as CDF 9/7) filter has better compression performance. However, it has irrational coefficients therefore its hardware requirements are very large.

This paper develops the Poly-DWT architecture to serve as a backbone for real-time multimedia applications to address their dynamic demands and constraints. In this paper we discuss some dimensions that provide this polymorphism to our architecture. The first dimension is the choice of suitable DWT filter. Different applications such as medical image processing, High Definition Television, video surveillance applications, and wireless video all have different real-time constraints [78, 76] and different filters may serve the requirements at different times.

The complexity of DWT hardware is another important design aspect. An implementation with diverse hardware requirements like multipliers, buffers etc will have a lower throughput due to increased processing time and is less favorable for Polymorphic architecture.

In this paper a binary coefficients 9/7 filter is implemented to allow cheaper implementation cost, higher throughput and ‘on-the-fly’ switching to 5/3 filter architecture. The term ‘binary coefficients filter’ refers to a filter whose coefficients can be exactly written in the form \( \frac{p}{2^q} \) where \( p \) and \( q \) are integers. Thus, we have the desired rational properties of Le Gall’s 5/3 filter and image compression performance similar to Daubechies’ 9/7 filter.
Table 2.1 Coefficients for the CDF 9/7 filter

<table>
<thead>
<tr>
<th>$i$</th>
<th>$h_0(i)$</th>
<th>$h_1(i)$</th>
<th>$g_0(i)$</th>
<th>$g_1(i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>±4</td>
<td>0.026748757411</td>
<td>0</td>
<td>0</td>
<td>0.026748757411</td>
</tr>
<tr>
<td>±3</td>
<td>-0.016864118443</td>
<td>0.091271763114</td>
<td>-0.091271763114</td>
<td>0.016864118443</td>
</tr>
<tr>
<td>±2</td>
<td>-0.078223266529</td>
<td>-0.057543526229</td>
<td>-0.057543526229</td>
<td>-0.078223266529</td>
</tr>
<tr>
<td>±1</td>
<td>0.266864118443</td>
<td>-0.591271763114</td>
<td>0.591271763114</td>
<td>-0.266864118443</td>
</tr>
<tr>
<td>0</td>
<td>0.602949018236</td>
<td>1.11508705</td>
<td>1.11508705</td>
<td>0.602949018236</td>
</tr>
</tbody>
</table>

2.1.1 Daubechies 9/7-Tap Bi-Orthogonal Filter

The bi-orthogonal Daubechies 9/7 filter is the most widely used filter for DWT operation. These wavelets have symmetric scaling and wavelet functions, i.e., both the low pass and high pass filters are symmetric. This filter has excellent image compression capabilities. There are four filters that comprise the two-channel bi-orthogonal wavelet system. The analysis and synthesis low-pass filters are denoted by $H_0$ and $G_0$ respectively. The analysis and synthesis high pass filters are denoted by $H_1$ and $G_1$ respectively and are obtained by quadrature mirroring the low-pass filters.

$$H_1(z) = z^{-1}G_0(-z), G_1(z) = zH_0(-z)$$

(2.1)

If we define $D(z) = G_0(z)H_0(z)$ the Perfect Reconstruction (PR) condition simplifies to the following:

$$D(z) + D(-z) = 2$$

(2.2)

This equation is solved using Lagrange Half Band Filters (LHBF), $L_k(z)$ where:

$$D(z) = L_k(z) = z^k \left(\frac{1 + z^{-1}}{2}\right)^{2k} \alpha(k),$$

$$\alpha(k) = \sum_{n=0}^{k-1} \binom{k+n-1}{n} \left(\frac{2 - (z + z^{-1})}{4}\right)^n$$

(2.3)

This is simplified for $k = 4$ to get the famous Cohen-Daubechies-Feauveau (CDF) or simply Daubechies bi-orthogonal 9/7 filter. The filter coefficients are irrational and their approximate values are given in
Table 2.2  Coefficients for Le Gall 5/3 filter

<table>
<thead>
<tr>
<th>$i$</th>
<th>$h_0(i)$</th>
<th>$h_1(i)$</th>
<th>$g_0(i)$</th>
<th>$g_1(i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>±2</td>
<td>-1/8</td>
<td>0</td>
<td>0</td>
<td>-1/8</td>
</tr>
<tr>
<td>±1</td>
<td>2/8</td>
<td>-1/2</td>
<td>1/2</td>
<td>-2/8</td>
</tr>
<tr>
<td>0</td>
<td>6/8</td>
<td>1</td>
<td>1</td>
<td>6/8</td>
</tr>
</tbody>
</table>

Table 2.1. Ansari [5] discuss the derivation in detailed steps.

### 2.1.2 Le Gall’s 5/3 Filter

[35] solved the PR condition by substituting $D(z) = a_0 + a_2 z^{-2} + a_3 z^{-3} + a_2 z^{-4} + a_0 z^{-6}$ with the condition $a_0 \in [-\frac{1}{8}, 0]$. For $a = \frac{1}{16}$ the simplification leads to the famous Le Gall’s 5/3 filter pair. The coefficients for this filter are given in Table 2.2. This filter has lower latency than the ones studied earlier but provides lesser image compression capabilities.

\[
\text{low}_{53}(i) = \frac{3}{4} \times x(i) + \frac{1}{4} \times (x(i - 1) + x(i + 1)) - \frac{1}{8} \times (x(i - 2) + x(i + 2)) \tag{2.4}
\]
\[
\text{high}_{53}(i) = x(i) - \frac{1}{2} \times (x(i - 1) + x(i + 1)) \tag{2.5}
\]

### 2.2 Background and Related Work

Our Poly-DWT architecture must enable dynamic control of the allocated resources in order to yield high performance subject to many external parameters. Although this architecture serves different needs depending on the target multimedia application, one constant across many variations is the use of wavelets for high-quality compression of image or video data.

Recent works in partial reconfiguration of FPGAs provide an insight into the state-of-the-art. [26] gives a comparison of embedded reconfigurable video-processing architectures. They propose a hybrid of two hardware platforms: one providing easy reconfiguration of modules and the other providing easy implementation with higher clock frequency, to achieve an optimal FPGA-based dynamically and partially reconfigurable platform for real-time video and image processing. The tool ReCoBus-Builder [48] simplifies the generation of dynamically reconfigurable systems to almost a push button process. The work also describes a communication infrastructure for dynamically reconfigurable systems. [27]
presents an IP core that enables fast on-chip dynamic partial reconfiguration close to the maximum achievable speed. [83] present a scheme for self-optimization of power and performance according to the run-time specific requirements. The work discusses power optimization of signal routing for application-specific dynamic performance requirements.

Contrary to the above mentioned approaches, in this paper we refer to ‘reconfiguration’ as the dynamic switching of hardware architectures to save power resources. Thus, this switching can be implemented in both FPGA-based and ASIC-based designs. We next discuss the existing work and developments in the theory of wavelet transform and presents the motivation for hardware implementation of this algorithm. Section 3.1 discusses the development of the theory of wavelet transform, and its efficient image processing capabilities. Section 3.2 describes some recent attempts at implementing DWT on reconfigurable platforms.

2.2.1 Wavelet Transform Background

The efficient representation of time-frequency information by the wavelet transform has lead to its popularity for signal processing applications. DWT provides superior rate-distortion and subjective image quality performance over existing standards. Applying a 2-D DWT to an image of resolution $M \times N$ results in four images of dimensions $\frac{M}{2} \times \frac{N}{2}$: three are detailed images along the horizontal (LH), vertical (HL) and diagonal (HH), and one is coarse approximation (LL) of the original image. LL represents the low frequency component of the image, while LH, HL, and HH represent the high frequency components. This LL image can be further decomposed by DWT operation. Three levels of such transforms are applied and shown in Figure 2.4. The coarse information is preserved in the LL3 image and this operation forms the basis of Multi-Resolution Analysis for DWT [107].

Spectral factorization in the frequency domain and lifting schemes are the two common schemes for achieving wavelet decomposition. The spectral factorization method first pre-assigns a number of Vanishing Moments on the Bi-orthogonal Wavelet Filter Banks (BWF Bs), then obtains a trigonometric polynomial (known commonly as a Lagrange Half-Band Filter or LHBF) and then the filter coefficients are determined according to the perfect reconstruction condition. As will be seen in the following section, we implement a spectral factorization based approach which also obtains a low hardware
BWFBs are commonly used for image processing but they have irrational coefficients, the associated DWT requires a high precision implementation, leading to an increased computational complexity. In a hardware implementation, rational binary coefficients can help in achieving a multiplier-free implementation of filter coefficients [70, 86]. These multiplier-free implementations involve image reconstruction quality trade-offs. Many other researchers have also faced the problem of reducing DWT complexity [4, 88, 69]. What differentiates our work is that we consider applications that could make use of run-time (not one-time) hardware resource allocation. To fulfill this requirement we design a new polymorphic architecture that can enable dynamic control over the properties of the allocated hardware resources.

2.2.2 Hardware Implementation of DWT

Much research has been done in the development of DWT architectures for image processing [11, 12, 88, 49, 70]. A good survey on architectures on DWT coding is given by [104]. The paper gives insight on hardware implementations for JPEG2000 scheme which is based on DWT computations. The computational complexity analysis of JPEG2000 by [2, 55] explains that EBCOT coding and DWT operations together contribute more than 80% of the overall complexity. More details of the JPEG2000 standard are given in [24, 96].
The DWT architectures can be broadly classified into lifting based, convolution-based and B-spline based architectures. The lifting based architectures are popular and became the mainstream because they need fewer multipliers and adders and have a regular structure. Similarly B-spline-based architectures have been proposed to minimize the number of multipliers by using B-spline factorization [41]. However, the lifting based architecture has a larger critical path. Convolution-based approaches have a lower critical path but require a larger number of multipliers.

In this paper, we rationalize the filter coefficients which over-rides the past limitations of convolution based approaches. We introduce a multiplier-free implementation and further introduce a switching structure that enables efficient hardware resource sharing between low and high pass filters of DWT. By pipelining we are able to achieve a good performance with our approach.

Chang et al. [20] propose several optimization techniques aimed at providing the developer with more control over the area-to-error trade-off during data path precision optimization that would not be available with simple truncation. An error model is developed for adder and multiplier circuits. However, one of the problems faced is the uncertainty in actual error of the system which depends on the actual value of the input. The upper bound on error skews toward larger positive values as we reduce the bit allocated per pixel. In this paper we make use of a dynamically reconfigurable architecture to modify the resource allocation for the system based on the image quality required by the application. Benkrid et al. [12] discuss that the overall performance and area depends significantly on the precision of intermediate bits used in the design. This motivates us to further look at bit allocation as another aspect of polymorphism in our Poly-DWT structure.

[70] propose a multiplier-free VLSI architecture for the famous 9/7 wavelet filters. The novelty of their architecture is the possibility of combining the 5/3 wavelet data path into the 9/7 data path, resulting in a reduced number of adders compared to other solutions. This implementation approximates the filter coefficients into two decimal places of accuracy and then implements a folded structure for achieving a hardware-efficient DWT implementation. This implementation requires 19 adders, an improvement over 21 adders required in their previous implementation [69]. Our work obtains a different expression for wavelet filter coefficients to obtain all binary rational coefficients. This reduces the number of adders required by our implementation significantly and also achieves significantly bet-
ter image reconstruction results over the original filter. As will be described in Section 4, our folded implementation reduces the number of adders to just 9.

In [102], the author presents a technique to rationalize the coefficients of wavelet filters that will preserve bi-orthogonality and perfect reconstruction. This approach also preserves regularity of the structure by preserving most of the zeros at $z = -1$. This approach has been developed further in this paper to facilitate the development of a polymorphic structure.

## 2.3 Poly-DWT Filter

A look at Table 2.1 explains the inherent difficulties in the hardware implementation of the original Daubechies 9/7 filter. While this filter has high compression performance, it will lead to lossy compression due to truncation involved in filter coefficients in a reasonable fixed point hardware representation such as a 16-bit representation (12-bits for integer and 4-bits for fractional part values). The number of bits required for accurate representation increases as we increase the number of levels of wavelet decomposition. On the other hand floating point implementation implies a higher hardware cost (32 bits for single precision representation). Moreover hardware multipliers would be needed to implement this in our design with reasonable precision.

We alleviate this problem by searching for an integer coefficients filter that can offer a higher PSNR at a smaller word size. A parameterized filter design allows us to obtain a family of 9/7 filters. This new design is then searched for rational coefficients to obtain new filters to alleviate the above mentioned problems.

### 2.3.1 Parameterized Filter Design

A parameterized design alleviates the problem of irrational coefficients. [102] poses this constraint on $D(z)$ to derive the binary rational coefficients and achieve new sets of 9/7 filters by adding more
Table 2.3 Analysis high pass filter coefficients ($H_1$) for the bi-orthogonal 9/7 tap filter

<table>
<thead>
<tr>
<th>$i$ \ $\alpha$</th>
<th>1.6848</th>
<th>-1.667</th>
<th>-1.8</th>
<th>-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pm 3$</td>
<td>0.091271763114</td>
<td>1/16</td>
<td>1/16</td>
<td>1/16</td>
</tr>
<tr>
<td>$\pm 2$</td>
<td>-0.057543526229</td>
<td>-1/16</td>
<td>-1/16</td>
<td>0</td>
</tr>
<tr>
<td>$\pm 1$</td>
<td>-0.591271763114</td>
<td>-9/16</td>
<td>-9/16</td>
<td>-9/16</td>
</tr>
<tr>
<td>0</td>
<td>1.11508705</td>
<td>9/8</td>
<td>9/8</td>
<td>1</td>
</tr>
</tbody>
</table>

degrees of freedom to the original LHBF equation (by introducing a free parameter $\alpha$):

\[
H_0(Z) = K_h (Z + 1)(Z^3 + AZ^2 + VZ + C) \quad (2.6)
\]

\[
G_0(Z) = K_g (Z + 1)^2(Z + \alpha) \quad (2.7)
\]

\[
D(Z) = K_h K_g (Z + 1)^3(Z + \alpha)
\times (Z^3 + AZ^2 + BZ + C). \quad (2.8)
\]

The PR condition on $D(Z)$ gives simultaneous constraint equations which simplify to give solutions for $A$, $B$, and $C$ (and simultaneously for the filter coefficients) in terms of $\alpha$:

\[
A = -(3 + \alpha) \quad (2.9)
\]

\[
B = \frac{9\alpha^3 + 35\alpha^2 + 48\alpha + 24}{3\alpha^2 + 9\alpha + 8} \quad (2.10)
\]

\[
C = \frac{8(1 + \alpha)^3}{3\alpha^2 + 9\alpha + 8}. \quad (2.11)
\]

Here, we have $Z = (z + z^{-1})/2$. Setting $\alpha$ to $-1.6848$ gives back the original 9/7 filter pair.

2.3.2 Numerical Study

The parameter $\alpha$ can be varied to achieve a family of bi-orthogonal filter pairs for DWT implementation. Setting $\alpha = -1.6848$ gives us the CDF-9/7 filter which have been proven to have good compression performance. Next, we perform a numerical study to explore a set of binary coefficients filter which is in close proximity to the CDF-9/7 filter. We ran MATLAB experiments to obtain the quanti-
Figure 2.5  Numerical analysis of quantization error for seven bit finite representation of filter coefficients

Quantization error for the filter coefficients with $\alpha$ varying from $-1.6$ to $-2$ (in vicinity of the $\alpha = -1.6848$ value). The results are presented in Fig. 2.5. It can be observed that the minimization of this error occurs at $\alpha = -2$, where quantization error drops down to 0. A zero quantization error indicates that the filter coefficients derived with $\alpha = -2$ are (exactly) rational. We can also observe local minima of curves around two regions in the vicinity of $\alpha = -1.6848$ (at $\alpha = -1.66$ and $\alpha = -1.8$ approximately). We also derive approximate filter coefficients from these minima to obtain a binary coefficients 9/7 filter. These filter coefficients are reported in Tables 2.3 and 2.4.

Table 2.4  Analysis low pass filter ($H_0$) coefficients for the bi-orthogonal 9/7 tap filter

<table>
<thead>
<tr>
<th>$i$</th>
<th>$\alpha$</th>
<th>1.6848</th>
<th>-1.667</th>
<th>-1.8</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>±4</td>
<td></td>
<td>0.026748757411</td>
<td>1/32</td>
<td>1/32</td>
<td>1/64</td>
</tr>
<tr>
<td>±3</td>
<td></td>
<td>-0.016864118443</td>
<td>-1/32</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>±2</td>
<td></td>
<td>-0.078223266529</td>
<td>-1/16</td>
<td>-3/32</td>
<td>-1/8</td>
</tr>
<tr>
<td>±1</td>
<td></td>
<td>0.266864118443</td>
<td>9/32</td>
<td>1/4</td>
<td>1/4</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>0.602949018236</td>
<td>19/32</td>
<td>5/8</td>
<td>23/32</td>
</tr>
</tbody>
</table>
2.3.3 Candidate Filters

Let us consider an input signal \( x(i) \). The low and high pass outputs of this filter (\( \text{low}(i) \) and \( \text{high}(i) \) respectively) are obtained by convolution of \( x(i) \) with \( h_0(i) \) and \( h_1(i) \), respectively:

\[
\text{low}(i) = \sum_{k=-4}^{k=4} h_0(k) \cdot x(i-k),
\]

\[
\text{high}(i) = \sum_{k=-3}^{k=3} h_1(k) \cdot x(i-k).
\]

Substituting the values of filter coefficients from Tables 2.2, 2.3, and 2.4 we can factorize our 9/7 filter coefficients in terms of 5/3 filter outputs. The subscripts \( A \), \( B \), and \( C \) are used to denote the filters obtained with \( \alpha = -1.67 \), -1.8, and -2, respectively:

\[
\text{low}_A(i) = \frac{19}{32} \times x(i) + \frac{9}{32} \times (x(i-1) + x(i+1)) - \frac{1}{16} \times (x(i-2) + x(i+2))
- \frac{1}{32} \times (x(i-3) + x(i+3)) + \frac{1}{32} \times (x(i-4) + x(i+4))
\]

\[
\text{high}_A(i) = \frac{9}{8} \times x(i) - \frac{9}{16} \times (x(i-1) + x(i+1)) - \frac{1}{16} \times (x(i-2) + x(i+2))
+ \frac{1}{16} \times (x(i-3) + x(i+3))
\]

\[
\text{low}_B(i) = \frac{5}{8} \times x(i) + \frac{1}{4} \times (x(i-1) + x(i+1)) - \frac{3}{32} \times (x(i-2) + x(i+2))
+ \frac{1}{32} \times (x(i-4) + x(i+4))
\]

\[
\text{high}_B(i) = \frac{9}{8} \times x(i) - \frac{9}{16} \times (x(i-1) + x(i+1)) - \frac{1}{16} \times (x(i-2) + x(i+2))
+ \frac{1}{16} \times (x(i-3) + x(i+3))
\]

\[
\text{low}_C(i) = \frac{23}{32} \times x(i) + \frac{1}{4} \times (x(i-1) + x(i+1)) - \frac{1}{8} \times (x(i-2) + x(i+2))
+ \frac{1}{64} \times (x(i-4) + x(i+4))
\]
The Daubechies 9/7 filter has $\alpha = -1.68$. Thus, the compression performance of A will be slightly greater than B and C. However, we can also see that the C architecture requires fewer number of addition operation. The simpler coefficients value in C (coefficients being 0 or easily represented in exponents of 2) promises a cheaper hardware implementation. This implies a trade-off between image reconstruction quality vs. hardware resources required by various filters. In the next subsection we discuss the hardware resource requirements of these architectures.

### 2.3.4 Hardware Architectures

We performed several optimization steps to reduce the cost of underlying hardware. The following optimization steps were performed:

- Observe in Tables 2.2, 2.3, and 2.4 that the coefficients of $x(i \pm k)$ are the same. Thus they can be grouped together to reduce the hardware complexity.

  \[
  w_0 = x(0), \quad (2.20) \\
  w_1 = x(i - 1) + x(i + 1), \quad (2.21) \\
  w_2 = x(i - 2) + x(i + 2), \quad (2.22) \\
  w_3 = x(i - 3) + x(i + 3). \quad (2.23)
  \]

The Daubechies 9/7 filter requires 9 data values - four each corresponding to four previous and next values and one for the present pixel value. After this optimization, we reduced this number from nine to five. This also reduces the requirement of multipliers in implementing equations such as Eq. 12 and 13 in hardware from nine to five.

- Division by binary coefficients (e.g. $1/64, 1/16, 1/4$) was performed using arithmetic shift operations. This eliminates the need for multipliers in the circuits. The coefficients as given in Tables 2.3, 2.4 are rational and most of them have some simple binary value. Therefore we switch our design to a multiplier-free design requiring limited adders in the implementation.
(a) $\alpha = -1.67$

(b) $\alpha = -1.8$
Figure 2.6  Hardware architectures for bi-orthogonal 9/7 filter
• The input stream was pipelined. Thus, as shown in Fig. 2.6 our architecture takes one pixel (or channel input) as the input and outputs the low and high pass signal coefficients with a finite latency. This helps us to achieve a good throughput and a higher clock frequency. The pipeline stages are implemented by clocking the cascaded registers to the left in the figure.

Figure 2.6(a-c) provides a visual overview of the three designs with the value of \( \alpha = -1.67, -1.8 \) and \(-2\) respectively. As can be seen in Fig. 2.6, our Le Gall’s 5/3 filter implementation requires only six adder/subtractor units. Our 9/7 filter implementations for \( \alpha = -1.67 \) required 19 adders. For \( \alpha = -1.8 \), our design requires 17 adder/subtractor units. But we observe that the design for \( \alpha = -2 \) requires only 12 adder/subtractor units. This is a significant improvement over any reported existing work as reported in the experiment section.

As described in Fig. 2.1, the reconfigurable implementation must allow dynamic switching between wavelet filters. Our implementation allows for easy enabling and disabling of the extra hardware to obtain the choice between a more power-efficient binary 5/3 filter versus a more compression-efficient 9/7 filter. In the remainder of this section we describe an architecture to allow for this dynamic switching.

Let us consider an input signal \( x(i) \). The low and high pass outputs of this filter (\( low(i) \) and \( high(i) \) respectively) are obtained by convolution of \( x(i) \) with \( h_0(i) \) and \( h_1(i) \) respectively:

\[
low(i) = \sum_{k=-4}^{4} h_0(k) \cdot x(i-k), \quad 2.24
\]

\[
high(i) = \sum_{k=-3}^{3} h_1(k) \cdot x(i-k). \quad 2.25
\]

Substituting the values of filter coefficients from Table 2.2, 2.3, and 2.4, we can factorize our 9/7 filter coefficients in terms of 5/3 filter outputs.

\[
low_A(i) = 1/2 \times low_{5/3}(i) - (1/4 + 1/16) \times high_{5/3}(i)
+ (1/2 + 1/32) \times w_0 + 1/32 \times (w_4 - w_3) \quad 2.26
\]

\[
high_A(i) = 1/2 \times low_{5/3}(i) + (1/2 + 1/4) \times high_{5/3}(i)
- 1/4 + 1/16 \times w_1 + 1/16 \times (w_3) \quad 2.27
\]
\[
\text{low}_B(i) = \frac{1}{2} \times \text{low}_{53}(i) + \frac{1}{4} \times \text{high}_{53}(i)
+ \frac{1}{32} \times (w_4 - w_3)
\] (2.28)

\[
\text{high}_B(i) = \frac{1}{2} \times \text{low}_{53}(i) + (\frac{1}{2} + \frac{1}{4}) \times \text{high}_{53}(i)
- (\frac{1}{4} + \frac{1}{16}) \times w_1 + \frac{1}{16} \times (w_3)
\] (2.29)

\[
\text{low}_C(i) = \text{low}_{53}(i) - \frac{1}{32} \times w_0 + \frac{1}{64} \times w_4
\] (2.30)

\[
\text{high}_C(i) = \frac{1}{2} \times \text{high}_{53}(i) - \frac{1}{32} \times w_1
+ \frac{1}{32} \times w_3
\] (2.31)

Figure 2.6 (a-c) provides the implementation details of these architectures. The dark (yellow) region is the hardware required for the implementation of Le Gall’s 5/3 filter. The architecture has registers, adders, and multiplexers. The right shift operation (can be implemented by adjusting the wires) is represented by small triangles. A triangle with the number ‘x’ means a shift to the right over ‘x’ positions, or a division by \(2^x\). All the architectures are designed as extensions of Le Gall’s 5/3 filter. This gives the feature of ‘on the fly’ switching from 9/7 filter to Le Gall’s mode of operation.

The low and high pass filter outputs can be \(\text{low}_{A/B/C}(i)\) and \(\text{high}_{A/B/C}(i)\), or \(\text{low}_{53}(i)\) and \(\text{high}_{53}(i)\) depending upon the mode of operation. When operating in 5/3 filter mode only the yellow shaded region of the architecture would be used thus reducing considerably the power consumption of the system. This figure shows the conceptual design and architecture and does not include the pipeline stages of these structures. A folded architecture can be developed for the \(\alpha = -2\) case where the low and high pass output coefficients are dependent only on low and high pass values respectively of 5/3 filter. This is presented in Fig. 2.6(d). This design requires only 9 adders in the circuit.

### 2.4 Fixed Point Implementation

An image channel is generally represented at 8-bit precision. This encourages us to develop a fixed point hardware. We avoid the floating point implementation of the system to avoid non-optimal usage of resources. Chang et al. [20] discuss the issues involved in the fixed point analysis with respect to
the output error. There are two conflicting issues that affect the decision to decide the hardware bit allocation for internal representation of variables:

(a) Increased number of bits generally implies better performance in terms of image quality and reduced error.

(b) Reduced number of bits imply a better hardware utilization, and lower power consumption.

For certain applications such as a static HDTV encoding system we may always require a large number of bits that ensure high quality and high resolution multimedia transmission. However, certain applications such as remote tele-medicine applications and remote distributed surveillance applications are highly power and performance sensitive. They may require a dynamic trade-off. The Poly-DWT provides a good trade-off in achieving a dynamic hardware reconfiguration for such applications. Similarly [20] report that the error (in image reconstruction in case of DWT) is skewed or biased, only in the positive direction. Thus static analysis may not be applicable in all situations and we need custom hardware to adapt itself according to the present conditions. The image statistics (like Peak Signal to Noise Ratio (PSNR), Mean Absolute Deviation (MAD)) provide the system a performance feedback and allows it to take steps to lead to a more efficient representation. These metrics can be used as performance measure of image compression systems and we can switch hardware to reach a desired compression level with minimum hardware resources.

We present a simple scheme to change the bit allocation for hardware implementation. The main factors or sources for the change in hardware bit allocation can be summarized in the following headings:

- **Functional Requirements of the Chip** There may be several computational kernels such as image enhancement, noise filtering, etc, which may be optionally required for a multimedia application. Depending on input from the source some of them may not be required to be functional at all times. The extra hardware available in such cases can be dedicated to the Poly-DWT architecture to improve its performance.

- **Quality Requirements of the Application** Many DWT kernels or instances of DWT hardware may be required by different applications. Moreover, with the change in input images we may
dynamically require different levels of accuracy.

- **Level of Decomposition using DWT** In image compression algorithms such as SPIHT, CEZW, and EBCOT more than one level of DWT operation is done. [33] discuss the changes in numeric range in higher level decomposition using DWT. For example, the eight bit input can have maximum magnitude of 255 and can be well represented using 8.0 fixed point format representation (8 bits to represent integer and 0 bits for fractional part). An analysis of the coefficients of each filter bank shows that a 2-D low-pass FIR filter at most increases the range of possible numbers by a factor of 2.9054. As a result, the coefficients at different wavelet levels require a variable number of bits above the decimal point to cover their possible ranges. At fourth wavelet decomposition level, 17 bit representation may be required to accommodate the magnitude range of coefficients. A dynamic word width allocation may make a lower level DWT kernel fit for decomposition at higher level if required by the application.

- **User Preferences** In our proposed system, the user has the final say in all the subjective image quality/cost trade-offs. Applications and users may differ in their subjective view of good performance of the system. [20] also discuss the importance of defining a user defined error constraint.

- **Other considerations** A Poly-DWT implementation may include other considerations like the number of DWT kernels required, separation/folding of row and column processing DWT kernels etc. We have not discussed these aspects in our present Poly-DWT analysis and they are left as a future work.

### 2.5 Hardware (Re)-Allocation

Poly-DWT allows several levels of hardware resource (re-)allocation to obtain a power-efficient design, which are explained as follows:

1. The number of DWT kernels in the wavelet decomposition can be varied depending upon the application requirements.

2. On-the-fly switching of filter design from 9/7 to 5/3 filter architecture in finite cycles latency.
3. The number of bits allocated for internal registers in the design can be varied to obtain an application-specific trade-off between clock frequency and reconstruction quality vs. hardware usage.

The variations in number of DWT kernels in wavelet decomposition is specific to the requirements of the multimedia encoding scheme and its dynamic requirements. In this paper, we therefore restrict our discussion to reconfiguration of design of the individual DWT kernels to meet the performance vs. power trade-off dynamically in hardware. Figure 2.7 gives the architecture design of poly-DWT kernel to achieve these trade-offs.

2.5.1 ‘On-the-fly’ Switching

We first consider ‘on-the-fly’ switching of filter designs from 9/7 to 5/3 architectures. The \( switch_{hw} \) signal in Figure 2.7 is used to switch between 5/3 and 9/7 architectures. The two multiplexers (unshaded in Figure 2.7) ensure the correctness of the input and output of poly-DWT hardware. As seen in this figure, we can divide the hardware into two categories:

1. **Type A hardware.** The hardware common to both 5/3 and 9/7 filter architectures is called as type A hardware and it is unaffected by ‘on-the-fly’ switching. This includes the registers and adders in the shaded portion of design in Figure 2.7.

2. **Type B hardware.** The hardware used by 9/7 filter architecture which is obsolete to 5/3 filter is called as type B hardware. This hardware is switched off when \( switch_{hw} \) signal is changed to 0. This corresponds to the registers and adders in the unshaded portion of design in Figure 2.7.

The following steps are involved in switching from 9/7 to 5/3 filters (the 5/3 filter hardware is shaded in Figure 2.6):

1. The input pipeline for the 5/3 filter is smaller than the 9/7 filter. In order to use the same pipeline we need a latency of two cycles to ensure that the pipelining registers have proper inputs. The values in the pipeline registers \( x(i - 4) \) and \( x(i - 3) \) are pipelined to the 5/3 filter hardware before they are switched off.
2. The extra hardware for computation can be switched off in a single clock cycle. This can be enforced by driving the signal $\text{switch}_{hw}$ from 1 to 0 (shown in Figure 2.8(b) and explained in next subsection).

3. The input and output multiplexer can be switched from input port 1 to input port 0 in one cycle.

Since the above-mentioned operations can be performed together, we require only a latency of two cycles to switch from a 9/7 to a 5/3 filter. A similar argument can be constructed to explain that it would take a latency of two cycles to switch from 5/3 to 9/7 filter architecture.

While the proposed architecture is capable of switching between 5/3 and 9/7 filter architectures at run-time of a few $ns$, such a design will incur a large overhead in transmitting control information to ensure the correctness of the output at the decoder (We will need to send 1 bit per clock cycle for one filter kernel used). However, in practical scenarios we can restrict the switching between 5/3 and 9/7 filters between different levels of wavelet decomposition. Thus, the overhead involved in such a switching is reduced to a few bits (3-10 bits per frame) and can be integrated into frame header.
2.5.2 ‘Bit-width’ Switching

We discuss the scheme for bit-width switching in this section. We break the internal registers in design into multiples of four bit registers. Thus, a $N = 16$ bits register is represented as four 4-bit registers. As shown in Figure 2.7, the registers are represented as four 4-bit registers. Figure 2.8 explains the working of ‘bit-width’ switching scheme with individual registers. The four signals $R_4$, $R_8$, $R_{12}$ and $R_{16}$ are used to switch the registers on or off at run-time. When $R_4$, $R_8$, and $R_{12}$ are on, the register has 12 bits available for use while the other 4-bit register is switched off to save power. This is done with the help of chip enable (CE) signal as indicated in the Figure 2.8. Similar changes can be made to the design of adders to partially switch off the LUTs corresponding to an adder hardware. Figure 2.8(a) explains the bit-width switching of type A hardware. The two inputs $\text{switch}_{hw}$ and the register select input ($R_4/R_8/R_{12}/R_{16}$) are ANDed to get the chip-enable (CE) signal for individual 4-bit type B registers. This enabling/disabling of registers for type B hardware is illustrated in Figure 2.8(b).

The dynamic power consumption of a circuit is given by the following equation:

$$P = ACV^2F$$

where $A$ is the activity factor ($0 \leq A \leq 1$), $C$ is the switched capacitance of the circuit, $V$ is the supply voltage and $F$ is the clock frequency. By switching off the extra hardware we reduce the
switched capacitance \( C \) of the circuit, thereby obtaining a useful dynamic trade-off between the power and performance constraints.

### 2.6 Experiments

This section presents quantitative results for the performance of Poly-DWT architecture presented in this paper. We evaluate our approach on the Xilinx Virtex-V XC5VLX30 FPGA by generating the different DWT architectures. The polymorphic architecture presented in this paper has been analyzed in terms of image reconstruction and kernel area considerations. As previously mentioned, the trade-off between the two is dynamically reached in a polymorphic architecture.

We present the results of analysis for various word widths for internal and external configurations of DWT kernel and also examine the performance of different kernels. The standard color test images (e.g. Lena, Barbara) were used for the purpose of simulation. Each pixel in a color image has 3 channels, with 8 bits of data per channel. Unless otherwise specified, we used 8.4 fixed point arithmetic for internal computations.

Our design is written in VHDL and synthesized using Xilinx ISE 9.1i. ModelSim simulations were performed to test the waveforms. The more detailed analysis of image reconstruction performance of various filters is performed in MATLAB. To verify the correctness of the various filters implemented in the FPGA, we compared it against a pure software implementation on a Intel Core 2 Duo processor running at 2.0 GHz. Both implementations generate the same numerical results for transformed output. In the following subsections, we analyze the working of our proposed DWT hardware with respect to area, performance and quality perspectives.

#### 2.6.1 Image Reconstruction Quality

The proposed Poly-DWT filter gives a more efficient representation than the original Daubechies 9/7 filter as well as the Le Gall’s 5/3 filter as illustrated in Figure 2.9(a). It can be seen in the figure that Poly-DWT provides provides very little high pass information (white marks in black background in higher frequency subbands). The reduction in high level information in our Poly-DWT filter makes it more suitable for the compression applications.
Figure 2.9 (a) Results of one level of DWT and (b) Energy decomposition by respective filters

A more accurate representation over fixed point hardware gives a better image reconstruction for Poly-DWT filter than the Daubechies 9/7 filter. Results over several test images showed similar results. The bars in Figure 2.9 illustrate the superior performance of the Poly-DWT filter for limited hardware resources. The ratio of energy of the low and high pass components is measured. Poly-DWT is found to outperform other filters in retaining low pass energy. This property, also known as energy compaction property of the filter is helpful to achieve a better compression efficiency.

The image compression performance of Poly-DWT filter was evaluated on a SPIHT image coder [91]. We tested the performance on an open-source filter bank based implementation provided by [Tian]. We chose the intermediate variables in 9.4 fixed point format for this experiment setup. In case of low bit-rate applications, this property helps in better reconstruction of images from low pass coefficients. The performance over some test sequences has been reported in Table 2.5. The results are reported over bit rates of 0.5 bpp (bits per pixel) and 2 bpp. It can be seen that the compression efficiency of Poly-DWT filter is comparable to Daubechies 9/7 filter. A performance comparison with another multiplier-free implementation provided by [70] illustrates that our design requires a fewer number of adders and gives a higher compression performance as evident by higher PSNR values.
Table 2.5 Image compression performance on SPIHT coder (PSNR values).

<table>
<thead>
<tr>
<th>Image</th>
<th>Bitrate=0.5 bpp Daub. 9/7</th>
<th>Bitrate=0.5 bpp Poly-DWT</th>
<th>Bitrate=0.5 bpp Martina,07</th>
<th>Bitrate=2 bpp Daub. 9/7</th>
<th>Bitrate=2 bpp Poly-DWT</th>
<th>Bitrate=2 bpp Martina,07</th>
</tr>
</thead>
<tbody>
<tr>
<td>lena</td>
<td>28.213</td>
<td>29.46</td>
<td>27.7</td>
<td>38.47</td>
<td>38.17</td>
<td>36.5</td>
</tr>
<tr>
<td>surveillance</td>
<td>26.1</td>
<td>28.1</td>
<td>26.54</td>
<td>38.41</td>
<td>42.21</td>
<td>39.21</td>
</tr>
<tr>
<td>lecture</td>
<td>34.35</td>
<td>33.8</td>
<td>32.73</td>
<td>48.3</td>
<td>51.25</td>
<td>43.71</td>
</tr>
<tr>
<td>helicopter</td>
<td>33.75</td>
<td>35.7</td>
<td>35.01</td>
<td>48.59</td>
<td>54.72</td>
<td>47.14</td>
</tr>
</tbody>
</table>

2.6.2 Hardware vs Software Performance

The hardware performance of DWT kernels proposed in the paper was compared with a software based implementation on the PC platform. Table 2.6 gives the speedup achieved by an FPGA based implementation of DWT kernels. The software implementation of both Daubechies 9/7 filter and Poly-DWT (9/7) filter takes the same time as the number of filter taps in both cases is the same. The FPGA based design outputs one pixel per clock cycle for every DWT kernel. The computation times for one level of DWT for different image sizes is presented in Table 2.6. The is reported in microseconds (μs). The proposed Poly-DWT filter obtains a speedup of about a factor of 10 for CIF images (standard images of size 352×288 pixels). The speedup for Q-CIF images (Quarter-CIF) is also about 10. The smaller speedup in smaller sized images is attributed to the overheads in I/O operations which are more significant in the case of small sized images. A line based image scan architecture [25] is used for data I/O operations.

The results as summarized in Table 2.6 show the advantages of a hardware implementation of this class of algorithms. This is due to the fact that the required calculations are simple, allowing for a high throughput implementation. By pipelining the individual adder and add operations, we were able to achieve very high clock frequencies (394 MHz on our target Virtex-5 platform and 4 bits word length). The actual speedup achieved by the Poly-DWT kernel (Table 2.7) over Daubechies filter is greater (three times more) than the results indicated in Table 2.5 because of memory access computations involved in image compression results.
2.6.3 Hardware Comparison

Direct implementation of the CDF-9/7 filter gave a clock frequency of 107 MHz, while requiring 9 multiplier units. A clock frequency of 110 MHz was reported when we forced the design to map the constant multiplications into Lookup Tables. [70] implement Daubechies 9/7 filter with approximate coefficients and report a clock frequency of about 200 MHz through a multiplier-free implementation, targeting 0.13 \( \mu m \) VLSI technology.

Table 2.7 summarizes the performance of our Xilinx Virtex-V implementation, and compares our results with other recent works. All the parameterized binary implementations outperform the existing implementations in terms of number of required adders and clock frequency.

Our initial non-pipelined design obtained a clock frequency of about 108 MHz, due to its long

\[
\begin{array}{cccccc}
\text{Image} & \text{Le Gall 5/3 Filter} & \text{Daubechies 9/7 Filter} & \text{Poly-DWT 9/7 Filter} \\
& \text{SW} & \text{HW} & \text{Speedup} & \text{SW} & \text{HW} & \text{Speedup} & \text{HW} & \text{Speedup} \\
\hline
\text{CIF} & 1420 & 197 & 7.06\times & 2980 & 330 & 9.03\times & 288 & 10.35\times \\
\text{Q-CIF} & 370 & 68 & 5.45\times & 790 & 91 & 8.68\times & 77 & 10.26\times \\
\end{array}
\]
critical path. The critical path of the circuit lies from the \( w_i \) registers to the final output \( \text{low}_{C(i)} \) or \( \text{high}_{C(i)} \), passing through signals \( \text{low}_{C(i)} \) or \( \text{high}_{C(i)} \). We then pipelined this computation into several stages and obtained a faster implementation. The \( \alpha = -2 \) architecture showed a clock frequency of about 317 MHz. This design requires less FPGA resources (registers and LUTs) than the \( \alpha = -1.67 \) and \( \alpha = -1.8 \) architectures and is most fit for Poly-DWT implementation.

The folded architecture variant for \( \alpha = -2 \) was also implemented, resulting in a faster clock frequency and less adders (leading to fewer logic slices). The design of binary coefficients filter also helped us to achieve perfect reconstruction of image signals. This proposed architecture can run (over line-based DWT architectures) at 389 MHz, enabling it to process High Definition Video frames \((1440 \times 1080)\) in an estimated 5 ms time. As previously mentioned, the shaded (yellow) regions in Fig. 2.6 show the baseline 5/3 filter implementation. Thus the architecture can be optimized to switch on-the-fly to 5/3 mode in order to save power. The folded architecture and the simple architecture of Poly-DWT filter both have the same performance in terms of image reconstruction and they differ only in hardware requirements. The input data width was 8 bits corresponding to one channel of an image stream. The proposed binary filter reaches perfect reconstruction with lesser number of bits than the Daubechies 9/7 filter. Thus the overall area requirements are less. The hardware resources utilized in these DWT kernels are summarized in Table 2.7. Here, a comparison of hardware resources utilization is provided against existing works. [70] present a multiplier-free implementation which is suitable for polymorphic switching between 9/7 and 5/3 filters. However, they approximate the original Daubechies filter coefficients to two decimal places which leads to its poor PSNR performance. Our architecture provides both more efficient hardware usage and better compression performance.

Figure 2.10 shows the change in synthesized clock frequency for the various implementations of DWT with varying input word width. The change in external data width as shown in Figure 2.10 leads to reduction of clock frequency and hence reduced throughput.

[125] present a switching between 5/3 and 9/7 filters using partial reconfiguration of the bit streams and a lifting based implementation of the DWT. They used a platform based on Xilinx Virtex-4 FPGA for experimental implementation. However, this implementation requires a switching time of 40.2 ms. Thus, this system introduces a delay/ lag of 2 frames (at CCIR resolution of \( 720 \times 576 \) pixels per frame.
and 50 MHz clock). As compared to these results, poly-DWT has a very small switching time of two clock cycles (equivalent to 5.14 ns, assuming a 389 MHz clock).

2.6.3.1 ASIC Synthesis

In order to make a more fair comparison with related work, we also synthesized our Poly-DWT architecture to ASIC technology. We used the Synopsys Design Compiler environment to perform our experiments using the freePDK 45nm cell library [97]. The results of ASIC synthesis indicate that we can achieve a clock frequency comparable to Le Gall’s filter with an insignificant increase in the number of cells in the design (as reported in Table 2.8). We were able to achieve a clock speed of 500 MHz for the folded 9/7 filter design.

2.6.4 Dynamic Bit Allocation

In this subsection we study the effect of bit allocation on the clock frequency and image quality. The implementation used fixed point arithmetic over VHDL. First the input data was kept at 8.0 format and the word width of internal registers was changed. Figure 2.11 shows the change in reconstruction quality of the images depending on changes in hardware resources (single bit registers or flip-flops).
Figure 2.12  Comparison of register usage for the binary filter implementations

The x axis here refers to the total number of bits given to an internal processing register. Figure 2.12 compares the implementation of our structure with other filters. It is observed that changes in bit-width of internal registers from 9.0 to 9.6 fixed point representation leads to a linear increase in hardware requirements (number of single bit registers or flip-flops) and a slight decrease in achievable clock frequency. The folded Poly-DWT filter register usage on an FPGA chip approaches the implementation of 5/3 filter, while its compression performance approaches the Daubechies’ 9/7 filter. This indicates the hardware efficient feature of our design.

2.6.5 Real-World Application

We consider a real-time scenario where we propose a DWT based video-surveillance system. Lake Pontchartrain Causeway in southern Louisiana has a bridge that runs 23.87 miles. A surveillance system featuring 29 cameras mounted at different points along the bridge is used to keep guard with cameras placed at approximately every 3 miles. Employees monitor the bridge traffic with the help of this system. We propose a dynamic power-saving solution using Poly-DWT considering the usage of
surveillance cameras. There are two usages associated with these cameras:

1. **Idle-usage.** Most of the time, the cameras are used for monitoring the traffic and a low resolution version of these 29 images is provided to the users. Essentially, a very coarse version of the input video is provided to the employees at monitoring station.

2. **Active-usage.** When a suspected activity is detected, the employee scans for a high resolution version of the video. A high resolution version of the surveillance video from concerned video camera is sought. This may be the case of traffic congestion, or someone trying to commit suicide or a car broken down mid-way in the bridge.

The 9/7 poly-DWT filter has higher hardware requirements and hence consumes more power than the 5/3 filter. Using Xilinx Xpower analyzer for our Xilinx Virtex-5 FPGA, we obtain a power consumption of 0.34W for the 5/3 filter and 0.46W for 9/7 filter. Using the poly-DWT filter during active-usage time and switching to the 5/3 filter during idle-usage time will save us 0.12W power. The respective values were 0.0477 W for 5/3 filter and 0.06 W for 9/7 filter using low power Xilinx Spartan FPGAs.

Most of the time (nearly 99 percent of time) is idle-time for each camera. We get a power saving by a factor of \( \frac{0.46}{0.46 \times 0.99 + 0.34 \times 0.01} = 1.348 \) (for Virtex-5) and by using our poly-DWT filter.

Another practical scenario is the usage of speed cameras for monitoring traffic. Speed cameras use several different types of technology, most commonly lasers or radar, to pinpoint cars that are exceeding the marked speed limit. When a speeding car is detected, the radar or laser signal triggers the camera to record the car’s license plate and that data is used to issue a ticket to the car’s owner. Reading the number plate requires a DWT filter with large taps such as the 9/7 filter. On the other hand, the normal usage of camera can be to monitor traffic (at coarse resolution) which is served better by 5/3 filter. The 9/7 poly-DWT filter can be used to get a more accurate view of car’s license plate when triggered by radar/laser signal triggers while we can switch to Le Gall’s 5/3 filter for keeping a record of traffic movements and also make power-savings.

Time-crucial surveillance applications such as meteorology, remote scientific experiments, defense applications require such rapid switching (in one-two cycles as provided by Poly-DWT) of the hardware architectures.
2.7 Conclusions and Future Work

This paper introduces the concept of polymorphic wavelet architecture for image processing and compression. Polymorphism allows for real-time implementations to dynamically configure the device to allocate hardware resources to suit its instantaneous needs and obtain an area/power optimized design. We presented a low hardware (binary rational) implementation of Daubechies 9/7 filter and its derivation from Le Gall’s 5/3 filter outputs to allow on the fly switching between the transform structures upon the demands of application. Moreover, a study of filter performance with the changes in word width allocation was performed. We discussed how internal hardware resource allocation for computational purpose changes the area / reconstruction quality performance of the DWT kernel. The experiments favored the theory of polymorphic wavelet architecture design for dynamic image compression applications.

As a future work, such architectures can be developed for other image compression modules. Moreover, most aspects of DWT implementation and dynamic reconfiguration can be explored further. For example - the number of DWT kernels utilized in image transform and the multiplexing between row and column kernels can be studied to add yet another dimension of polymorphism to our architecture.
Table 2.7  Comparison of binary filter features and hardware resources requirements

<table>
<thead>
<tr>
<th>Features</th>
<th>Daub. 9/7</th>
<th>$\alpha = -2$ folded</th>
<th>$\alpha = -2$</th>
<th>$\alpha = -1.8$</th>
<th>$\alpha = -1.67$</th>
<th>Tay, 2001</th>
<th>Kotteri, 2005</th>
<th>Huang, 2001</th>
<th>Martina, 2007</th>
<th>Martina, 2005</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adders</td>
<td>15</td>
<td>9</td>
<td>12</td>
<td>17</td>
<td>19</td>
<td>19</td>
<td>15</td>
<td>8</td>
<td>19</td>
<td>21</td>
</tr>
<tr>
<td>Multipliers</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PSNR</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>C</td>
<td>C</td>
<td>A</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>Reconf.</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Registers</td>
<td>144</td>
<td>208</td>
<td>213</td>
<td>253</td>
<td>294</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LUTs</td>
<td>80</td>
<td>175</td>
<td>194</td>
<td>217</td>
<td>289</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Bit Slices</td>
<td>210</td>
<td>245</td>
<td>259</td>
<td>311</td>
<td>375</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Clock(MHz)</td>
<td>107</td>
<td>389</td>
<td>317</td>
<td>311</td>
<td>310</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>200</td>
<td>-</td>
</tr>
</tbody>
</table>
Table 2.8  Performance evaluation on 45nm standard cell libraries

<table>
<thead>
<tr>
<th></th>
<th>Poly-DWT</th>
<th>Le Gall’s</th>
<th>Daub. 9/7</th>
<th>[70]*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area</td>
<td>2135</td>
<td>1370</td>
<td>6693</td>
<td>-</td>
</tr>
<tr>
<td>Cells</td>
<td>544</td>
<td>194</td>
<td>1022</td>
<td>-</td>
</tr>
<tr>
<td>Clock Frequency</td>
<td>500</td>
<td>500</td>
<td>300</td>
<td>200</td>
</tr>
</tbody>
</table>

* [70] reports a gate count of 2.68K using a 130nm cell library.
CHAPTER 3. THE SECURE WAVELET TRANSFORM

There has been an increasing concern for the security of multimedia transactions over real-time embedded systems. Partial and selective encryption schemes have been proposed in the research literature, but these schemes significantly increase the computation cost leading to tradeoffs in system latency, throughput, hardware requirements and power usage. In this paper, we propose a light-weight multimedia encryption strategy based on a modified Discrete Wavelet Transform (DWT) which we refer to as the Secure Wavelet Transform (SWT). The SWT provides joint multimedia encryption and compression by two modifications over the traditional DWT implementations: (a) parameterized construction of the DWT and (b) subband re-orientation for the wavelet decomposition. The SWT has rational coefficients which allow us to build a high throughput hardware implementation on fixed point arithmetic. We obtain a zero-overhead implementation on custom hardware. Furthermore, a Look-up table based reconfigurable implementation allows us to allocate the encryption key to the hardware at run-time. Direct implementation on Xilinx Virtex FPGA gave a clock frequency of 60 MHz while a reconfigurable multiplier based design gave a improved clock frequency of 114 MHz. The pipelined implementation of the SWT achieved a clock frequency of 240 MHz on a Xilinx Virtex-4 FPGA and met the timing constraint of 500 MHz on a standard cell realization using 45nm CMOS technology.

The recent emergence of embedded multimedia applications such as mobile-TV, surveillance, video messaging, and tele-medicine have increased the scope of multimedia in our personal lives. These applications increase the concerns regarding privacy and security of the targeted subjects. Another growing concern is the protection and enforcement of intellectual property rights for images and videos. These and other issues such as image authentication, rights validation, identification of illegal copies of a (possibly forged) image are grouped and studied under the label of Digital Rights Management (DRM).
The computer security protocols (e.g. SSL ([111]), TLS ([18])) and cryptographic ciphers (e.g. AES ([31]), DES ([32]), IDEA ([50])) drive much of the world’s electronic communications, commerce, and storage. These techniques have been used for conventional multimedia encryption and authentication.

In one version of these schemes, some form of private-key encryption algorithm is applied over the full or partial output bit stream from the video compression engine. This naive approach is usually suitable for text, and sometimes for small bitrate audio, image, and video files that are being sent over a fast dedicated channel. Secure Real-time Transport Protocol, or shortly SRTP ([10]), is an application of the naive approach. In SRTP, multimedia data is packetized and each packet is individually encrypted using AES. The naive approach enables the same level of security as that of the used conventional cryptographic cipher.

Consequently, a multimedia compression engine (such as a MPEG or H.264 encoder ([104])) has an additional encryption engine to ensure multimedia security. Depending on the scheme used, the encryption operation is performed either at some intermediate level during compression or after the final compression. However, these cryptographic ciphers require a large amount of computational resources and often incur large latencies. Hardware implementations of AES are often pipelined, leading to a significantly large latency for real-time applications (31 cycles for AES ([39])).

The large data volumes, interactive operations, real-time responses, and scalability features that are inherent to real-time multimedia delivery restrict the practical application of these naive cryptographic schemes. Selective encryption schemes have been proposed in research literature ([61, 56, 68, 16, 30]) to reduce the computational requirements of full encryption schemes. [56] present a scheme for encryption of Discrete Cosine Transform (DCT) coefficients’ signs and watermarking of DCT coefficients. [56] uses Exp-Goloumb codes for the encryption operation. [22] propose a DWT-based partial encryption scheme which encrypts only a part of compressed data. Only 13 – 27% of the output from quadtree compression algorithms is encrypted for typical images. A good summary of efforts in selective or partial encryption of images can be found in [61].

Furthermore, embedded multimedia systems have constraints on power consumption, available computation power and performance. Real-time embedded systems face additional constraints on
power consumption, hardware size and heat generation in the chip which requires design and mapping of computation-savvy encryption schemes for such architectures. Recently, power-aware designs have been proposed for video coding in embedded scenarios ([21]). The authors in [21] propose a multi-mode embedded video codec with DRAM area and external access power savings to support a real-time encoding of CIF images (having resolution of 352x288 pixels). Adding a sequential or pipelined encryption stage to the system in ([21]) will add to system latency and further increase the power/heat budget of such a design.

Such limitations can be alleviated through the development of parameterized compression blocks that can achieve simultaneous encryption. Thus, the compression operation itself uses a key to encode the input data and no external cryptographic engine is required. Recently, some schemes have been developed using this compression-combined-encryption approach. [36] introduce a parameterization in the arithmetic coding stage of multimedia compression. This parameterization is used to build a key scheme. However, the performance of such scheme for embedded systems remains untested. [45] presents a variation of ([36]) that improves the security performance of parameterized arithmetic coding scheme but increases the complexity in hardware implementation.

[65] presents a joint signal processing and cryptographic approach to multimedia encryption. They use index mapping and constrained shuffling to achieve confidentiality protection. This ensures that the encrypted bitstream still complies with the state-of-the-art multimedia coding techniques. The scheme gives good results, however, it requires extra computations (and hence extra hardware resources) to implement such a scheme. [57] presents a multimedia encryption scheme based on wavelet coefficients confusion. However, a scheme based on wavelet coefficients permutations alone is bound to be separable and weak against any cryptanalysis. In this work, we do use a wavelet coefficient permutation called 'subband re-orientation’ which is optimized for implementation without any computation overheads. However, our overall scheme has more parameters that build the key space which prevents an adversary from easily cracking our scheme by parallel brute force trials in the individual sub bands.

Fast Encryption Algorithm for Multimedia (FEA-M) has been proposed for real-time multimedia encryption ([121]). It works with Boolean matrix and can be implemented efficiently on hardware. However, there have been several attacks against such algorithms and proposals have been written to
improve the security ([112]).

This paper presents a multimedia encryption scheme based on parameterized construction of the DWT and subband re-orientation for the wavelet decomposition, called the Secure Wavelet Transform (SWT). An efficient hardware implementation (direct implementation and a Reconfigurable Constant Multiplier (RCM) based implementation) of the SWT using both FPGA and ASIC technology is also presented in this paper. The initial results regarding parameterized construction of the DWT were presented in [81].

Section II gives the theory and mathematical preliminaries of the proposed SWT architecture. Section III discusses the image security provided by the SWT. In Section IV we present an optimized hardware architecture for the SWT. Hardware optimizations, FPGA and ASIC implementation results and a Reconfigurable Constant Multiplier implementation has been presented in this section. Section V concludes the paper with insight of future works.

3.1 Preliminaries

Prior works in signal processing establish that the 1-D DWT can be viewed as a signal decomposition using specific low pass and high pass filters [98]. A single stage of image decomposition can be implemented by successive horizontal row and vertical column wavelet transforms. Thus, one level of DWT operation is represented by filtering with high and low pass filters across row and column respectively. After each filtering stage, down sampling is done by a factor of two to remove the redundant information.

The two most common DWT filters used in image compression are the Le Gall’s 5/3 and the Daubechies 9/7 filters [24], accepted in the JPEG2000 standard. The Le Gall’s filter has rational coefficients and its hardware implementation requires less resources. The Daubechies 9/7 filter has better compression performance, however, it has irrational coefficients and leads to lossy compression. Applying a 2-D DWT to an image of resolution $M \times N$ results in four images of dimensions $\frac{M}{2} \times \frac{N}{2}$. Subsequent levels of DWT-based decomposition yield a multi-resolution structure suitable for image compression.
3.1.1 Parameterized Construction of DWT

There are four filters that comprise the two-channel bi-orthogonal wavelet system. The analysis and synthesis low-pass filters are denoted by \( H_1 \) and \( H_2 \) respectively. The analysis and synthesis high pass filters are denoted by \( G_1 \) and \( G_2 \) respectively and are obtained by quadrature mirroring the low-pass filters.

\[
G_1(z) = z^{-1}H_2(-z), \quad G_2(z) = zH_1(-z)
\]

The Perfect Reconstruction (PR) condition for a DWT filter simplifies to the following:

\[
H_1(z)H_2(z) + H_1(-z)H_2(-z) = 2
\]

[63] present a parameterized construction of Bi-orthogonal Wavelets Filter Banks (typically used for image compression). For even number of vanishing moments, \( H_1(z) \) and \( H_2(z) \) are represented as follows:

\[
H_1(z) = \left( z^{-\frac{1}{2}} + z^{\frac{1}{2}} \right)^{2l_1} \times \left( \alpha + (1 - \alpha) \left( z^{\frac{1}{2}} + z^{-\frac{1}{2}} \right)^2 \right)
\]

\[
H_2(z) = \left( z^{-\frac{1}{2}} + z^{\frac{1}{2}} \right)^{2l_2} \times Q(z)
\]
where

\[ Q(z) = \sum_{n=0}^{3} q_n \times (z^{l_1} + z^{l_2})^{2n}, \quad l_1, l_2 \geq 0, \{l_1, l_2\} \in \mathbb{Z} \]

and \( \alpha \) is the free parameter introduced in the design. The values \( q_n \) are calculated by the following expression:

\[ q_n = \sum_{k=0}^{n} \left( \frac{L + n - k - 1}{L - 1} \right) \left[ 2(1 - \alpha) \right]^k, \quad n = 0, \ldots, L - 1 \]

and

\[ q_L = \frac{1}{2\alpha} \left\{ \left( \frac{2L - k - 1}{L - 1} \right) \left[ 2(1 - \alpha) \right]^k + (1 - 2\alpha) \sum_{n=0}^{L-1} q_n \right\} \]

with \( L = l_1 + l_2 \).

For the 9/7 filter, the values of \( q_n \) were approximated using Taylor’s series expansion and obtained as follows:

\[ q_0 = 1; \quad q_1 = 5 - 2a; \quad q_2 = 4a^2 - 14a + 16; \]
\[ q_3 = 36a - 8a^2 - 60 + 32/a; \]

Simplifying these equations, we get the following expression for \( H_1(z) \) and \( H_2(z) \).

\[ H_1(z) = (-9\alpha/64 + \alpha^2/32 + 15/64 - 1/(8\alpha))(z^4 + 1/z^4) \]
\[ +(-\alpha^2/16 + 11\alpha/32 - 11/16 + 1/(2\alpha))(z^3 + 1/z^3) \]
\[ +(1/8 - 1/(2\alpha))(z^2 + 1/z^2) \]
\[ +(-11\alpha/32 + \alpha^2/16 + 15/16 - 1/(2\alpha))(z + 1/z) \]
\[ +(9\alpha/32 - \alpha^2/16 - 7/32 + 5/(4\alpha)) \]

\[ H_2(z) = (1/32 - \alpha/32)(z^3 + 1/z^3) + (1/8 - \alpha/16)(z^2 + 1/z^2) + (7/32 + \alpha/32)(z + 1/z) + (1/4 + \alpha/8) \]

There are several useful features of parameterized DWT construction that make it suitable for being a part of the SWT:
3.1.1.1 Rational Coefficients

The expressions for $H_1(z)$ and $H_2(z)$ have product of exponents in $\alpha$ and $z$ with rational coefficients. All these rational coefficient multiplication operations can be simplified into shift-add operations. For example, $\frac{A}{16} \equiv A \gg 4$ and $\frac{15B}{64} \equiv (B \gg 2) - (B \gg 6)$ where $\gg$ denotes a right shift operation.

3.1.1.2 Feasible range of parameter $\alpha$

The numerical value of free parameter $\alpha$ can be varied over a wide range while retaining the perfect reconstruction property of the wavelet transform. However, as we vary the value of $\alpha$ over the range $(-\infty, +\infty)$, the output values of the DWT operation have a very large dynamic range requiring a larger number of bits for representation. This would reduce the compression rates achievable with the DWT-based coders.
Numerical experiments show that parameterized DWT has a good PSNR value for image reconstruction with Set-Partitioning in Hierarchical Trees (SPIHT) based coder when \( \alpha \) varies in the range 1 to 3. When \( \alpha \) varies beyond this range, the output DWT coefficients are spread over a large dynamic range. At low bit rates, the encoder is not able to efficiently encode such a large range of input coefficients leading to poor compression results. Figure 3.1 illustrates the significant decline in PSNR values (in db) for \( \alpha > 3 \).

### 3.1.1.3 Key-space

We divide this interval \([1, 3]\) into \(2^m\) sub-intervals. Thus, a one-dimensional DWT operation is represented by \(m\) bits. One level of wavelet decomposition involves successive filtering with row and column filters. If we have \(N\) levels of decomposition using DWT, we can choose different \( \alpha \) values for all \(2^N\) filters (represented by \(16mN\) bits).

The actual choice of \(N\) and the number of sub-intervals is subjective and depends on input images and desired sensitivity of images. For example- the image sequences which are input to highly-crucial image processing applications such as medical imaging can use more sub-intervals while some applications, such as counting the number of cars crossing an intersection - will allow low number of bits. Figure 3.4 shows the MSE (Mean Square Values) for image encoded with one \( \alpha \) value and reconstructed with the adjacent \( \alpha \) value for various bit-width. It can be seen that 5 or less bits give a large MSE (MSE\(_{>8}\)) while some applications may allow \(m=8\).

Figure 3.2 shows the image performance of the parameterized DWT. We took three sample images: the first and third being an aerial survey of some landscape while the second image is a snapshot of Shakespeare’s written text (Scene II from Julius Caesar). The results are presented when an encryption (or image compression) was performed with the \( \alpha \) parameter set to 2.0 and decryption (or image reconstruction) was performed with different \( \alpha \) values. We can see that the images decrypted with the wrong key values (Fig. 3.2 (b, d, e)) have poor visual quality. These images miss many important details of the original scene or text. In this experiment, we have visualized the impact of only using the parameterized DWT and a single key for all levels of decomposition.

It can be seen that wrong guesses for DWT parameter \( \alpha \) leads to high reconstruction errors in
Figure 3.3  (a) Image decomposition with DWT (6 levels) leading to 19 subbands. 3 bits are assigned for each subband’s re-orientation information. (b) Possible transpose relationships for sub bands. A is the original matrix. The eight permutations are achieved using transpose relationship (’), and reverse-ordering of the subbands (− for reverse, + for forward read access) along both rows and columns images. However, we need further dimensions to increase the key space and make image reconstruction more obscure in case of wrong guesses for the key value.

3.1.2 Subband Re-orientation

The parent-child coding gain in the DWT-based coders was quantified by [66]. These dependencies are generally credited for the excellent mean square error (MSE) performances of zero-tree-based compression algorithms such as embedded zero-tree coding of wavelet coefficients (EZW) and SPIHT. The subbands were rotated by 90° with respect to the previous scale prior to zero-tree coding. These experiments indicate that the coding gain due to these dependencies is not considerable for natural images (typically around 0.40 dB for SPIHT-NC and 0.25 dB for SPIHT-AC). However, the image reconstruction quality will considerably change with the rotations of subbands. Simple transformations such as
transposing the subband matrix, reverse-ordering of the subbands along the rows and columns can be implemented in the subband images simply by modifying the memory access pattern of the computing block, without any computational overhead. Such simple modifications in subband orientation can highly affect image perception and can be implemented without any computational overheads. It can be used as a parameter for the SWT operation. A prior knowledge of these parameters is a must in order to decompress the image correctly. There are several useful features of subband re-orientation that make it suitable for being a part of the SWT:

3.1.2.1 Zero computational overhead

Subband re-orientation can be achieved by intelligently writing the outputs of DWT filter to the memory without any overheads in computational costs of the system.

3.1.2.2 Feasible subband re-orientations

In Figure 3.3(b), we illustrate how we can represent the same subband in eight different orientations: we have four orientations of the subband decided by the forward or reverse ordering of the matrix along
Figure 3.5  Image reconstruction with different keys. A- Aerial map image, B- San Francisco Golden gate aerial image, C- Brick wall (texture) image and D- Airplane image. (i)- Original image encrypted with key0, (ii)- Image decrypted with same key, (iii)-(vi)- Image decrypted with randomly generated keys.
Figure 3.6  Image reconstruction with different keys. A- Aerial map image, B- San Francisco Golden gate aerial image, C- Brick wall (texture) image and D- Airplane image. (i)- Original image encrypted with key0, (ii)- Image decrypted with same key, (iii)-(vi) Image decrypted with hamming distance of 1,4, 6 and 8
Figure 3.7 Image reconstruction with randomly generated keys. (a)-(d) give result of 1000 random trials on the four sample images respectively. The x-axis gives results with different keys. The 500th trial (with 500th key) refers to the test case with decryption with same key as the encryption key. The y-axis represents the PSNR value of the reconstructed images.

rows or columns. We get four more orientations by transposing the above four, summing up to eight possible transformations for each subband. We need a 3 bit value to represent this transformation for a single subband.

3.1.2.3 Key-space

Figure 3.3(a) shows the nineteen different subbands obtained by a 6 level wavelet decomposition. In general, we obtain $3N + 1$ subbands for a $N$ level wavelet decomposition, each requiring 3 bits. Thus, we get a keyspace of $9N + 3$ bits using subband re-orientation.
Table 3.1  PSNR values (in db) for image reconstruction with various random keys
(encoded with key0)

<table>
<thead>
<tr>
<th></th>
<th>Key0</th>
<th>Key1</th>
<th>Key2</th>
<th>Key3</th>
<th>Key4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aerial</td>
<td>∞</td>
<td>12.36</td>
<td>11.17</td>
<td>11.67</td>
<td>11.77</td>
</tr>
<tr>
<td>San Francisco</td>
<td>∞</td>
<td>18.40</td>
<td>17.34</td>
<td>18.21</td>
<td>18.46</td>
</tr>
<tr>
<td>Brick Wall</td>
<td>∞</td>
<td>14.75</td>
<td>13.39</td>
<td>14.34</td>
<td>13.58</td>
</tr>
<tr>
<td>Airplane</td>
<td>∞</td>
<td>13.19</td>
<td>11.26</td>
<td>11.63</td>
<td>12.43</td>
</tr>
</tbody>
</table>

3.2 Security

In this section a brief evaluation of the security features of proposed scheme is presented. A key-space of $16mN + (9N + 3)$ bits can be obtained from $N$ levels of wavelet decomposition. For an image size of $512 \times 512$ pixels this upper limit of $N (N_{\text{max}})$ is 9. However, choosing $N$ close to $N_{\text{max}}$ will lead to the innermost subband size being very small.

We selected wavelet decomposition level of $N = 6$ for images of dimension $512 \times 512$ pixels to allow a standard block size of $8 \times 8$ pixels for the innermost subbands. $m = 8$ is set for applications sensitive to image quality while $m = 5$ works for all general applications.

Shannon’s 1949 paper [94], which serves as the foundational treatment of modern cryptography calls this property as the ‘confusion’ property. Ideally, change in one bit of the key should change the cipher text completely.

Figure 3.5 gives the performance of our scheme against attacks with random keys. The images decrypted with wrong keys have little resemblance to original images as indicated by the PSNR values for these reconstructed images (as shown in Table 3.1). Figure 3.7(a-d) gives the plot of PSNR values of reconstructed images for the four test images. 1000 such trials were run with different random keys. The single peak in each graph is observed for the 500th trial where the original key (for encryption) and the decryption key are the same.

The hamming distance ($h.d.$) between two strings of equal length is the number of positions for which the corresponding symbols are different. i.e. the minimum number of bits that must be “flipped” to go from one word to the other. An ideal encryption scheme must give entirely random output if the $h.d.$ between the encryption and decryption keys is non-zero. That is the case with block ciphers.
Table 3.2 Variations in image reconstruction quality (PSNR values) with ham- 
mimg distance

<table>
<thead>
<tr>
<th>Hamming distance</th>
<th>0</th>
<th>1</th>
<th>4</th>
<th>6</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aerial</td>
<td>∞</td>
<td>50.3</td>
<td>23</td>
<td>16.04</td>
<td>13.18</td>
</tr>
<tr>
<td>San Francisco</td>
<td>∞</td>
<td>36.27</td>
<td>30.98</td>
<td>22.61</td>
<td>21.09</td>
</tr>
<tr>
<td>Brick Wall</td>
<td>∞</td>
<td>50.27</td>
<td>37.5895</td>
<td>25.9</td>
<td>23.2</td>
</tr>
<tr>
<td>Airplane</td>
<td>∞</td>
<td>44.28</td>
<td>21.64</td>
<td>21.43</td>
<td>16.16</td>
</tr>
</tbody>
</table>

such as AES or DES which allow enough mixing between bit values in multiple rounds to achieve that effect. The performance of SWT, is thus going to be less than the conventional cryptographic schemes.

We tested our scheme for image reconstruction performance with small $h.d.$ between the two keys. Our scheme provides security as evident by the low PSNR values (for $h.d. \geq 4$) in Table 3.2. 1000 simulations were run to obtain the average PSNR value of reconstructed image with different hamming distances between the encoder and decoder key. It can be observed from the PSNR values that a hamming distance of 6 and above gives a perceptible reduction in image appearance (indicated by low PSNR value). The visual results are shown in Fig. 3.6. Different bit positions in the key have different effect on the image quality degradation. This is attributed to the fact that changing different bit positions in value of $\alpha$ will lead to different degrees of distortions. This attributes to the fact that Figure 3.6 (D)(vi) has less quality degradation compared to Figure 3.6 (D)(v). To quantify the image degradation with increasing $h.d.$, we ran 1000 simulations and recorded the average values in Table 3.2

3.3 Hardware Implementation

Figure 3.8 gives an overview of the 1-D SWT hardware architecture. The input data (one pixel input per cycle) $x$ is pipelined for eight cycles. We observe that $z^i$ and $z^{-i}$ values in expressions for $H_1(z)$ and $H_2(z)$ have the same coefficients. Thus, these values can be added together to simplify further computations. In Figure 3.8, eight of the nine inputs are passed through four adders to reduce the number of input to five. These values (labeled $w_0, w_1, w_2, w_3$ and $w_4$) are multiplied with $\alpha, \alpha^2$ and $\alpha^{-1}$ to obtain the necessary intermediate values which are input to shift and add logic. The high and low pass filter coefficients are the final output of the 1-D SWT filter.
Figure 3.8  Hardware Architecture for the 1-D SWT Filter
Table 3.3 Hardware Utilization of DWT architecture on Xilinx Virtex XCVLX330 FPGA

<table>
<thead>
<tr>
<th></th>
<th>SWT (a)</th>
<th>SWT (b)</th>
<th>SWT (c)</th>
<th>[82]</th>
<th>[70]</th>
<th>Daub. 9/7</th>
<th>[43]</th>
<th>[110]</th>
<th>[40]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slices FlipFlop</td>
<td>5580</td>
<td>-</td>
<td>649</td>
<td>245</td>
<td>-</td>
<td>210</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Multipliers</td>
<td>0</td>
<td>13</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>12</td>
<td>36</td>
<td>12</td>
</tr>
<tr>
<td>Adders</td>
<td>11</td>
<td>41</td>
<td>41</td>
<td>9</td>
<td>19</td>
<td>15</td>
<td>16</td>
<td>36</td>
<td>16</td>
</tr>
<tr>
<td>Registers</td>
<td>120</td>
<td>-</td>
<td>92</td>
<td>208</td>
<td>-</td>
<td>144</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Critical Path</td>
<td>4(T_a + T_l)</td>
<td>T_{m}+5T_a</td>
<td>T_{m}+5T_a</td>
<td>3T_a</td>
<td>5T_a</td>
<td>T_{m}+4T_a</td>
<td>T_{m}+2T_a</td>
<td>T_{m}+4T_a</td>
<td>4T_{m}+8T_a</td>
</tr>
<tr>
<td>Clock Frequency (MHz)</td>
<td>114(np)</td>
<td>-</td>
<td>60(np)</td>
<td>243(p)</td>
<td>120(np)</td>
<td>200</td>
<td>107</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Security</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Note: (a) Design with Reconfigurable Constant Multipliers mapped to FPGA, (b) Design mapped to 45 nm VLSI technology, and (c) Design mapped to FPGA directly.

\(T_m\) and \(T_a\) are the time delay in multiplier and adder circuits respectively.

\(np\): not pipelined  \(p\): pipelined
We performed several optimization steps to reduce the cost of the underlying hardware. Division by binary coefficients (e.g. $1/64, 1/16, 1/4$) was performed using arithmetic shift operations. This reduces the number of multipliers in the circuit from 69 to 23. Reducing the number of inputs from nine to five reduces the number of adders in the design from 70 to 41 and the number of multipliers from 23 to 13. The input stream was then pipelined to achieve a higher clock frequency (and hence higher throughput).

A Xilinx XC5VLX330 FPGA was targeted for our experiments, using ModelSim 6.4 and Xilinx ISE 10.1 for simulations and synthesis. The non-pipelined design had clock frequency of 60 MHz while a pipelined design with four extra cycles of latency achieves a clock frequency of 242 MHz. The design was also implemented using Synopsys Design Compiler with the freePDK [97] 45nm cell library. Under the timing constraints of 500 MHz, the design required 4259 cells and a chip area of 22066 $\mu m^2$.

The design used 13 10x9 bit multipliers, 41 adders (20 18-bit adders and 21 9-bit adders). The hardware requirements of our implementation are summarized and compared with other implementations in Table 3.3. The critical path of the implementation is $T_m + 5T_a$ where $T_m$ indicates the time delay in multiplier and $T_a$ indicates the time delay in adder circuit.

The subband re-orientation part in DWT is done by changing the write pattern of the subbands after the SWT operation. Thus, no computational overhead is involved in such an operation. It is noteworthy that ours is the first proposal for image and video security based on SWT and its hardware implementation.

The initial parameterized DWT design obtained a clock frequency of about around 60 MHz, due to its long critical path. The critical path of the circuit lies from the $w_i$ registers to the final low pass output. We then pipelined this computation into several stages and obtained a faster implementation. By adding 4 pipelining stages we obtained a clock frequency of 242 MHz.

### 3.3.1 Reconfigurable Constant Multiplier (RCM)

The expression for low and high pass filter coefficients were obtained in section 2.1. It was implemented in Figure 3.8 using several multiplier units. The $w_i, i \in \{0, 1, 2, 3, 4\}$ values are obtained
65

by summing the inputs for symmetric taps in the SWT implementation as shown in Figure 3.8. \( w_i \) is calculated as follows:

\[
w_i(k) = x(k + i) + x(k - i), \quad i \in \{0, 1, 2, 3, 4\}
\]

Then, we can represent the filter expressions as:

\[
H_1(k) = \sum_{i=0}^{4} K_i(a) \times w_i(k)
\]

and

\[
H_0(k) = \sum_{i=0}^{3} \hat{K}_i(a) \times w_i(k)
\]

Here \( K_i(a) \) and \( \hat{K}_i(a) \) are the functions of the variable \( a \), and \( w_i \) are obtained from the pipelined input. The values of functions \( K_i(a) \) and \( \hat{K}_i(a) \) remains the same as long as we have the same \( a \) parameter. This implies that this value of these functions behave as a constant and changes only when we change the encryption key (and the associated parameter \( a \)). This value can thus be computed and hard-coded into the circuit. This constant multiplication can easily be mapped to a reconfigurable hardware with programmable LUTs. If the input is represented by \( B_1 \) bits and constant is represented by \( B_2 \) bits. We can use \((B_1 + B_2) \) \( B_2 \)-input LUTs to get the output values \( H_1(k) \) and \( H_2(k) \). Alternatively we can break down a \((B_1 \times B_2) \) bit multiplication into smaller input LUTs. Thus the LUTs based multiplier can be reconfigured corresponding to incorporate any changes in encryption key.

This idea is used to build a Reconfigurable Constant Multiplier or RCM. A RCM has one operand which is kept constant and mapped to LUTs while the other multiplicand is a variable and changes its value every clock cycle. The constant operand can be changed dynamically by reconfiguring the LUT values on-the-fly.

We discuss the implementation of a 4x4 bit RCM to explain the LUT mappings.

### 3.3.1.1 4 \times 4 Bits Multiplier using LUTs

Let A and B be the two operands, both being 4 bits long. Let us define two new binary variables:

\[
P_i = A_i \oplus B_i, \quad G_i = A_i B_i
\]
The output bit and the sum at each stage can be represented as:

\[ S_i = P_i \oplus C_i C_{i+1} = G_i + P_i C_i \]

On simplification [64], we get

\[ C_1 = \text{initial carry} \]
\[ C_2 = G_1 + P_1 C_1 = A_1 B_1 + (A_1 \oplus B_1) A_0 B_0 \]
\[ C_3 = G_2 + P_2 G_1 + P_2 P_1 C_1 \]
\[ C_4 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_1 \]

and

\[ S_1 = A_1 \oplus B_1 \oplus C_1 = A_1 \oplus B_1 \oplus A_0 B_0 \]
\[ S_2 = A_2 \oplus B_2 \oplus C_2 = A_2 \oplus B_2 \oplus (A_1 B_1 + (A_1 \oplus B_1) A_0 B_0) \ldots \]

We make some interesting observations and inferences.

- \( C_i \) values can be expanded and expressed in terms of \( A_i \) and \( B_i \) values.

- Similarly, a complex logical expression can be generated for each \( S_i \) value. Each \( S_i \) value is characterized uniquely by a logical expression.

- If one of the inputs (say \( B \)) is a constant, \( S_i \) can be represented as a logic function of bit values of \( A \).

\[ S_i = f_i(A_3, A_2, A_1, A_0) \]

- No matter, how complex the function \( f_i() \) may be, the truth table can be represented by a 4-LUT. Essentially, all the complex expressions for \( S_i \) can be expressed in terms of truth table of 4-LUT.

- We can represent the eight output bits for \( 4 \times 4 \) bits multiplier with eight 4-LUTs.

In general, we can implement a \( M \times K \) bit constant multiplication using \( (M+K) \) K-input LUTs.
It has been discovered that the LUT size of 4 to 6 provides the best area-delay product for an FPGA [3]. Most commercial reconfigurable devices such as FPGAs have 4-input LUTs. We therefore discuss the mapping of an $M \times K$ bit constant multiplier into 4-LUTs in the next subsection.

3.3.1.2 Mapping a generic RCM into LUTs

The multiplication of two inputs $A$ and $B$ ($M$-bit variable input $A$, $K$-bit reconfigurable constant $B$) can be mapped to LUTs similar to $4 \times 4$ bits multiplier by obtaining a generic expression for $S_1, S_2, \ldots, S_{M+K-1}$. $S_i$ values can be represented as $f(A_{M-1}, A_{M-2}, \ldots A_1)$ and can be therefore mapped into an M-input LUT. We have $(M + K - 1)$ $S_i$ values, requiring $(M + K - 1)$ M-input LUTs to implement the multiplication of $A$ and $B$.

A $(K+1)$-input LUT can be build from 2 $K$-input LUTs (as shown in the Figure 3.9). For example, we can build a 8-LUT from 2 7-LUTs which can be synthesized from $2 \times 2 = 4$ 6-LUTs. Thus, one 8-LUT can be made from $2^4 = 16$ 4-LUTs. Thus, we can build an arbitrary M-LUT from $2^{M-4}$ 4-LUTs.

Figure 3.10 gives an example of multiplication of 8-bit number with 12-bit constant (M=8, K=12). Figure 3.10(a) gives an implementation using 8-LUTs. 20 8-LUTs or equivalently 128 4-LUTs are used in the design.

Figure 3.10(b) gives an alternative implementation of the same multiplication by breaking the input number into multiples of 4-bit values. 4-input LUTs are used to obtain the X and Y values which are then added together using an adder. This implementation requires 32 4-LUTs and a 20 bits adder. This design requires less LUTs but the presence of 20-bit adder may slow down the clock speed in such a design.
3.3.2 Implementation Results

We estimated the hardware performance of our architecture by synthesizing the design on a Xilinx Virtex-4 XC4VLX140 FPGA, using ModelSim SE 6.4 for simulation and Xilinx ISE 10.1 for synthesis. This design just serves as the proof of concept for our architecture. An ASIC implementation with fixed interconnects for LUTs can achieve significant improvements in clock speed and throughputs.

Table 3.3 shows the performance comparison of SWT architecture with existing works and amongst different various configurations. A direct implementation of Daubechies 9/7 DWT filter requires 16 multiplier and 12 adders in the design. The critical path is $T_m + 4T_a$, where $T_m$ is the time latency of multiplier and $T_a$ is the time latency of adder. Several optimizations were proposed including those by [43, 110, 40]. Our earlier work [82] obtains a multiplier-less optimized architecture for DWT that has time latency of only $3T_a$ cycles. On a Virtex-4 FPGA, it obtained a clock frequency of 120 MHz.

A direct implementation of SWT using hardware multipliers gave a clock frequency of 60 MHz. The critical path has one multiplier and five adders ($T_m + 5T_a$). We removed all the multipliers in the design with RCM blocks which reduced the critical path to four adders and one look-up operation.
(4T_a + T_i). (The entire expression for filter coefficients, earlier spanning many multipliers and adders is now represented by a single RCM). The use of reconfigurable multipliers reduces the critical path of the SWT circuit and leads to an improved clock frequency of 114 MHz.

All the reported clock frequency except the VLSI implementation represent implementation on Vitex-4 FPGA. These FPGAs are built on a 90nm process technology. To test the speed of VLSI implementation of proposed architecture, we used freepdk 45 nm cell library [97]. We were able to target a clock frequency of 500 MHz.

It can easily support HD video at 30 frames per second and resolutions higher than 1440 × 1080.

3.4 Conclusion and Future Work

We proposed a DWT design in which the choice of filter coefficients and the orientation of subbands are performed in accordance with a key. The system provides both encryption and security and thwarts brute force attacks. The major contributions of this work are as follows:

1. DWT kernel was parameterized to incorporate the encryption feature and promise reasonable security for real-time embedded multimedia systems.

2. A zero computation overhead subband re-orientation scheme is proposed and implemented in this paper.

3. An optimized hardware implementation of the DWT architecture is presented. The proposed hardware implementation has low critical path and thus achieves a high clock frequency. Reconfigurable hardware based implementation is presented in this paper to embed the key information into the reconfigurable bit stream.

The proposed SWT operation provides a large key-space for multimedia encryption when used as a part of image compression system. As a future work, we propose to parameterize and integrate encryption to other steps in multimedia compression. However, if used by itself, it is prone to cryptanalysis because it retains correlation amongst subbands and some other properties useful in subsequent compression operations.
CHAPTER 4. CHAOTIC FILTER BANKS

Chaotic filter bank schemes have been proposed in the research literature to allow for the efficient encryption of data for real-time embedded systems. Some security flaws have been found in the underlying approaches which makes such a scheme unsafe for application in real life scenarios. In this paper, we first present an improved scheme to alleviate the weaknesses of the chaotic filter bank scheme, and add enhanced security features, to form a Modified Chaotic Filter Bank (MCFB) scheme. Next, we present a reconfigurable hardware implementation of the MCFB scheme. Implementation on reconfigurable hardware speeds up the performance of MCFB scheme by mapping some of the multipliers in design to reconfigurable Look-Up Tables, while removing many unnecessary multipliers. An optimized implementation on Xilinx Virtex-5 XC5VLX330 FPGA gave a speedup of 30% over non-optimized direct implementation. A clock frequency of 88 MHz was obtained.

4.1 Introduction

4.1.1 Chaos and Cryptography

Chaos theory plays an active role in modern cryptography. As the basis for developing a crypto-system, the advantage of using chaos lies in its random behavior and sensitivity to initial conditions and parameter settings to fulfill the classical Shannon requirements of confusion and diffusion [94]. A tiny difference in the starting state and parameter setting of these systems can lead to completely different outputs over a few iterations. Thus, sensitivity to initial conditions manifests itself as an exponential growth of error and the behavior of system appears chaotic.

Quite a bit of research has been devoted to the study of continuous-time chaotic systems such as the oscillator circuits [19, 58, 89]. However, these schemes need a synchronization procedure. On the other hand, discrete-time chaotic systems behave like private-key encryption algorithms [90] and are


amenable to implementation on fixed point hardware.

Many chaotic block ciphers [8, 47, 37, 120, 84] have been proposed in research literature. For example, Baptista [8] builds a block cipher based on chaotic encryption. Each character of the message is encoded as the integer number of iterations performed in the logistic equation, in order to transfer the trajectory from an initial condition towards a pre-defined interval inside the logistic chaotic attractor.

Some limitations of such block ciphers and the logistic chaotic attractor are explained as follows:

Firstly, the distribution of the ciphertext is not flat enough to ensure high security since the occurrence probability of cipher blocks decays exponentially as the number of iterations increases. Secondly, the encryption speed of these cryptographic schemes is very slow since at least 250 iterations of the chaotic map are required for encrypting an 8-bit symbol. The number of iterations may vary up to 65532. Thirdly, the length of ciphertext is at least twice that of plaintext, $X$ bits of message may result in several tens of thousands of iterations that need $2X$ bytes to carry. Despite the improvements proposed by subsequent research, block ciphers based on Baptista’s work remain slow to satisfy the encryption needs of the real-time data encryption systems.

A stream cipher was designed over chaotic maps and presented in early 1991 by [38]. Its cryptanalysis was presented in the same conference [13]. Chen et al. [37, 120] constructed a block cipher based on three-dimensional maps while [84] proposed a cipher by direct discretization of two dimensional Baker map. A good survey and introductory tutorial on these schemes is found in [118, 46]. The authors in [71] present a crypto-system based on a discretization of the skew tent map. [72] presents chaotic Feistel and chaotic uniform operations for block ciphers. Although various schemes/maps have been proposed in the research literature, the logistic map remains one of the simplest maps and is used in many schemes.

4.1.2 Wavelets and Chaotic Filter Banks

Chaotic Filter Banks based cipher was proposed in 2007 by Ling et al. [59]. It allows great flexibility in the design and gives the following advantages:

1. One can embed signals in different frequency bands by employing different chaotic functions.

2. The number of chaotic generators to be employed and their corresponding functions can be se-
lected and designed in a flexible manner because perfect reconstruction does not depend on the invertibility, causality, linearity and time invariance of the corresponding chaotic functions.

3. The ratios of the subband signal powers to the chaotic subband signal powers can be easily changed by the designers and perfect reconstruction is still guaranteed no matter how small these ratios are.

4. The proposed cryptographic system can be easily adapted to the international multimedia standards, such as JPEG 2000 and MPEG4[59].

The encryption procedure is carried out by decomposing the input plaintext signal into two different subbands and masking each of them with a pseudo-random number sequence generated by iterating the chaotic logistic map. The authors [59] use the Discrete Wavelet Transform (DWT) based filter banks in their approach to maintain compatibility with existing image compression standards such as JPEG2000 [24].

[6] presents a cryptanalysis of [59] which exposes weaknesses of chaotic filter bank against known plain-text attacks and also exposes the limitation of reduction of key space by use of logistic map.

4.1.3 Scope and Organization

In this paper we present the design and implementation of a chaotic stream cipher that uses less hardware, has promising security and has high throughput to serve the requirements of real-time embedded systems. The main contributions of this paper can be summarized as follows:

1. The proposed Modified Chaotic Filter Bank (MCFB) scheme is a lightweight cipher designed to satisfy the resource requirements of real-time embedded systems, security requirements of modern communication systems and format-compliance with existing multimedia compression standards such as JPEG2000, MPEG-4, etc.

2. To the best of knowledge of the authors, this is the first hardware implementation of a chaotic filter bank scheme in hardware.

3. A clock frequency of 88 MHz was obtained for a Virtex-5 XC5VLX330 FPGA. The design was synthesized and implemented using Xilinx ISE 10.1 tool.
The paper is organized as follows: Section 4.2 gives a brief overview of the wavelet transform. Section 4.3 gives details of the chaotic filter bank scheme proposed earlier. In Section 4.4, we discuss the MCFB Scheme and subsequently discuss its distinguishing features in Section 4.5 and 4.6. Section 4.5 explains the Improved Chaotic Oscillator and Section 4.6 gives an overview of wavelet parameterization. Section 4.7 gives the details of hardware implementation over Xilinx Virtex-5 FPGA and the proposed optimizations, while section 4.8 concludes the paper with directions of future work.

4.2 Wavelets

The efficient representation of time-frequency information by the wavelet transform has led to its popularity for signal processing applications. It provides superior rate-distortion and subjective image quality performance over existing standards. Applying a 2-D DWT to an image of resolution \( M \times N \) results in four images of dimensions \( \frac{M}{2} \times \frac{N}{2} \): three are detailed images along the horizontal (LH), vertical (HL) and diagonal (HH), and one is coarse approximation (LL) of the original image. LL represents the low frequency component of the image, while LH, HL, and HH represent the high frequency components. This LL image can be further decomposed by DWT operation. Three levels of such transforms are applied and shown in Figure 2.4. The coarse information is preserved in the LL3 image and this operation forms the basis of Multi-Resolution Analysis for DWT [107].

Prior works in signal processing explain that the 1-D DWT can be viewed as a signal decomposition using specific low pass and high pass filters. A single stage of image decomposition can be implemented by successive horizontal row and vertical column wavelet transforms. Thus one level of DWT operation is represented by filtering with high and low pass filters across row and column successively and is explained in Figure 2.3. After each filtering a down sampling is done by a factor of 2 to remove the redundant information.

4.2.1 Commonly Used DWT Filters

The two most common DWT filters used in image compression are Le Gall’s 5/3 filter and the Daubechies 9/7 filter [24]. They are accepted in the JPEG2000 standards. The Le Gall’s filter has rational coefficients and its hardware implementation requires less resources. The Daubechies 9/7 (also
commonly known as CDF 9/7) filter has better compression performance. However, it has irrational coefficients therefore its hardware requirements are very large.

The details have been discussed in details in chapter 2.

4.2.2 Reconfigurable Hardware Implementation

Much research has been done in the development of DWT architectures for image processing [11, 12, 88, 49, 70]. A good survey on architectures on DWT coding is given by Tseng et al. [104].

Recent works in partial reconfiguration of FPGAs implement DWT in a reconfigurable fashion. [26] gives a comparison of embedded reconfigurable video-processing architectures. They propose a hybrid of two hardware platforms: one providing easy reconfiguration of modules and the other providing easy implementation with higher clock frequency, to achieve an optimal FPGA-based dynamically and partially reconfigurable platform for real-time video and image processing. The tool ReCoBus-Builder [48] simplifies the generation of dynamically reconfigurable systems to almost a push button process. The work also describes a communication infrastructure for dynamically reconfigurable systems.

4.3 Chaotic Filter Bank Scheme

The chaotic filter bank scheme is illustrated in Figure 4.1. A chaotic function $\alpha_i()$ is used to create chaotic response to the system.

$$\alpha_i(n) = n + s_i(n), \ i \in \{1, 2\}$$

where $s_i(n)$ is the output of chaotic map.

The various signals in Figure 4.1 are expressed as follows:
Figure 4.1 Block Diagram representation of the Chaotic Filter Bank Scheme. (a) The encryption module and (b) The decryption module

\[ y_0[n] = \sum_{\forall m} x[m]h_0[2n - m], \]

\[ y_1[n] = \sum_{\forall m} x[m]h_1[2n - m], \]

\[ z_0[n] = y_0[n] + \alpha_0(y_1[n]), \]

and \[ z_1[n] = y_1[n] - \alpha_1(y_0[n]), \]

\[ \Rightarrow z_0[n] = y_0[n] + y_1[n] + s_0[n], \]

and \[ z_1[n] = y_1[n] + y_0[n] - s_1[n] \]

The reconstructed signal \( x'[n] \) must be the same as the original signal \( x[n] \). At the decoder, first the effect of mixing with chaotic signals is reversed and then corresponding inverse wavelet transform is applied.
\[ y'_1[n] = z_1[n] + \alpha_1(z_0[n]), \]
\[ y'_0[n] = z_0[n] - \alpha_0(z_1[n]), \]
\[ x'[n] = \sum_{\forall m} y'_0[m]g_0[n - 2m] + \sum_{\forall m} y'_1[m]g_1[n - 2m] \]

where \( h_0, h_1 \) are so-called analysis and \( g_0, g_1 \) are synthesis filters. Choosing Le Gall’s 5/3 filter or Daubechies 9/7 filters allow correct recovery of the plain text signal.

### 4.3.1 Chaotic Maps

As explained above, the chaotic filter bank scheme uses two chaotic maps \( \alpha_0() \) and \( \alpha_1() \) for its operation. These chaotic maps are based on the logistic map.

The logistic map is a polynomial mapping of degree 2. It demonstrates chaotic behavior although using a simple non-linear dynamical equation. Mathematically, the logistic map is written as:

\[ x_{n+1} = \lambda_{LM} \times x_n(1 - x_n) \]

where \( \lambda_{LM} \) is a positive number.

The behavior of logistic map is dependent on the value of \( \lambda_{LM} \). At \( \lambda_{LM} \approx 3.57 \) is the onset of chaos, at the end of the period-doubling cascade. We can no longer see any oscillations. Slight variations in the initial population yield dramatically different results over time, a prime characteristic of chaos. Most values beyond 3.57 exhibit a chaotic behavior, but certain isolated values of \( \lambda_{LM} \) appear to show non-chaotic behavior and are called as islands of stability. Beyond \( \lambda_{LM} = 4 \), the values eventually leave the interval \([0, 1]\) and diverge for almost all initial values.

A rough description of chaos is that chaotic systems exhibit a great sensitivity to initial conditions – a property of the logistic map for most values of \( \lambda \) between about 3.57 and 4. This stretching-and-folding does not just produce a gradual divergence of the sequences of iterates, but an exponential divergence, evidenced also by the complexity and unpredictability of the chaotic logistic map.
4.3.2 Key Space

The authors in [59] suggest using the initial values of logistic map and the value of parameter $\lambda_{LM}$ to build the key space.

[6] present a cryptanalysis of the above mentioned scheme and exposes some weaknesses of the scheme. They are enumerated as follows:

1. **Reduction of the key-space** [59] propose to use the entire range $[3, 4]$ as the key space. The values of $\lambda_{LM}$ in the interval $[3, 3.57]$ does not produce any chaos. Besides this, there are many points (known as islands as islands of singularity) in the interval $[3.57, 4]$ where iteration on logistic map leads to oscillation among finite values (see Figure 4.2(d)). Another issue is the non-uniform distribution of output values (as shown in Figure 4.2(a-b)).

2. **Vulnerability to known plain-text attack** The value of $\lambda_{LM}$ can be calculated very accurately from two successive iterations of the logistic map leading to successful plain text attacks on the scheme.

4.4 The MCFB Scheme

The MCFB Scheme makes three modifications to the original scheme, making it more secure and also improving its frequency resolution.

1. The Chaotic Filter Bank scheme [59] involves mixing of low pass and high pass coefficients. This mixing hampers the compression performance of the Wavelet Transform. The equations for $z_0[n]$ and $z_1[n]$ have $y_1[n]$, and $y_0[n]$ terms in expressions for $z_0[n]$ and $z_1[n]$ respectively which lead to loss of frequency resolution of Discrete Wavelet Transform.

The new relationship between $z_0[n]$ and $z_1[n]$ is given by the following equations:

$$z_0[n] = y_0[n] + s_0[n],$$

and

$$z_1[n] = y_1[n] + s_1[n]$$
Figure 4.2 Histogram for 50000 samples obtained using Logistic map with initial seed 0.100010 and (a) $\lambda_{LM} = 3.61$ and (b) $\lambda_{LM} = 3.91$ (c) $\lambda_{LM} = 4$ and (d) $\lambda_{LM} = 3.83$

2. We use an Improved Chaotic Oscillator (ICO) instead of the standard logistic map. This chaotic oscillator, although derived from the standard logistic map, is strong against known cryptanalysis of Logistic Map-based ciphers and chaotic filter banks. Moreover, it has a large continuous key space as against logistic map which has very limited key space with regions of stability within the same range.

3. We replace the DWT filter banks with a parameterized filter bank that yields has the same properties as the original filters but allows us to choose from a very large number of possible filters while implementing a filter bank.

The choice of filter bank and parameters for the chaotic oscillators used in the design is governed by a key. The overall system is shown in figure 4.3.
The improved chaotic oscillator and parameterized wavelet transform are explained in following two sections.

4.5 Improved Chaotic Oscillator

In this subsection, we give a brief description of an improved chaotic oscillator, based on a modified logistic map, that alleviates the problems associated with chaotic generator proposed in [59]. The proposed scheme is robust to the choice of initial conditions (due to lack of any unsuitable $\lambda$ values), achieves real-time encryption speed and resistant to known attacks.
4.5.1 The Modified Logistic Map (MLM)

Our initial experimentation involved generation of pseudo-random number sequences by varying the parameter $\lambda_{LM}$ in the range $[3.57, 4]$. It led to several observations:

1. The histogram obtained for different $\lambda_{LM}$ values (with 50000 samples) is skewed and not uniform or flat. This is illustrated for $\lambda_{LM} = 3.61$ and $\lambda_{LM} = 3.91$ values in figure 4.2(a-b). The distribution for $\lambda_{LM} = 4$ is most flat and symmetric (see figure 4.2(c)). It is desirable to have a flatter distribution of samples drawn from the logistic map in order to increase its randomness.

2. For $\lambda_{LM} = 4$, the logistic map equation $x_{n+1} = \lambda_{LM} \times x_n(1 - x_n)$ has the same domain and range intervals $(0, 1)$. For $\lambda_{LM} < 4$ and input $x_n$ in range $(0, 1)$, the range of $x_{n+1}$ in the expression is $(0, \lambda_{LM}/4]$ and the distribution of random numbers is biased towards 0 or 1 (as seen in distributions in figure 4.2(a-b)). It is desirable to have a distribution of random numbers symmetric around 0.5.

3. There are certain isolated values of $\lambda_{LM}$ that appear to show non-chaotic behavior and are called as islands of stability. For example: $\lambda_{LM} = 1 + \sqrt(8) \approx 3.83$ show oscillation between three values.

4. $\lambda_{LM} = 4.0$ has most flat, uniform and symmetric histogram than other $\lambda_{LM}$ values.

We address these issues by developing a MLM, defined by the following equation:

$$x_{n+1} = \lambda \times x_n(1 - x_n) + \mu$$

where the $x_n$ values are restricted to the interval $[\alpha, 1 - \alpha]$, $\alpha < 0.5$. The maxima of this function occurs at $x_n = 0.5$ and the maximum value is $\lambda/4 + \mu$ while the minimum (in specified domain) occurs at $x_n = \alpha$ or $x_n = 1 - \alpha$ and the minimum value is $\lambda \times \alpha(1 - \alpha) + \mu$. Equating the maximum and minimum values to the range $[\alpha, (1 - \alpha)]$ leads to the following equations:

$$\alpha = \lambda\alpha(1 - \alpha) + \mu$$
Figure 4.4  Histogram for 50000 samples obtained using Modified Logistic map with $\alpha$ values corresponding to (a) $\lambda_{LM} = 3.61$ and (b) $\lambda_{LM} = 3.91$

$$1 - \alpha = \frac{\lambda}{4} + \mu$$

On solving these equations, we get $\lambda = \frac{4}{1-2\alpha}$ and $\mu = \frac{\alpha(2\alpha-3)}{1-2\alpha}$. Substituting these values, we get a flatter histogram for the new logistic map as evident in Figure 4.4. This modified logistic map addresses the requirements of flatter and symmetric distribution and also avoids islands of stability by generating a flat distribution for all values of $\alpha$.

The output of the modified logistic map ($x_n$) is quantized to get a 16 bit value $p_n$. $x_n, 0 < x_n < 1$ is represented in fixed point as follows:

$$x_n = \sum_{j=0}^{N-1} \{a_j\} \times 2^{j-N}$$

where $a_j$ are individual bit values.

Thus, $p_n$ is given by:

$$p_n = \sum_{j=0}^{15} \{a_j\} \times 2^{j-N}$$

The quantization step or truncation of more significant bits is non-linear in nature (it is a many-one mathematical function), thereby increasing the complexity of any attacks that try to recover the logistic map information from the cipher text using any cryptanalysis.
We generate another pseudo-random sequence $s_n$ from the given sequence $p_n$ by the following operation:

$$s_n = p_n \oplus p_{n-1} \oplus p_{n-2}$$

There is no linear correlation between the two sequences $p_n$ and $s_n$. Statistical de-correlation makes it difficult to back-track $p_n$ from $s_n$.

### 4.6 Wavelet Parameterization

We now present a new layout and configuration scheme for the parameterized DWT. A new parameterized construction of the DWT filter with rational coefficients has dual advantages. The parameterized construction can be used to build a key scheme while the rational coefficients of the DWT enable an efficient hardware architecture using fixed point arithmetic (as shown in previous chapter). We get the following expression for $H_1(z)$ and $H_2(z)$.

$$H_1(z) = (-9/64a + 1/32a^2 + 15/64 - 1/8/a)$$

$$(z^4 + 1/z^4)$$

$$+(-1/16a^2 + 11/32a - 11/16 + 1/2/a)$$

$$(z^3 + 1/z^3)$$

$$+(1/8 - 1/2/a)(z^2 + 1/z^2)$$

$$+(-11/32a + 1/16a^2 + 15/16 - 1/2/a)$$

$$(z + 1/z)$$

$$+(9/32a - 1/16a^2 - 7/32 + 5/4/a)$$

$$H_2(z) = (1/32 - 1/32a)(z^3 + 1/z^3)$$

$$(z^3 + 1/z^3)$$

$$+(1/8 - 1/16a)(z^2 + 1/z^2) +$$

$$+(7/32 + 1/32a)(z + 1/z) + (1/4 + 1/8a)$$
We get different DWT filters simply by changing the $a$ values. The choice of the $a$ value is secretly determined using a secret key. The numerical value of free parameter $a$ can be varied over a wide range while retaining the perfect reconstruction property of the wavelet transform. However, as we vary the value of $a$ over the range $(-\infty, +\infty)$, the output values of the DWT operation have a very large dynamic range requiring a larger number of bits for representation. This would reduce the compression rates achievable with the DWT-based coders. Numerical experiments show that parameterized DWT has a good PSNR value for image reconstruction with Set-Partitioning in Hierarchical Trees (SPIHT) based coder when $a$ varies in the range 1 to 3. When $a$ varies beyond this range, the output DWT coefficients are spread over a large dynamic range. At low bit rates, the encoder is not able to efficiently encode such a large range of input coefficients leading to poor compression results for natural images.

4.7 Resistance of Chaotic Generator against Cryptanalysis

The performance and accuracy of discrete chaotic ciphers is a translation of properties of the underlying dynamical system (or chaotic map). The chaotic properties of logistic maps and hence MLM have been established in the past decades by several researchers [73].

Shannon [94] explains that a good crypto-system must show diffusion and confusion properties. Confusion refers to making the relationship between the key and the ciphertext as complex and involved as possible while diffusion means that the output bits should depend on the input bits in a very complex way i.e. a change in a bit in input plain text should imply a change in output bit with a probability of $\frac{1}{2}$. Chaotic systems show random behavior and inherently exhibit confusion with respect to the initial conditions ($x_0$) and the parameter ($\alpha$) that make the key. We perform some statistical tests to test the pseudo-random nature of the key obtained.

4.7.1 Randomness Tests

We perform the following randomness tests to study the pseudo-random nature of sequence $(b_n)$ generated using the proposed scheme.
4.7.1.1 Frequency Test

In a randomly generated N-bit sequence we would expect approximately half the bits in the sequence to be ones and approximately half to be zeroes. The frequency test checks that the number of ones in the sequence is not significantly different from N/2.

Based on 1000 simulations on strings of length 10000 each generated using variable initial values and control parameter, the probability for zero and one were obtained to be 0.4993 and 0.5007 respectively for the sequence $b_n$. For the non-binary sequence $z_n$, frequency test was performed by discretizing the sequence around its mean value. We observed the probability of zeros and one in this sequence to be 0.4981 for 1000 simulations of length 10000.

4.7.1.2 Serial Test

The serial test checks that the frequencies of the different transitions in a binary sequence (i.e., 11, 10, 01, and 00) are approximately equal. This will then give us an indication as to whether or not the bits in the sequence are independent of their predecessors.

For the sequence $b_n$, 1000 simulations of 10000 samples were run. The probabilities for getting 00, 01, 10 and 11 were found to be 0.2503, 0.2491, 0.2480, and 0.2526 respectively (the ideal distribution would give 0.25 for all probabilities).

4.7.1.3 Runs Test

The binary sequence is divided into blocks (runs of ones) and gaps (runs of zeroes). The runs test checks that the number of runs of various lengths in our sequence are similar to what we would expect to find in a random sequence. This test is only applied if the sequence has already passed the serial test in which case it is known that the number of blocks and gaps are in acceptable limits.

This is a test of the hypothesis that the values in a sequence come in a random order, against the alternative that the ordering is not random. For non-binary sequences (such as $z_n$) the test is based on the number of runs of consecutive values above or below the mean of input sequence. Too few runs is an indication of tendency of high values to cluster together, and low values to cluster together. Too many runs is an indication of a tendency for high values and low values to alternate. Tests were
performed using Matlab simulations. The result is $H=0$ if the null hypothesis (“sequence is random”) cannot be rejected at the $5\%$ significance level, or $H=1$ if the null hypothesis can be rejected at the $5\%$ level. We ran 10000 simulations with different initial values and parameter settings, giving us 8916 successful simulations with $H=0$.

### 4.7.1.4 Statistical Properties

Some of the necessary conditions for a secure stream cipher are long period, large linear complexity, randomness and proper order of correlation immunity [90]. A long period is assured by taking a large value of $N$ (say 64). Figure 4.5 (a) and (b) show the low correlation between sequences obtained using slightly different (a) initial value $x_0$ and (b) parameter $\alpha$. It can be seen that a very poor correlation is obtained amongst sequences generated using slightly different initial condition or parameter.

### 4.7.2 Bifurcation Map

If the dynamical system under consideration is a chaotic map, then the orbit derived from any initial condition covers the whole phase space. This is seen with the help of bifurcation diagram of logistic maps. A bifurcation diagram is the plot of sample set of $x_n$ obtained against the variations in initial parameter $\lambda_{LM}$. 
Figure 4.6  Bifurcation Diagram for (a) Logistic Map showing the white spaces (islands of stability) and asymmetricity and (b) Modified Logistic Map with symmetric and flatter distribution

The bifurcation map of logistic map is shown in figure 4.6(a). It is observed that for some value of $\lambda_{LM}$, the logistic map reaches a few stable states and oscillate around them. These regions must be removed carefully from the key space. Hence, an exhaustive elimination of stable points (corresponding to white spaces in bifurcation diagram) is necessary to build a scheme based on Logistic Map.

Figure 4.6(b) shows the bifurcation map of MLM as a function of free parameter $\alpha$. It can be seen that there are no free white spaces in the bifurcation diagram, indicating no in-between regions of stable oscillations in MLM. Thus, the entire range of parameter $\alpha$ can be used to build the key space.

4.7.3 Lyapunov Exponent

Lyapunov exponent is a measure of stability of non-linear systems. It characterizes the rate of separation of infinitesimally close trajectories. The maximum Lyapunov exponent is defined by the following expression:

$$\Lambda = \lim_{t \to \infty} \frac{1}{t} \ln \left( \frac{\delta Z(t)}{\delta Z_0} \right)$$

where $\delta Z(t)$ is the separation at time $t$ and $\delta Z_0$ is the initial divergence. In our cipher, if we choose two different initial values $x_{0a}$ and $x_{0b}$, which are very close to each other such that $x_{0a} - x_{0b} \approx \delta Z_0$, a positive Lyapunov exponent will indicate that the two trajectories will diverge from each other. The
Probabilities of Zero | 0.4993  
Probabilities of One | 0.5007  

Table 4.1 Statistical performance of Generated Sequence $b_n$ (results based on 1000 sequences of length 10000 each).

discrete time equivalent expression to find Lyapunov exponent of MLM will be:

$$\Lambda = \lim_{n \to \infty} \frac{1}{n} \ln \frac{\delta x_n}{\delta x_0}$$

$$= \lim_{n \to \infty} \frac{1}{n} \ln \frac{\delta x_n}{\delta x_{n-1}} \frac{\delta x_{n-1}}{\delta x_{n-2}} \cdots \frac{\delta x_2}{\delta x_1} \frac{\delta x_1}{\delta x_0}$$

An analysis similar to logistic map [115] can be performed to prove the positive Lyapunov exponent for logistic maps.

$$x_n = \lambda x_{n-1} (1 - x_{n-1}) + \mu$$

Hence,

$$\left| \frac{\delta x_n}{\delta x_{n-1}} \right| = |\lambda \times (1 - 2x_{n-1})|$$

Therefore, we can express $\Lambda$ as follows:

$$\Lambda = \lim_{n \to \infty} \frac{1}{n} \left( \sum_{j=1}^{n} \ln \left| \frac{\delta x_j}{\delta x_{j-1}} \right| \right)$$

$$= \lim_{n \to \infty} \frac{1}{n} \left( \sum_{j=1}^{n} \ln |\lambda (1 - 2x_j)| \right)$$

The value of $\Lambda$ can be calculated by running a numerical trial of large number of samples (say 10,000) starting with any randomly picked initial value $x_0$. The values of Lyapunov exponent for Logistic Map and MLM are plotted in figure 4.7(a) and (b). This value was found to be $ln2$ for MLM which is the same as the value for Logistic Map with $\lambda_{LM} = 4$. Thus, the divergence rate of MLM, measured by Lyapunov coefficient is always greater than or equal to the value for Logistic Map. This indicates better confusion properties of MLM. Moreover, it is independent of $\alpha$ indicating the invariance of confusion properties with the change in parameter $\alpha$. 

Figure 4.7 Plot of Lyapunov Coefficient ($\Lambda$ - solid line) for (a) Logistic map as a function of parameter $\lambda_{LM}$ indicating regions of non-chaotic behavior and (b) Modified Logistic map showing higher divergence than Logistic Map and independence of $\Lambda$ from parameter $\alpha$.

4.8 Security Enhancement

A serious drawback of chaotic crypto-systems is that they are weak against known-plaintext attacks. If the plain-text and the cipher-text are known, it is easy to XOR both the values and obtain the key value that was XORed to the original plaintext. Our proposed scheme has many advantages over Logistic Map:

- The Modified Logistic Map has better security properties than the Logistic Map. Figure 4.5 shows the sensitivity of MLM to the initial conditions. A slight difference in the initial condition leads to outputs which are completely uncorrelated. The bifurcation map for LM and MLM are shown in Figure 4.6. The absence of any white space in the keyspace of MLM allows us to build a continuous key-space. Figure 4.7 shows the graph for Lyapunov exponent for MLM which is higher than LM. A positive and higher Lyapunov exponent indicates the rate of divergence of two closely related inputs for the system.

- The random feedback scheme makes it difficult to predict the key value XORed to the original plaintext.

- The sequences $s_n$ and $p_n$ are linearly uncorrelated from each other making it difficult to reverse
engineer the values of $p_n$ from $s_n$.

- The sequence $p_n$ is obtained by sampling of $x_n$ which is used to iterate the chaotic map. In the hardware implementation (presented in next section), we sample the Least Significant 16 bits (out of 64) of $x_n$ to get $p_n$. Because, the chaotic map is more sensitive to the MSB than to the LSB (and we have 48 unknown MSB bits), it is practically impossible to trace back the $x_n$ value.

- We allowed 100 iterations of MLM in the beginning to allow the diffusion of initial key bits and parameter values. It was found that within approximately 20 iterations of Logistic Map the initial parameter values are fully diffused: the two logistic maps with a slight difference in initial conditions will appear completely de-correlated in their outputs after at most 20 iterations. Allowing 100 iterations, help us to be on a safer side to allow full diffusion of the initial key parameters.

- Use of DWT parameterization adds to the security of the scheme. The exact choice of DWT filter is given by a secret key. Lack of this knowledge will lead to inexact extraction of plain-text after decrypting the cipher-text.

The ICO shows good results against runs test, serial test, correlation test etc which are used to prove the randomness of output $s[n]$ or $s_n$.

### 4.9 Hardware Implementation

Figure 4.8 shows the hardware architecture for Modified Chaotic Filter Bank (MCFB) Scheme. The input $x[n]$ is first pipelined for eight cycles and then the parameterized DWT filter is applied over it. The nine pipelined stages are then reduced to five by adding the stages with similar wavelet coefficients together to get $w_i[n]$ ($w_i[n] = x[n + i] + x[n - i]$, $i \in \{0, 4\}$). These are then multiplied with the $a$, $a^{-1}$ and $a^2$ values and summed up to get the low pass and high pass values $y_0[n]$ and $y_1[n]$. The outputs of two Improved Chaotic Oscillators is then added to these two signals to get $z_0[n]$ and $z_1[n]$ respectively.

The hardware architecture of ICOs is shown in Figure 4.9. Two instances of ICOs are required in the design.
Figure 4.8  Hardware architecture for the Modified Chaotic Filter Bank Scheme

Some optimization steps performed to reduce the cost of the underlying hardware are summarized below:

1. Division by binary coefficients (e.g. \(1/64, 1/16, 1/4\)) was performed using arithmetic shift operations.

2. The input stream was pipelined. As shown in Figure 4.8, our architecture takes one pixel (or channel input) as the input and outputs the low and high pass signal coefficients with a finite latency. Increasing the system latency allows us to achieve a higher clock speed (and hence higher throughput).
The hardware implementation of proposed architecture was done using the Xilinx ISE 10.1 tool. The target device is a Xilinx Virtex-5 XC4VLX330 FPGA. The input $x[n]$ is 8 bits wide, the intermediate values $y_i[n]$ and $z_i[]$ are represented in 16 bits precision. The Chaotic Oscillator is implemented with an internal bit width of 64 bits, while only last 16 bits of the output of Modified Logistic Map contribute to the pseudo-random number generated by ICO. This prevents any cryptanalysis of ICO while requiring some extra computations. The 16 bit output of each ICO are added to the outputs $y_i[n]$ to get the output signal $z_i[n]$. Modulating the amplitude of ICO output ($s_i[n]$) allows us to change the range of the subband signal power to the chaotic subband power dynamically.

As mentioned, the iterating value of MLM ($x(i)$) and the parameters $\lambda$ and $\mu$ are both implemented with 64 bits fixed-point precision. The permissible range of parameter $\alpha$ was chosen to be $[0, 0.375]$ which is represented in fixed point with 0 integer bits and 64 fractional bits. This is represented shortly as 0.64 in I.F (Integer.Floating point) format. The range for parameter $\lambda$ is then calculated to be $[4, 16]$ which is implemented with 5.59 I.F format. The range for $\mu$ is $[-3, -15.0975]$ which is represented using 5.59 I.F format. Thus, the multiplication $\lambda \times x(i) \times (1 - x(i))$ is truncated to 5.59 I.F format and then added to $\mu$ to obtain the new value for $x(i)$.

A direct implementation gave a clock frequency of 67.8 MHz while requiring 48 DSP48E slices.
present in the Virtex-5 FPGA for efficient multiplication and addition operations. We present two optimizations to improve the clock frequency of the design while reducing the hardware requirements of the design.

Reconfigurable Constant Multiplier design and implementation for SWT has been explained in previous chapter.

4.9.1 Hardware Optimizations for ICO

A single DSP48E slice can perform a maximum of $25 \times 18$ bits multiplication and hence 12 slices are required for a $64 \times 64$ bits multiplication. Two multiplication require 24 DSP48E slices.

We present an optimization of usage of DSP multipliers based on above observations for the multiplication of two 64 bit numbers $X$ and $Y$. $X$ is sign extended to 72 bits ($X_{SE}$ and represented by $X_aX_bX_c$ where $X_a$, $X_b$ and $X_c$ are each 24 bit long sequences.

\[
\{X_{SE}\}^{71}_0 = \{X_a\}^{48}_0\{X_b\}^{24}_0\{X_c\}^{23}_0
\]

Similarly, we can represent $Y$ as combination of four 16 bit numbers $Y_wY_xY_yY_z$.

\[
\{Y\}^{71}_0 = \{Y_w\}^{63}_0\{Y_x\}^{47}_0\{Y_y\}^{31}_0\{Y_z\}^{15}_0
\]

Numerically,

\[
X = X_{SE} = X_a \times 2^{48} + X_b \times 2^{24} + X_c
\]

, and

\[
Y = Y_w \times 2^{48} + Y_x \times 2^{32} + Y_y \times 2^{16} + Y_z
\]
The product \( X \times Y \) can then be represented as:

\[
X \times Y = (X_a \times 2^{48} + X_b \times 2^{24} + X_c) \times (Y_w \times 2^{48} + Y_x \times 2^{32} + Y_y \times 2^{16} + Y_z)
\]

\( \Rightarrow X \times Y = 2^{96} \times X_a Y_w + 2^{72} \times X_b Y_w + 2^{48} \times X_c Y_w + 2^{80} \times X_a Y_x + 2^{56} \times X_b Y_x + 2^{40} \times X_a Y_y + 2^{32} \times X_b Y_y + 2^{16} \times X_c Y_y + 2^{48} \times X_a Y_z + 2^{24} \times X_b Y_z + 2^{16} \times X_c Y_z
\]

Now, considering the product \( X_n(1 - X_n) \) in the logistic map, we multiply two 0.64 I.F values to get an output which is in 0.128 I.F format. We truncate the last 64 bits to get the 64 bit approximate value of \( X_{n+1} \). Because \( X \) is represented in 72 bits, we can discard lower 72 bits of the product. Each of the product \( X_\alpha Y_\beta \), such that \( \alpha \in \{a, b, c\} \) and \( \beta \in \{w, x, y, z\} \) is of size 40 bits and can be implemented in a single DSP48E slice.

Thus,

\[
X \times Y = 2^{96} \times X_a Y_w + 2^{72} \times X_b Y_w + 2^{48} \times X_c Y_w + 2^{80} \times X_a Y_x + 2^{56} \times X_b Y_x + 2^{40} \times X_a Y_y + 2^{32} \times X_b Y_y + 2^{16} \times X_c Y_y + 2^{48} \times X_a Y_z + 2^{24} \times X_b Y_z + 2^{16} \times X_c Y_z
\]

The other multiplication operation can also be optimized in a similar manner. Thus, we can reduce the hardware requirements and critical path for the implementation.

The above mentioned optimizations enhance the performance of original design. The use of reconfigurable LUTs instead of multipliers reduces the critical path of DWT architecture by replacing a multiplication operation with a Look-Up operation. The second optimization - truncating the extra
hardware for building ICO reduces the number of DSP slices used by the design by 33%.

The original design required 14 10x9 bits multipliers and 4 64x64 bits multiplier which required 48 DSP48E slices and Look Up Tables for implementation. The optimized implementation uses only 32 24x16 bits multiplier which are implemented in 32 DSP48E slices. Moreover, the achievable clock frequency increase by 30% from 67.8 MHz to 88.3 MHz.

4.10 Conclusions

This paper presents a novel chaotic filter bank based scheme for cryptographic operations. The scheme, based on modified logistic map is suitable for embedded real-time applications and resistant to known cryptanalysis. The scheme can be used with image compression algorithms such as JPEG2000.

This paper also presents a reconfigurable hardware implementation of the proposed scheme. Use of reconfigurable hardware allows partial removal of hard-multipiers from the design and gives improvement in clock frequency by 30%. The hardcoded key parameters (a values) can be changed by the use of partial reconfiguration techniques.
CHAPTER 5. CHAOTIC ARITHMETIC CODING

Arithmetic Coding (AC) is widely used for the entropy coding of text and multimedia data. It involves recursive partitioning of the range \([0,1)\) in accordance with the relative probabilities of occurrence of the input symbols. In this paper, we first present an interpretation of Arithmetic Coding (AC) in terms of iterations over piece-wise linear chaotic maps and then define a family of such maps, each yielding the same compression efficiency. We next present an image/video encryption scheme based on arithmetic coding, which we call as Chaotic Arithmetic Coding (CAC). CAC uses a key to make the exact choice of map chosen from the family of predefined maps to perform AC. It has the effect of scrambling the intervals without making any changes to the width of interval in which the codeword must lie, thereby allowing encryption without sacrificing any coding efficiency and can be video encryption with compression algorithms such as H.264/AVC etc. We next describe Binary Chaotic Arithmetic Coding (BCAC), a special case of CAC with only two symbols (0 and 1). We finally present two security enhancements to alleviate the known limitations of arithmetic coding-based encryption procedures and give qualitative and quantitative performance of BCAC.

5.1 Introduction

The issue of providing both compression and security simultaneously is gaining importance given the ubiquitous nature of compressed media files, challenging demands of video compression systems and varsity of application requirements in modern context (v.i.z. mobile phones, ipods, notebooks, HDTV etc). The emerging cloud computing infrastructure, as of 2009, consists of reliable services delivered through data centers and built on servers. Video communications in such scenarios will require highly scalable, secure, easily search-able/index-able compressed bitstreams.

Video communication is characterized by a number of peculiarities, such as large data size, real-
time requirements, the use of standardized video codecs, standardized data compression formats, and application-specific security requirements.

Arithmetic coding is a data compression technique that encodes data by creating a code string which represents a fractional value on the interval \([0, 1)\). When a string is converted to arithmetic encoding, frequently-used characters are stored with fewer bits and not-so-frequently occurring characters are stored with more bits, resulting in fewer bits used in total [51]. It typically enables very high coding efficiency as multiple symbols are coded jointly and has been adopted for use in image compression standards, including JBIG-2, JPEG-LS, JPEG2000 and video standard H.264/AVC to provide lossless entropy coding.

Arithmetic coding is extremely efficient for compression efficiency in large data-sizes and it achieves the Shannon compression efficiency for large chunks of data. However, as conventionally implemented, it is not particularly secure. A naive choice is to use the well-known encryption methods such as the Advanced Encryption Standard (AES) in combination with traditional arithmetic coder to satisfy both compression and security needs. However, this proposal leads to increased computational complexity and the useful properties of compressed bitstream such as rate-adaptive transmission, scalability and DC-image extraction for content searching [65] are lost because of use of generic encryption algorithms such as AES or DES over compressed bitstream.

Many multimedia-specific encryption algorithms have been proposed in research literature. Many of these schemes alleviate the computational overhead of naive approach by the selective encryption of important segments/ portions of the video. A good survey of existing algorithms can be found in [60]. Many selective encryption schemes have been found to be insecure against cryptanalysis, because unencrypted coefficients leak significant amount of information [116]. Moreover, these schemes generally lead to compression inefficiency. They are also not compliant to the standardized video codecs because their implementation changes more or less the structure of the codec.

Recently some efforts have been made towards joint design of encryption and compression modules (particularly the entropy coding techniques such as arithmetic coding) to enable such properties in compressed bitstream. These techniques allow encryption at little/ no computational overhead and (in most cases) preserve the format compatibility of compressed bitstream. For example- Liu et al. [62]
Figure 5.1 A sample piece-wise linear map for arithmetic coding like compression (a) The entire map is shown ($\rho$), (b) A single linear part of the map ($\theta_k$) is zoomed. It can have a positive or negative slope depending on choice.

presented a system using table-based bit sequence substitutions to enable the arithmetic coding stage to be simultaneously used for encryption. The authors in [114, 117] associate a fixed length index to each variable length codeword to encrypt the indexes. However, all these approaches suffer from compression inefficiency while the latter also leads to generation of emulated markers. In [15], a chaos-based adaptive arithmetic coding technique was proposed. The arithmetic coder’s statistical model is made varying in nature according to a pseudo-random bitstream generated by coupled chaotic systems. Many other techniques based on varying the statistical model of entropy coders have been proposed in literature, however these techniques suffer from losses in compression efficiency that result from changes in entropy model statistics and are weak against known attacks [42]. Recently, Grangetto et al. [36] presented a Randomized Arithmetic Coding (RAC) scheme which achieves encryption by inserting some randomization in the arithmetic coding procedure at no expense in terms of coding efficiency. RAC needs a key of length 1-bit per encoded symbol. Wen and Kim et al. [45] presented a generalization of this procedure, called as Secure Arithmetic Coding (SAC). The SAC coder builds over a Key-Splitting Arithmetic Coding (KSAC) [113] where a key is used to split the intervals of an arithmetic coder and it adds input and output permutation to increase the security of coder. Some limitations and features of KSAC are presented next giving the motivation to improve the security.
performance of arithmetic coding:

1. SAC introduces loss in coding efficiency particularly for small sized inputs, which are later restricted to a small value by putting some constraints on the keyspace [113].

2. Every split, doubles the computational overhead of the coder. Thus, the SAC encoder may have to work with multiple sub-intervals and needs to compute one shortest representation arithmetic code for each subinterval, thereby significantly increasing the computational cost of encoder.

3. The memory requirements of SAC coder are at least double that of BAC.

4. Successful attacks have been demonstrated against SAC scheme[42, 126, 127, 100].

The joint compression and encryption algorithms, in general, and particularly this paper attempts to build a reasonably secure video encryption scheme without incurring any/little overhead in computational cost or the compression-ratios. It is suitable for use in low power embedded multimedia systems such as video camcorders, surveillance cameras, ipods, and other battery-operated devices. These schemes have an added advantage that they preserve the useful properties of compressed bitstream. Thus, our efforts complement and not compete against the secrecy promise of the naive implementation (BAC followed by strong cryptographic cipher such as AES).

With this motivation, we build a joint video encryption and compression scheme based on Piece-Wise Linear Chaotic Maps, called as Chaotic Arithmetic Coding. The general encoding and decoding procedure are explained in Section 5.2. From Section 5.3 onwards, we restrict our discussion to Binary CAC (BCAC) which is most relevant for commercial applications in image and video processing standards. We explain how recent advancements in joint video compression and encryption using Binary Arithmetic Coding(BAC) can be interpreted using BCAC, and discuss their limitations and strengths. Section 5.4 presents some security enhancements for BCAC scheme and discusses the strengths and weaknesses of proposed schemes against cryptanalysis. Section 5.5 gives experimental results on compression efficiency of the scheme. We conclude the paper in Section 5.6 with the discussion of results.
5.2 Piece-wise Linear Chaotic Maps

Let us consider a scenario where we have a string $S = x_1, x_2, ... x_N$ consisting of $N$ symbols to be encoded. The probability of occurrence of a symbol $s_i, i \in 1, 2, ... n$ is given by $p_i$ such that $p_i = N_i/N$ and $N_i$ is the number of times the symbol $s_i$ appears in the given string $S$. We next consider a piece-wise linear map($\rho$) with the following properties:

- It is defined on the interval $[0, 1)$ to $[0, 1)$ i.e.

$$\rho : [0, 1) \rightarrow [0, 1)$$

- The map can be decomposed into $N$ piece-wise linear parts $\varrho_k$ i.e.

$$\rho = \bigcup_{k=1}^{N} \varrho_k$$

- Each part $\varrho_k$ maps the region on x axis $[beg_k, end_k)$ to the interval $[0, 1)$ i.e.

$$\varrho_k : [beg_k, end_k) \rightarrow [0, 1]$$

The last two propositions lead to:

$$\bigcup_{k=1}^{N} [beg_k, end_k) = [0, 1)$$

- The map $\varrho_k$ is one-one and onto i.e.

$$\forall x \in [beg_k, end_k)$$

$$\exists y \in [0, 1) : y = \varrho_k(x), \text{ and}$$

$$\forall y \in [0, 1)$$

$$\exists x \in [beg_k, end_k) : \varrho_k(x) = y$$
Figure 5.2 The piece-wise chaotic map for N=4. Probability distributions for symbols A, B, C and D are given by \( p(A) = 0.4 \), \( p(B) = 0.3 \), \( p(C) = 0.2 \) and \( p(D) = 0.1 \). The mapping of maps and symbols is given by: \( \varrho_1(x) \equiv A \), \( \varrho_2(x) \equiv B \), \( \varrho_3(x) \equiv C \), and \( \varrho_4(x) \equiv D \).

- \( \varrho \) is a many-one mapping from \([0, 1)\) to \([0, 1)\). This implies that the decomposed linear maps (\( \varrho_k \)) don't intersect each other i.e.

\[
\forall (k \neq j) : [\text{beg}_k, \text{end}_k] \cap [\text{beg}_j, \text{end}_j] = 0
\]

- Each linear map \( \varrho_k \) is associated uniquely with one symbol \( s_i \). The mapping \( \varrho_k \rightarrow s_i \) is defined arbitrarily but one-one relationship must hold.

- The valid-input width of each map \( \varrho_k \), given by \( (\text{end}_k - \text{beg}_k) \) is proportional to a probability of occurrence of symbol \( s_i \).

\[
\text{end}_k - \text{beg}_k \propto p_i
\]

\[
\Rightarrow \text{end}_k - \text{beg}_k = C \times p_i
\]

We recall that \( \sum_{k=1}^{N} (\text{end}_k - \text{beg}_k) \) is same as the input width of \( \bigcup_{k=1}^{N} \varrho_k = \rho \), which is 1. Also, \( \sum_{i=1}^{N} p_i = 1 \). Thus, we get the value of constant \( C \) to be 1.

\[
\Rightarrow \text{end}_k - \text{beg}_k = p_i
\]
Table 5.1 Beginning and end Intervals for given example

<table>
<thead>
<tr>
<th>map</th>
<th>(beg_i)</th>
<th>(end_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\varphi_1)</td>
<td>0</td>
<td>0.4</td>
</tr>
<tr>
<td>(\varphi_2)</td>
<td>0.4</td>
<td>0.7</td>
</tr>
<tr>
<td>(\varphi_3)</td>
<td>0.7</td>
<td>0.9</td>
</tr>
<tr>
<td>(\varphi_4)</td>
<td>0.9</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 5.1(a) shows the full map with different parts \(\varphi_1, \varphi_2, \ldots \varphi_N\) present while Figure 5.1(b) zooms into individual linear part \(\varphi_k\). The maps are placed adjacent to each other so that each input point is mapped into an output point in the range \([0, 1)\). The total number of distinct ways of arranging N maps to obtain \(\rho\) fulfilling the properties mentioned above is given by \(N! = N.(N - 1).(N - 2)\ldots3.2.1\), where ! denotes factorial sign. It is same as arranging these N maps in a sequence, one after another, with the end interval of one map touching the begin interval of another.

However, there are N different piece-wise maps, each with two possible orientations (with positive or negative slope). Thus, the number of total permutations possible is given by \(N!2^N\). Thus, for N-ary arithmetic coding or arithmetic coding with N symbols, it is possible to have \(N!2^N\) different mappings each leading to same compression efficiency. Since we can arbitrarily choose any 1 of the \(N!2^N\) maps, the key space for encoding a single bit of data is \(\lceil \log_2(N!2^N) \rceil\) bits, where \(\lceil \rceil\) represents the greatest integer function. For \(N=2\), it gives 8 mappings. If we increase \(N\) to 4 this value increases to 384.
Table 5.2 Decoding the original sequence for initial value of 0.2

<table>
<thead>
<tr>
<th>Iteration #</th>
<th>I.V.</th>
<th>Int. Map</th>
<th>Symbol</th>
<th>It.V.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration 0</td>
<td>0.2</td>
<td>$0 \leq 0.25 &lt; 0.4$</td>
<td>$\varrho_1$</td>
<td>A</td>
</tr>
<tr>
<td>Iteration 1</td>
<td>0.5</td>
<td>$0.4 \leq 0.25 &lt; 0.7$</td>
<td>$\varrho_2$</td>
<td>B</td>
</tr>
<tr>
<td>Iteration 2</td>
<td>0.3333</td>
<td>$0 \leq 0.25 &lt; 0.4$</td>
<td>$\varrho_1$</td>
<td>A</td>
</tr>
<tr>
<td>Iteration 3</td>
<td>0.8333</td>
<td>$0.7 \leq 0.25 &lt; 0.9$</td>
<td>$\varrho_3$</td>
<td>C</td>
</tr>
</tbody>
</table>

I.V.= Initial Value, Int.= Interval, It.V.= Iterated Value

The equation for individual maps can be derived as follows:

$$y' = \varrho_k(x') = \left( \frac{x' - \text{beg}_k}{\text{end}_k - \text{beg}_k} \right) \text{ or } \left( 1 - \frac{x' - \text{beg}_k}{\text{end}_k - \text{beg}_k} \right)$$

The equation for the full map is given by

$$y = \rho(x) = \varrho_k(x) : \text{beg}_k \leq x < \text{end}_k$$

5.2.1 The coding procedure

The piece-wise linear maps form a dynamic system, thus, the trajectory depends precisely on initial value. We try to find an initial value which, when iterated will give us the set of symbols to be encoded. For example- For $N = 4$, we have four symbols A,B,C and D such that $p(A) = 0.4$, $p(B) = 0.3$, $p(C) = 0.2$ and $p(D) = 0.1$. The symbols are arbitrarily mapped to four maps: $\varrho_1(x) \equiv A$, $\varrho_2(x) \equiv B$, $\varrho_3(x) \equiv C$, and $\varrho_4(x) \equiv D$ and the mapping is shown in figure 5.2. The intervals $\text{beg}_i$ and $\text{end}_i$ are given in Table 5.1.

In this case, we try to find an initial value which when iterated over this map will give us the set of symbols (A,B,C, or D) to be encoded. For $\text{Num} = 4$ (where $\text{Num}$ is the length of string to be encoded), let us say that the initial value is 0.2. Then the first symbol to be decoded is A as $0 \leq 0.25 < 0.4$, and $\varrho_1 \equiv A$. Similarly, by three iterations on chaotic map $\rho$, we will get three values which will indicate the symbol to be decoded at each step. This has been illustrated in Table 5.2.
Table 5.3 Encoding the original sequence ‘ABAC’

<table>
<thead>
<tr>
<th>Back-Iteration #</th>
<th>I.I.</th>
<th>Symbol</th>
<th>Map</th>
<th>It.I.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Back-Iteration 0</td>
<td>[0,1)</td>
<td>C</td>
<td>$\rho_3$</td>
<td>[0.7,0.9)</td>
</tr>
<tr>
<td>Back-Iteration 1</td>
<td>[0.7,0.9)</td>
<td>A</td>
<td>$\rho_1$</td>
<td>[0.28,0.36)</td>
</tr>
<tr>
<td>Back-Iteration 2</td>
<td>[0.28,0.36)</td>
<td>B</td>
<td>$\rho_3$</td>
<td>[0.484,0.508)</td>
</tr>
<tr>
<td>Back-Iteration 3</td>
<td>[0.484,0.508)</td>
<td>A</td>
<td>$\rho_1$</td>
<td>[0.1936,0.2032)</td>
</tr>
</tbody>
</table>

I.I. = Initial Interval, It.I. = Iterated Interval

Table 5.4 Decoding the codeword 0.2 using Arithmetic coder

<table>
<thead>
<tr>
<th>Codeword</th>
<th>Interval</th>
<th>Corresponding Intervals for A,B,C and D</th>
<th>Decoded Cw.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>(0.1)</td>
<td>[0,0.16), [0.16,0.28), [0.28,0.36), [0.36,0.4)</td>
<td>A</td>
</tr>
<tr>
<td>0.2</td>
<td>(0.0,4)</td>
<td>[0.16,0.208), [0.208,0.244), [0.244,0.268), [0.268,0.28)</td>
<td>B</td>
</tr>
<tr>
<td>0.2</td>
<td>(0.16,0.208)</td>
<td>[0.16,0.1792), [0.1792,0.1936), [0.1936,0.2032), [0.2032,0.208)</td>
<td>A</td>
</tr>
<tr>
<td>0.2</td>
<td>(0.16,0.208)</td>
<td>[0.16,0.1792), [0.1792,0.1936), [0.1936,0.2032), [0.2032,0.208)</td>
<td>C</td>
</tr>
</tbody>
</table>

5.2.2 Correspondence to Arithmetic Coding

Thus, the decoded sequence is ‘ABAC’. Next, we try to back-iterate on the piecewise map to obtain the initial value that will give us the decoded sequence ‘ABAC’. We start from the back and proceed towards first symbol. First, we encode ‘C’. This implies that the value at 4th iteration must lie in the interval [0.7,0.9). We then back-iterate on the piece-wise map $\rho$ along $\rho_1$ because we know that the second last symbol is ‘A’. The entire procedure is shown in Table 5.3. The final interval [0.1936, 0.2032) represents the dynamic range within which the initial value must lie in order to correctly decode the input sequence (Note: 0.1936 ≤ 0.2 < 0.2032).

It can be observed that the particular choice of maps $\rho_i$ in the above example give the same result as arithmetic coding. The arithmetic coding of the sequence ‘ABAC’ will give the output interval as the above mentioned scheme. It can be verified as follows:

‘A’ = [0,0.4); ‘AB’ = [0.16,0.28); ‘ABA’=[0.16,0.208); and ‘ABAC’=[0.1936, 0.2032)

This result holds true for any arbitrary value of N. We next present a logical correlation between iterations on chaotic maps and arithmetic coding by making the following observation(s) in decoding procedure for both schemes: Table 5.2 give the procedure for decoding the CAC while Table 5.4 gives the decoding procedure for standard arithmetic coder. It can be observed that while CAC scales the codeword or initial value to map them to the intervals corresponding to different symbols, the standard
arithmetic coder keeps the codeword constant and instead scales the map in every iteration to find the symbol. It is immaterial - whether one scales the map to suit the codeword or scales the codeword to suit the map - the relative ratios remain the same, hence output of both procedures is the same.

5.2.3 Compression Efficiency

The compression efficiency of the procedure lies in the width of the final interval from which we need to choose the initial value from. Let us consider encoding a general sequence of \( N \) symbols such that probabilities of occurrence of \( i^{th} \) symbol is given by \( \frac{N_i}{N} \) where \( N_i \) is the number of occurrence of the symbol in the sequence. On every iteration, to encode an arbitrary symbol \( N_j \), the width of interval (originally \([0,1)\) and length 1) shrinks by a factor of \( \text{end}_j - \text{beg}_j \) (width of \( \varrho_j \)). Thus, the width \( \delta \) of final interval would be given by:

\[
\delta = \prod_{j=1}^{N} (\text{end}_j - \text{beg}_j)^{N_j}
\]

We have the relation \( \text{end}_j - \text{beg}_j = p_j = \frac{N_j}{N} \), hence

\[
\delta = \prod_{j=1}^{N} \left( \frac{N_j}{N} \right)^{N_j}
\]

The number of bits \( B \) needed to distinguish a point in the particular interval from points belonging to any other interval \( \delta \) of the same size is \( \lceil -\log_2(\delta) \rceil \).

\[
B = \lceil -\log_2(\delta) \rceil = \left\lceil -\log_2\left(\prod_{j=1}^{N} \left( \frac{N_j}{N} \right)^{N_j} \right) \right\rceil
\]

\[
= \left\lceil -\sum_{j=1}^{N} \log_2 \left( \frac{N_j}{N} \right)^{N_j} \right\rceil
\]

\[
= \left\lceil -\sum_{j=1}^{N} N_j \log_2 \left( \frac{N_j}{N} \right) \right\rceil
\]
The average number of bits required per symbol ($B_{av}$) is given by

$$B_{av} = \frac{B}{N} = \frac{1}{N} \left[- \sum_{j=1}^{N} (N_j) \log_2 \left( \frac{N_j}{N} \right) \right]$$

According to Shannon’s entropy equation, the number of bits needed to encode a string of symbols is given by

$$B_{sh} = - \sum_{j=1}^{N} p_i \log_2 p_i$$

Knowing that the symbol probability $p_i$ is given by $p_i = \frac{N_i}{N}$, we get the following expression for $B_{av}$:

$$B_{av} = \frac{1}{N} \left[ N \times B_{sh} \right] \leq \frac{1}{N} \left( N \times B_{sh} + 1 \right)$$

$$\Rightarrow B_{av} \leq B_{sh} + \frac{1}{N}$$

As $N \to \infty$, $B_{av} \to B_{sh}$. Thus, the proposed scheme gives optimal compression for large codewords.

5.2.4 Application to Multimedia/ Data Encryption

CAC is Shannon-optimal in terms of compression efficiency, as proven in the last section. We discussed a particular construct of piece-wise chaotic map and have shown how it is equivalent to arithmetic coding. However, there are many more possible unique maps which will lead to same compression efficiency but lead to completely different final interval and output codeword. By varying the mapping $\varphi_k \to s_i$, we can obtain different maps, all of which give same compression efficiency but different intervals for final codeword.

This parameterization of chaotic piece-wise maps allows us to build a keyspace for data/ video encryption using chaotic arithmetic coding. The choice of mapping is thus governed by an encryption key. A sample example is given now to illustrate how the choice of a wrong key will lead to completely wrong decoding:

Using the original map (see Section II.A and B), we coded the string ‘ABAC’ with the codeword 0.2. (See Tables 5.2 and 5.3 for details of encoding and decoding procedure). Let us define a slightly different map Let us use the chaotic map given in figure 5.3. The iterated values now are 0.2, 0.5,
0.6666 and 0.1111 respectively which will lead to decoded symbols to be ‘ABBA’.

Most arithmetic coders in practice are binary i.e. work with only two input symbols ‘0’ and ‘1’ because of large complexity of arithmetic coder when using multiple symbols [67]. For a Binary Chaotic Arithmetic Coder (BCAC), have eight possible maps for every encoded bit. Thus, we get upto 3 bits of encryption key per encoded symbol. The large keyspace puts an additional burden to communication cost. However, if we can develop an efficient strategy to manage the keys, this large keyspace has an added advantage of providing robustness to brute-force attacks.

As such, the CAC (or BCAC) can be used as a joint compression-cum-encryption technique for data encryption. It is particularly beneficial for data-intensive tasks such as multimedia encryption and compression and can be integrated into the standard video compression algorithms such as JPEG2000, JPEG, MPEG etc.

For full encryption, the entire volume of multimedia data is passed through CAC encoder while in case of selective encryption only the important parts of data are passed through CAC encoder.

If we reveal the first K bits of the key publicly, then a part of the bitstream can be decoded correctly while decoding the entire bitstream will require knowledge of the entire key. In that case, CAC can be used to provide conditional access to part of multimedia content or scalable video encryption [123]. Scalable Multimedia Encryption is required for pervasive/cloud-based multimedia applications where different types of users want to access the same multimedia content at different resolutions and access-privileges.

5.3 Binary Chaotic Arithmetic Coding

In the previous section we explained how arithmetic coding can be viewed as re-iteration on skewed binary map. There are, however, eight equivalent modes of skewed binary maps which can be used for iteration. They are shown in Figure 5.4. These modes differ from each other in the way input is mapped into the chaotic orbit. The maps differ in the interval in which the arithmetic code must lie for a symbol ‘0’ or ‘1’ but the width of interval remains the same. In this section, we will formulate a mathematical procedure to generate the eight maps and choose between them using the parameter $i$. 
5.3.1 Definition

Let us define the generalized skewed binary map with the following equations:

\[
y = \begin{cases} 
n_1 x + c_1 & \text{when } x \leq k \\
n_2 x + c_2 & \text{when } x > k \end{cases}
\]  

(5.1)

\[
\text{Decode } \begin{cases} 
'0' & \text{when } x \in [i1, i2] \\
'1' & \text{when } x \in [i3, i4]
\end{cases}
\]

(5.2)

Then, the back iteration on skewed binary map is defined by the following equations:

\[
x = \begin{cases} 
m_1 y + c_1 & \text{when '0'} \\
m_2 y + c_2 & \text{when '1'}
\end{cases}
\]

(5.3)

where \( n_1 = N1(i), n_2 = N2(i), c_1 = C1(i), c_2 = C2(i), m_1 = M1(i), m_2 = M2(i), \\
b_1 = B1(i), b_2 = B2(i), m_1 = M1(i), m_2 = M2(i), i_1 = I1(i), i_2 = I2(i), i_3 = I3(i), \) and \\
\( i_4 = I4(i) \) and \( i \) varies from 1 to 8 depending on the choice of chaotic map. Table 5.5 gives the value of these parameters for all eight chaotic maps.

5.3.2 Related works

5.3.2.1 Arithmetic coding with non-linear maps

The work by Nagraj et al. [77] derives the equivalence of arithmetic coding and chaotic piece-wise linear maps (which they refer to as GLS coding). They develop a theory for skewed maps (skewed with a non-linearization parameter \( a \) to be used for image encryption) to be used for encryption and compression purposes. The above approach has two main disadvantages, making the scheme prone to cryptographic attacks:

1. A wrong guess in value of the skew parameter \( a \) may lead to imperfect reconstruction and not necessarily to completely random output. The first few symbols of binary string may be correctly guessed by a wrong, but closely related value of \( a \).
2. It is possible to iteratively guess the value of $a$ by launching known plaintext attack. The closer the value of $a$ gets to the original $a$ value, the more symbols will be reconstructed properly.

5.3.2.2 Randomized Arithmetic Coding

Grangetto et al. [36] present a Randomized Binary Arithmetic Coding (RBAC) scheme where they change the ordering of ‘0’ and ‘1’ intervals in a Binary Arithmetic Coder (BAC) based on a key. RBAC can be seen as a special case of BCAC where only two of the eight modes of BCAC are used for encryption purposes (drawn in figure 5.4(a) and (e)).

5.3.2.3 Secure Arithmetic Coding

Kim [45] presented a Secure Arithmetic Coding scheme, based on extension of their work Key-splitting Arithmetic Coding (KSAC) [113] to include input and output permutation. KSAC can be represented in terms of piece-wise linear maps by removing the condition of continuity of individual maps ($\rho_i(x)$). Each part $\rho_i$ maps a discontinuous interval on x-axis to the interval [0,1).

5.3.3 Implementation efficiency

For a normal binary arithmetic coder, at each iteration the starting interval $[I_s, I_e]$ is updated at one end. On encoding a ‘0’ the final interval becomes $[I_s + p(I_e - I_s), I_e]$ while on encoding a ‘1’ the final interval becomes $[I_s, I_s + p(I_e - I_s)]$. Thus, every iteration requires one multiplication and two addition operations. The decoding procedure for a binary arithmetic coder involves updating the interval $[I_s, I_e]$ at one end depending on whether the last decoded symbol was a ‘0’ or a ‘1’. Thus, every iteration again requires one multiplication and two addition operations.

For chaotic arithmetic encoder, both end of interval are updated at every iteration using a linear transformation $x = my + c$ thus requiring two multiplications and two additions for encoding. The decoding is simple as it involves iteration on the chaotic map according to the linear transformation $y = nx + c$ involving a multiplication and an addition operation. There are some additional table lookups involved in chaotic coding to choose the right chaotic map at every iteration which can be
Table 5.5 Parameter List for the eight possible choices of chaotic encoder

<table>
<thead>
<tr>
<th></th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
<th>(e)</th>
<th>(f)</th>
<th>(g)</th>
<th>(h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>p</td>
<td>p</td>
<td>−p</td>
<td>−p</td>
<td>p</td>
<td>−p</td>
<td>−p</td>
<td>p</td>
</tr>
<tr>
<td>B1</td>
<td>0</td>
<td>0</td>
<td>p</td>
<td>p</td>
<td>1 − p</td>
<td>1</td>
<td>1</td>
<td>1 − p</td>
</tr>
<tr>
<td>M2</td>
<td>1 − p</td>
<td>p − 1</td>
<td>p − 1</td>
<td>1 − p</td>
<td>1 − p</td>
<td>p − 1</td>
<td>p − 1</td>
<td>1 − p</td>
</tr>
<tr>
<td>B2</td>
<td>p</td>
<td>1</td>
<td>1</td>
<td>p</td>
<td>0</td>
<td>0</td>
<td>1 − p</td>
<td>1 − p</td>
</tr>
<tr>
<td>N1</td>
<td>1/p</td>
<td>1/p</td>
<td>−1/p</td>
<td>−1/p</td>
<td>1/(1 − p)</td>
<td>−1/(1 − p)</td>
<td>−1/(1 − p)</td>
<td>−1/(1 − p)</td>
</tr>
<tr>
<td>C1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>N2</td>
<td>1/(1 − p)</td>
<td>−1/(1 − p)</td>
<td>−1/(1 − p)</td>
<td>1/(1 − p)</td>
<td>1/p</td>
<td>−1/p</td>
<td>−1/p</td>
<td>1/p</td>
</tr>
<tr>
<td>C2</td>
<td>−p/(1 − p)</td>
<td>1/(1 − p)</td>
<td>1/(1 − p)</td>
<td>−p/(1 − p)</td>
<td>(p − 1)/p</td>
<td>1/p</td>
<td>1/p</td>
<td>(p − 1)/p</td>
</tr>
<tr>
<td>I1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>(1 − p)</td>
<td>(1 − p)</td>
<td>(1 − p)</td>
<td>(1 − p)</td>
</tr>
<tr>
<td>I2</td>
<td>p</td>
<td>p</td>
<td>p</td>
<td>p</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>I3</td>
<td>p</td>
<td>p</td>
<td>p</td>
<td>p</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>I4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1 − p</td>
<td>1 − p</td>
<td>1 − p</td>
<td>1 − p</td>
</tr>
<tr>
<td>K</td>
<td>p</td>
<td>p</td>
<td>p</td>
<td>p</td>
<td>1 − p</td>
<td>1 − p</td>
<td>1 − p</td>
<td>1 − p</td>
</tr>
</tbody>
</table>
efficiently implemented in software/hardware. Thus, CAC encode requires more computations than BAC encode while CAC decode requires less computations than BAC decode.

5.4 Cryptanalysis & Security Enhancements

As mentioned above, Arithmetic coding based security scheme have been found to be vulnerable to simple cryptographic attacks. An attacker can guess the key, in $O(N)$ operations by giving different known inputs to the system (known-plaintext attack). In this section, we mention two security enhancement modes for BCAC, which add considerable levels of security to the design.

5.4.1 Feedback ($Fb$) Mode

In the feedback mode, the output compressed (and encrypted) bits are XORed to the key every iteration. Thus, the key is changed every iteration, making it difficult to perform a cryptanalysis. For an N-bit BCAC coder, the key length is 3N bits while the compressed output is M bits ($M \leq N$). Thus, the M bits are repeated to obtain a 3N bits long string for XORing. It is explained in Algorithm 1.

$Fb$ mode offers some security to the encrypted stream by preserving the key.

5.4.2 Pair Wise Independent Keys ($PWIK$) Mode

In $PWIK$ mode, independent keys are generated for each iteration of the BCAC coder using two initial values. The same values can be reconstructed in the decoder side with prior knowledge of these initial values. However, the generated key values are pairwise independent from each other. This method uses Galois field mathematics and we take $3N = 256$ or $N = 85$ for BCAC to simplify the operation. It is explained in Algorithm 2. The generated keys are shown to be pairwise-independent by Jutla et al. [44]. This gives added security to the scheme against any cryptanalysis.

5.4.3 Resistance to Known Attacks

Assessing security for any encryption system is a challenging task because showing robustness against known attacks does not preclude the existence of unknown attacks against which the system may not be robust. This applies to mature encryption standards such as AES [31], DES [32]. We
Algorithm 1 Generate Keys - Fb mode

1: **GenerateKeys_Fb_mode()**
   
   \{Output_{i-1}\} : Encoded Output of (i-1)th pass
   
   \{M\} : Size of Output_{i-1}
   
   \{I\}nitValue : Initial Seed
   
   \{N\} : Length of encoded message
   
   \{INITIALIZE:\}

2: \(K_{e_0} = \text{InitValue}\)
   
   \{RECURSION:\}

3: **for** \(j = 1; j \leq 3N; j++\) **do**

4: \(\text{String}_i(j) = \text{Output}_{i-1}((j)\mod(M))\)

5: **end for**

6: \(\text{Key}_i = \text{Key}_{i-1} \oplus \text{String}_i\)

Algorithm 2 Generate Keys - PWIK mode

1: **GenerateKeys_PWIK_mode()**

   \{InitValue\_1 and InitValue\_2\} : 2 Initial Seeds of length 256
   
   \{Key\_j\} : Encoding Key for j-th pass
   
   \{p\} : Largest prime in GF(2^{256})
   
   \{INITIALIZE:\}

2: \(K_{e_0} = \text{InitValue\_1}\)

3: \(\text{Key}_i = (\text{Key}_{i-1} + \text{InitValue\_2}) \mod 2^{256}\)

4: **if** \((\text{Key}_i < \text{InitValue\_2})\) **then**

5: **return** \(\text{Key}_i = \text{Key}_i + 2^{256} - p\)

6: **end if**

therefore adopt a similar approach that considers known attacks and ensures that they cannot be used successfully.

One great security advantage of presented scheme is that the output from the engine is in the form of variable sized words and the individual bit output corresponding to inserted symbols cannot be determined. The authors in of KSAC [45] mention the weakness of Arithmetic coding based encryption schemes, which applies to the proposed scheme as well: "In the context of a secure arithmetic coder, potential weaknesses lie in the ability to correlate the input symbol stream with attributes of the output binary codeword and to use those correlations to infer key information. The core of the encoder, the Interval Splitting AC, when implemented without any input permutation and codeword permutation,
can be attacked using carefully constructed sequences that reveal split locations.'

They propose an input and output permutation with KSAC which obscures this relationship as a possible solution. However, recent cryptanalysis of KSAC paper has shown serious weaknesses of these permutations [42, 126, 127, 100]. The authors in [42] present a cryptanalysis of this scheme where they reveal that a key of length 2000 bits can be broken with as few as 50000 plaintexts.

Known-plaintext attacks are difficult to mount against arithmetic coding based encryption schemes in general, and BCAC in particular, because the back-iterations over chaotic map, give a rather uncorrelated outputs for even similar plaintext inputs.

Chosen-plaintext attacks can be easily mounted over RAC, KSAC and also over BCAC coder. However, intelligent key-scheduling, as mentioned in above subsection can help provide the desired level of security. The above mentioned encryption modes for BCAC change the encryption key for every iteration and both of them can therefore resist such attacks.

The specific advantage of Fb mode lies in the fact that the the key keeps changing every cycle, without any external key-scheduling mechanism. This reduces the implementation cost and gives us the flexibility of using BCAC+Fb mode for string of any length. However, the interdependence of the keys (amongst different iterations) makes them vulnerable to related-key attack [14] which can also be coupled with known plaintext attacks etc. Consider a case, where the attacker gives an all-zeros plaintext to BCAC+Fb coder. He observes the output, which is then XORed with the key. Successive iterations give specific details about the original key which has been flipped in some positions according to output bits. Observing this over a few iterations may yield important information about the key to the attacker.

The BCAC+PWIK mode allows us to resist these kind of attacks because the keys used in different iterations are pairwise independent, hence, an attacker cannot find any correlation between subsequent output bits corresponding to same plaintext value. However, it comes with an extra implementation cost of PWI Key generation module. Either of the two proposed modes have no effect compression efficiency, which is a significant advantage against some proposed techniques [45, 15, 77]. A drawback, of this mode is that it involves GF mathematics, and it would be preferrable to sacrifice the flexibility of choosing the length of input bits to suit the GF operations.
Table 5.6 Compression Performance of BAC and BCAC for various length strings. The average length and standard deviation of codeword is presented for various $p$ values and various length of input string.

<table>
<thead>
<tr>
<th></th>
<th>$N = 10$</th>
<th>$N = 100$</th>
<th>$N = 1000$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BAC</td>
<td>BCAC</td>
<td>BAC</td>
</tr>
<tr>
<td>$p = 3/5$</td>
<td>$8.7876 \pm 1.734$</td>
<td>$8.733 \pm 1.74$</td>
<td>$96.16 \pm 3.06$</td>
</tr>
<tr>
<td>$p = 6/7$</td>
<td>$5.3252 \pm 2.75$</td>
<td>$5.2222 \pm 3.15$</td>
<td>$58.30 \pm 9.19$</td>
</tr>
<tr>
<td>$p = 10/11$</td>
<td>$4.177 \pm 2.55$</td>
<td>$3.57 \pm 2.90$</td>
<td>$43.04 \pm 9.55$</td>
</tr>
</tbody>
</table>

Comparison with BAC+AES

BAC followed by encryption with AES is the naive candidate which should provide best security and compression. AES is extremely fast when it is fully pipelined in hardware [124]. However, the sequential nature of BAC coder becomes the bottleneck in a combined BAC+AES system.

The arithmetic operations required for one bit encoding and decoding using BAC is 4 adders and 2 multipliers (discussed in section 5.3.3). AES-128 bits require 40 sequential transformation steps composed of simple and basic operations such as table lookups, shifts, and XORs. It needs approximately 336 bytes of memory and approximately 608 XOR operations. BCAC coder requires 4 adders and 4 multipliers, and (for $N=128$) additional 128 bit (mod 128) adder (if using PWIK mode). Thus, the hardware requirements of BCAC coder are much less than BAC and AES combined.

Since, the key scheduling can be done in parallel, the throughput of BCAC is equivalent to BAC coder (much faster than BAC+AES).

5.5 Compression

BCAC gives the same compression efficiency as BAC coder. We performed some experiments to verify these facts. We ran an implementation of BCAC over Matlab 7.8.0 (R2009a) and used variable precision arithmetic (vpa) tools in Symbolic Mathematics Toolbox to run simulations for large values of N (such as N=100,1000).

The simulation results show a slight better performance for CAC over normal arithmetic coder (AC) especially for small values of N. However, as mentioned above there is no objective reason for such occurrence. The results are presented in Table 5.6 (The reported value is the average length of output
114

bitstream and the standard deviation). 1000 simulations each were run in Matlab to obtain the mean value of output bitstream lengths.

The slight difference in mean values of BAC and BCAC can be ignored as the standard deviation of the output obtained over 1000 simulations is much greater than the difference between mean values.

5.6 Summary

In this paper we presented a joint compression and encryption scheme for multimedia data using chaotic maps. We introduced data/video coding using piecewise-linear maps and then parameterized the maps to obtain a key-space for video encryption. We presented some security enhancements to alleviate the weaknesses of presented scheme against cryptanalysis. The presented scheme incurs no loss to compression performance, and it was shown that it achieves higher throughput than the naive encryption algorithms.

Future works include developing a mathematical model to demonstrate the robustness of the security enhancements against cryptanalysis. The proposed scheme can be implemented in hardware to obtain a high throughput. It can be used with MPEG/ JPEG2000 encoders by incorporating content-adaptive models into it.
Figure 5.4 (a-h) show the eight modes of the skewed binary map (p=0.6).
CHAPTER 6. CONCLUSIONS AND FUTURE WORK

In this research, we have made significant contributions to the development of algorithms and architectures for security of embedded multimedia systems.

The proposed schemes allow efficient multimedia encryption without the need of any conventional cipher. Secure Wavelet Transform, Chaotic Filter banks and Chaotic Arithmetic Coding scheme can be tied together to build a stronger crypto-system. We envision that the proposed approach - of parameterizing the compression operations to build a keyspace, can be used to embed encryption features into other compression modules, and can be extended to other scenarios such as audio coding, and network coding. We also demonstrated how the efficient use of signal processing expertise can lead to efficient hardware implementation (in case of Poly-DWT and SWT).

Aside from the particular technical solutions proposed in the three areas above, this project has the potential of bringing a security focus to image processing and computer vision algorithms themselves and power-awareness to the design of secure multimedia systems. The conducted research can be extended to serve the security requirements of mission critical surveillance systems used by the police and armed forces, and also ensure the security of widely used portable multimedia systems.

This project also leads several open dimensions for future researchers to work upon and develop them into working ideas. Some of them are enumerated as follows:

1. Interested researchers can further develop the work in Chaotic Arithmetic Coding to incorporate context-adaptive features and provide a hardware architecture for efficient implementation of CAC.

2. Other video compression blocks such as motion compensation and estimation, DCT etc can be similarly parameterized and a combined cryptosystem can be built.
3. The thesis provides parameterization as the basis of building joint multimedia encryption and compression schemes. Similar schemes can be developed for network coding, data and speech coding algorithms.
HONORS AND PUBLICATIONS

- **Awarded as the Design Contest Winner 3** at the 22nd IEEE International Conference on VLSI Design, New Delhi, 5-9 January 2009. The paper titled, “Novel Polymorphic Reconfigurable Hardware Support for Discrete Wavelet Transform” was awarded with a memento, certificate of merit and a cash prize of 10,000 INR at the awards distribution banquet.

**International Journal Publications**


**Refereed Book Chapters**


International Conference Publications


Journal papers under Review/ In Preparation


APPENDIX A. VIDEO COMPRESSION BASICS

A video encoder consists of three main functional units: a temporal model, a spatial model and an entropy encoder. The temporal model takes raw multimedia data as input and attempts to reduce temporal redundancy. This involves frame packaging and prediction, motion estimation and compensation operations. The output residual frame is applied to a spatial model (usually the Discrete Cosine Transform (DCT) or Discrete Wavelet Transform (DWT)), where transformed coefficients are quantized to remove insignificant coefficients. Finally, an entropy coder is used to remove any statistical redundancy from the output parameters of the spatial and temporal model.

A.1 Discrete Wavelet Transform (DWT)

The foundations of the DWT go back to 1976 when Croiser, Esteban, and Galand [1] devised a technique to decompose discrete time signals. Crochiere, Weber, and Flanagan did a similar work on coding of speech signals in the same year. They named their analysis scheme as subband coding. In 1983, Burt [17] defined a technique very similar to subband coding and named it pyramidal coding which is also known as multiresolution analysis. Later in 1989, Vetterli and Le Gall [108] made some improvements to the subband coding scheme, removing the existing redundancy in the pyramidal coding scheme. All these schemes are based on Wavelet Transform.

Since then, the Discrete Wavelet Transform (DWT) has emerged as a powerful tool for compression and is being used in many multimedia and signal processing applications.

Prior works in signal processing explain that the 1-D DWT can be viewed as a signal decomposition using specific low pass and high pass filters. A single stage of image decomposition can be implemented by successive horizontal row and vertical column wavelet transforms.

The DWT of a signal $x$ is calculated by passing it through a series of filters. First the samples are
passed through a low pass filter with impulse response $h_0$ resulting in a convolution of the two:

$$y_0[n] = (x \ast h_0)[n] = \sum_{k=-\infty}^{\infty} x[k]h_0[n-k].$$

The signal is also decomposed simultaneously using a high-pass filter $h_1$ to get the output $y_1$. The outputs $y_0$ and $y_1$ give the detail coefficients (from the low-pass filter) and approximation coefficients (from the high-pass filter) as a result of one dimensional DWT operation. A 2-D DWT is obtained using successive row-wise and column-wise 1-D DWT operations.

The two filters ($h_0$ and $h_1$) are known as quadrature mirror filters and related to each other by the following relation:

$$|H_0(e^{j\Omega})|^2 + |H_1(e^{j\Omega})|^2 = 1$$

where $\Omega$ is the frequency, and the sampling rate is normalized to $2\pi$. In other words, the power sum of the high-pass and low-pass filters is equal to 1. The filter responses are symmetric about $\Omega = \pi/2$.

$$|H_0(e^{j\Omega})| = |H_1(e^{j(\pi-\Omega)})|^2$$

The output values after low pass and high pass filtering are now double the count than the original count of input pixels. However, since half the frequencies of the signal have now been removed, half the samples can be discarded according to Nyquist’s rule. The filter outputs are thus subsampled by 2.

$$y_0[n] = \sum_{k=-\infty}^{\infty} x[k]h_0[2n-k]$$

$$y_{1h}[n] = \sum_{k=-\infty}^{\infty} x[k]h_1[2n-k]$$

This decomposition has halved the time resolution since only half of each filter output characterizes the signal. However, each output has half the frequency band of the input so the frequency resolution has been doubled. Thus one level of 2-D DWT operation on an image is represented by filtering with high and low pass filters across row and column successively and is illustrated in figure 2.3.

Applying a 2-D DWT to an image of resolution $M \times N$ results in four images of dimensions
Figure A.1 (a) Resulting subbands after three levels of wavelet decomposition, and (b) Three levels of wavelet decomposition of a sample image

Amongst different entropy-coding methods, and their possible applications in compression applications, arithmetic coding stands out in terms of elegance, effectiveness and versatility, since it is able to work most efficiently in the largest number of circumstances and purposes.

When applied to independent and identically distributed (i.i.d.) sources, the compression of each
symbol is provably optimal. It is effective in a wide range of situations and compression ratios. The same arithmetic coding implementation can effectively code all the diverse data created by the different processes such as transform coefficients, signaling, modeling parameters and raw data.

A.2.0.1 Example

The idea behind arithmetic coding is to have a probability line, 0-1, and assign to every symbol a range in this line based on its probability, the higher the probability, the higher range which assigns to it. Once we have defined the ranges and the probability line, we start encoding symbols, every symbol defines where the output floating point number lands. Consider the following example (the relative occurrence of a, b and c is 3:1:1 respectively):

We start with the full interval [0,1) and depending on the input symbol, we keep reducing the interval successively. Any codeword from the final interval can be selected, and the decoder will be able to decode the coded input. Let us say we want to encode ‘abc’. We first encode ‘a’ to find that the range has shrunk to [0,0.6). Let us say that we select 0.5 and transmit that value. In this case, we find that the decoder will be able to obtain the value ‘a’ using Table A.

Now we re-partition this interval according to source probability:

To encode ‘b’ after ‘a’, the interval shrinks to [0.36,0.48). Now, to encode ‘c’, we re-partition the interval to get the following Table.
Table C

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Probability</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>3</td>
<td>[0.36, 0.432)</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>[0.432, 0.456)</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>[0.456, 0.48)</td>
</tr>
</tbody>
</table>

To transmit the information ‘abc’, one can equivalently transmit a number in the range [0.456,0.48).

Let us send 0.46875 which lies in this interval and is represented in binary using only 5 bits (01111).

The decoder first decodes ‘a’ (0 ≤ 0.46875 < 0.6) and then the decoder also generates Table B. It
decodes ‘b’ (0.36 ≤ 0.468675 < 0.48) and then decodes ‘c’ (0.456 ≤ 0.46875 < 0.48). The decoder
is cognizant of string size and thus it decodes only upto three characters and obtains back the string
‘abc’.
APPENDIX B. MULTIMEDIA SECURITY

B.1 Chaos Theory and Logistic Maps

Chaos theory is a field of study in mathematics, physics, and philosophy studying the behavior of dynamical systems that are highly sensitive to initial conditions. This sensitivity is popularly referred to as the butterfly effect. Small differences in initial conditions (such as those due to rounding errors in numerical computation) yield widely diverging outcomes for chaotic systems.

Chaos used to be treated as stochastic and unpredictable phenomena. Nowadays, this stochastic-like behavior that chaotic oscillations presents, characterized by a large broadband frequency spectrum, has been used to hide information, in order to safely transmit secret messages. It has been used in cryptography to build stream ciphers based on iterations on chaotic maps.

B.1.1 Logistic Map

The logistic map is a polynomial mapping of degree 2, often cited as an archetypal example of how complex, chaotic behaviour can arise from very simple non-linear dynamical equations. The map was popularized in a seminal 1976 paper by the biologist Robert May, in part as a discrete-time demographic model analogous to the logistic equation first created by Pierre François Verhulst.[1] Mathematically, the logistic map is written

\[ x_{n+1} = ax_n(1 - x_n) \]

where:

\( x_n \) is a number between zero and one, and represents the population at year \( n \), and hence \( x_0 \) represents the initial population (at year 0). \( a \) is a positive number, and represents a combined rate for
reproduction and starvation. The

By varying the parameter $a$, the following behavior is observed:

1. With $a$ between 0 and 1, the population will eventually die, independent of the initial population.

2. With $a$ between 1 and 2, the population will quickly stabilize on the value $r$, independent of the initial population.

3. With $a$ between 2 and 3, the population will also eventually stabilize on the same value, but first oscillates around that value for some time. The rate of convergence is linear, except for $a=3$, when it is dramatically slow, less than linear.

4. With $a$ between 3 and (approximately 3.45), the population may oscillate between two values forever. These two values are dependent on $r$.

5. With $a$ between 3.45 and 3.54 (approximately), the population may oscillate between four values forever. With $a$ increasing beyond 3.54, the population will probably oscillate between 8 values, then 16, 32, etc.

6. At $a$ approximately 3.57 is the onset of chaos, at the end of the period-doubling cascade. We can no longer see any oscillations. Slight variations in the initial population yield dramatically different results over time, a prime characteristic of chaos.

7. Most values beyond 3.57 exhibit chaotic behavior, but there are still certain isolated values of $r$ that appear to show non-chaotic behavior; these are sometimes called islands of stability. For instance, beginning at (approximately 3.83) there is a range of parameters $a$ which show oscillation between three values, and for slightly higher values of $r$ oscillation between 6 values, then 12 etc.

8. Beyond $a = 4$, the values eventually leave the interval $[0,1]$ and diverge for almost all initial values.
B.1.2 Chaos and Logistic Map

The relative simplicity of the logistic map makes it an excellent point of entry into a consideration of the concept of chaos. A rough description of chaos is that chaotic systems exhibit a great sensitivity to initial conditions – a property of the logistic map for most values of $r$ between about 3.57 and 4 (as noted above). A common source of such sensitivity to initial conditions is that the map represents a repeated folding and stretching of the space on which it is defined. In the case of the logistic map, the quadratic difference equation (1) describing it may be thought of as a stretching-and-folding operation on the interval $(0,1)$.

B.2 Multimedia Encryption

The high redundancy, large volumes, real-time operations, and transcoding of multimedia data require that the multimedia encryption schemes should satisfy certain requirements. For example, since the multimedia data is highly redundant, it may be not safe to encrypt data with a traditional cipher. Moreover, there would be very large computational cost involved in such operation. Furthermore, the relation between encryption and compression should be investigated in order to avoid changes in compression ratio. Moreover, certain real-time applications such as mobile TV, remote surveillance etc require the encryption operation to be efficient enough to avoid service delay.

Some requirements of multimedia encryption schemes are discussed below in the following subsections:

Security is the basic requirement of multimedia content encryption. Multimedia data requires both perceptual security and cryptographic security i.e. the scheme must be unintelligible to human perception and also secure against cryptographic attacks. For some time-crucial applications, the encryption scheme may be regarded as secure if the cost of breaking it is no smaller than the significance of multimedia data. For example- some surveillance information may be of no use after one hour. Then the encryption operation may be regarded as secure if the attacker can not break the encryption algorithm over the course of an hour.

Cryptographic security is determined by the ability to resist the cryptanalysis methods, including such attacks as differential analysis, related-key attack, and statistical attack. Some metrics used to
measure the cipher’s resistance to such attacks are key sensitivity, plain text sensitivity or cipher text randomness.

Key sensitivity is a measure of the changes in cipher text’s changes with change in the change in encryption key. If \( CT_1 \) and \( CT_2 \) are the two encrypted outputs obtained after encryption of plain text image \( PT_1 \) by keys \( K_1 \) and \( K_2 \) (differing by 1 bit only), we can define key sensitivity by the following relationship:

\[
KS = \frac{\sum_{k=1}^{M} \sum_{l=1}^{N} CT_{2(k,l)} \oplus CT_{1(k,l)}}{M \times N} \times 100\%
\]  

(B.1)

Plain text sensitivity can be similarly measured by the change in encryption output bits with change in plain text \( PT_1 \) and \( PT_2 \) and keeping the key unchanged (\( K_0 \)).

\[
PS = \frac{\sum_{k=1}^{M} \sum_{l=1}^{N} CT_{2(k,l)} \oplus CT_{1(k,l)}}{M \times N} \times 100\%
\]  

(B.2)

Cipher text randomness can be measured by counting the value of encrypted bit stream. The histogram of the output bits must approach a random distribution.

Perceptual Security is determined by the intelligibility of the cipher text to the observer. The typical metric for image quality is Peak Signal to Noise Ratio (PSNR). PSNR is initially used to measure images quality losses caused by such operations as compression, noising, transmission errors, etc. It is computed by comparing the original image and the operated image. If \( PT_1 \) is the unencrypted image and \( CT_1 \) is the encrypted image, PSNR is defined as follows:

\[
PSNR = 10 \log_{10} \left( \frac{255^2}{\frac{1}{MN} \sum_{k=1}^{M} \sum_{l=1}^{N} (CT_{1(k,l)} - PT_{1(k,l)})^2} \right)
\]  

(B.3)

Multimedia encryption may be applied to multimedia data before compression, during compression or after compression, depending on the applications. However, in all cases, multimedia encryption algorithms should not change the compression ratio or should at least keep the changes in a small
range. The Changed Compression Ratio is defined as

\[ CCR = \frac{R_1 - R_0}{R_0} \times 100\% \]  

(B.4)

where \( R_1 \) is the data rate with encryption, \( R_0 \) is the encryption rate without encryption.

The multimedia encryption algorithms should be efficient so that they don’t delay the transmission or access operations in real-time scenarios. Generally, two kinds of methods are traditionally adopted to alleviate the computational burden of encryption schemes to ensure the real-time promise: the first is to reduce the encrypted data volume, and the other is to adopt lightweight encryption algorithms.

Partial encryption is the algorithm that encrypts only a part of the multimedia content while leaving other parts unchanged. This scheme is summarized in Figure B.1. The volume of encrypted data is reduced at the cost of reduced security in this scheme. In practice, the more crucial information in video decompression such as I-frame information [87] is encrypted while other information is transmitted without any encryption. An intelligent data splitting operation is crucial to provide reasonable security using partial encryption schemes.

Thus, a study of related works highlights the need to develop efficient algorithms and architectures for ensuring multimedia security that cater to the requirements of embedded real-time systems.
Figure B.1 An overview of partial encryption scheme
BIBLIOGRAPHY


