Embedded Multimedia Security Systems

Amit Pande • Joseph Zambreno

Embedded Multimedia Security Systems

Algorithms and Architectures

Springer

Amit Pande
Department of Computer Science
University of California Davis
Davis CA, USA

Joseph Zambreno
Electrical and Computer Engineering
Iowa State University
Ames IA, USA

ISBN 978-1-4471-4458-8 ISBN 978-1-4471-4459-5 (eBook)
DOI 10.1007/978-1-4471-4459-5
Springer London Heidelberg New York Dordrecht

Library of Congress Control Number: 2012948199

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Dedicated to our teachers
. . . who taught us everything in life.

Foreword

Multimedia technology has become a powerful force in modern society. Mobile videos generated from mobile devices such as smartphones and wireless sensors are increasing manifold in demand. Power consumption, secrecy and privacy of multimedia content is becoming an increasing important concern for the industry and academia. In this context, I feel privileged to introduce Embedded Multimedia Security Systems: Algorithms and Architectures, a book written to give hands-on understanding of multimedia coding and encryption problem to researchers.

The first three chapters form Part I of the book. They give a foundation to the readers about multimedia coding, encryption and architecture issues. After a brief overview of the existing approaches to secure multimedia content, the authors introduce the notion of developing algorithms and architectures for joint compression and encryption of multimedia data. The later four chapters, forming the Part II of book, give practical examples, design considerations, derivation of building such building blocks for joint compression and encryption issues. These chapters give a flavor to research in this area and invite researchers to delve into and come up with new solutions in this area.

I believe that this new book establishes a skill base for those who would wish to practice the subject (secure embedded multimedia systems) and become seriously involved in the design and creation of useful and effective multimedia applications for solution of real world problems in a variety of different contexts. Moreover, it establishes credence in terms of its usefulness, accuracy and the reliability of the work. The book provides a wealth of knowledge for beginners as well as practitioners.

Prasant Mohapatra

Professor
Tim Bucher Family Endowed Chair
Department of Computer Science
University of California, Davis

Preface

Welcome to the first edition of the book “Embedded Multimedia Security Systems: Algorithms and Architectures”. The main objective behind this book is NOT to serve as a textbook or a book which gives readers a thorough understanding of security approaches in embedded multimedia systems. Rather, our goal is to stimulate creative thinking in the readers to come up with newer schemes for video encryption problems.

This book gives the perspective that architecture, coding (compression) and encryption can be viewed together. Instead of designing a compression and an encryption algorithm separately and then fitting it into a target architecture, we can make a complete design altogether. Some food for thought is given to the readers in the form of research articles in this book, but we hope that the readers will take the frontiers beyond what has been presented here.

Part one of this book gives a brief introduction to the security and encryption approaches in existing literature, as well as motivation of the joint approach for compression and encryption. The second example gives more concrete examples of how to do this, in a modular way, by augmenting encryption to video compression blocks. Polymorphic Wavelet Transform discusses a hardware architecture which can be modified at run-time to suit the needs of application and requires much less computing resources, all with the help of novel signal processing approach designed to suit the hardware requirements. Secure Wavelet Transform is an architecture that uses wavelet transform stage for encryption, and then optimizes the implementation for hardware. Chaotic Filter Banks introduce the idea of chaotic maps to wavelet filter banks for encryption. Chaotic Arithmetic Coding is presented as an interpretation of chaotic maps to replace the existing arithmetic coding scheme.

Intended Audience

This book is suitable for advanced undergraduates and first-year graduate students in computer science, computer engineering and electrical engineering majors, and for students in other disciplines who are interested to study the marriage of multimedia (video) coding, encryption and hardware implementation. The book will also be useful for many professionals, like software firmware/algorithm engineers, hardware engineers, chip and system architects, technical marketing professionals and researchers in multimedia, communication, security, semiconductor, and computer industries.

Davis, CA, USA Amit Pande
Ames, IA, USA Joseph Zambreno

Acknowledgements

The authors would like to thank the publishers, the many friends and family members whose cooperation has been instrumental in bringing forth this book.

Contents

Acronyms

AES	Advanced Encryption Standard (data encryption algorithm)
ASIC	Application-Specific Integrated Circuit
BWFB	Bi-orthogonal Wavelet Filter Bank
CAC	Chaotic Arithmetic Coding
DCT	Discrete Cosine Transform (frequency domain representation of signal)
DES	Data Encryption Standard (data encryption algorithm)
DWT	Discrete Wavelet Transform (frequency domain representation of signal)
EBCOT	Embedded Block Coding with Optimized Truncation
FPGA	Field Programmable Gate Array
JPEG	Joint Photographic Expert Group (image coding standard)
JPEG2000	Joint Photographic Expert Group (image coding standard based on DWT, created in 2000)
LUT	Look-Up Table
LHBF	Lagrange Half-Band Filter
MHT	Multiple Huffman Table
MCFB	Modified Chaotic Filter Bank
MLM	Modified Logistic Map
MPEG	Moving Pictures Expert Group (video coding standard)
Poly-DWT	Polymorphic Discrete Wavelet Transform
PSNR	Peak Signal-to-Noise Ratio
RCM	Reconfigurable Constant Multiplier
SAC	Secure Arithmetic Coding
SPIHT	Set Partitioning in Hierarchical Trees
SRTP	Secure Real-time Transport Protocol
SWT	Secure Wavelet Transform

Part I
Multimedia Systems

This is part one of the book. It intends to give the reader overall directions in multimedia delivery problem and a strong grasp in basic concepts in coding, communication, security, and architectures for video coding—together referred to as "Multimedia Systems".

Chapter 1
Introduction

Abstract In the past few years, multimedia communication has gained significant attention, particularly over mobile and embedded devices (such as wireless sensors). Multimedia applications are data-intensive, therefore requiring special treatment as against common data coding, communication and encryption applications. This section discusses some basics about video coding, data encryption and computer architectures used for data processing.

Chapter goals:

- Basic introduction to video coding techniques.
- Basic introduction to video encryption techniques.
- Basic introduction to 'general-purpose' and 'custom' computer architectures.

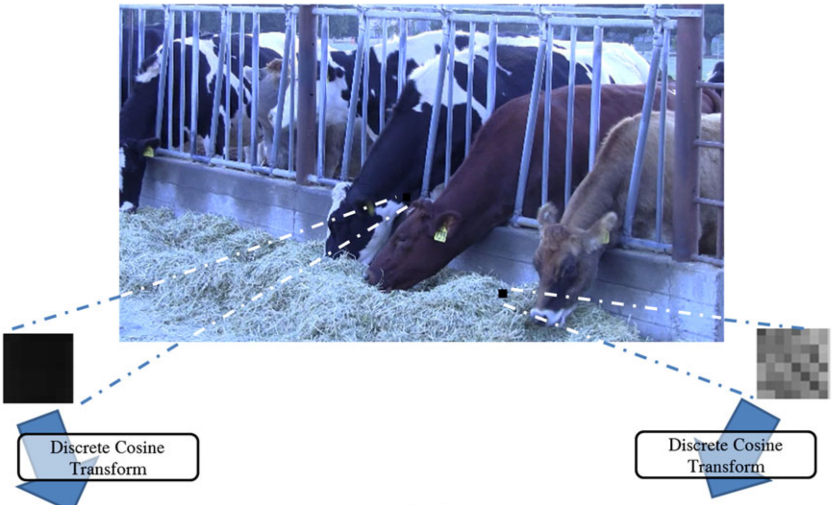
1.1 Video Coding

Digital video takes up a very large amount of storage space or bandwidth in its original, uncompressed form. Video compression makes it possible to send or store digital video in a smaller, compressed form. The source video is compressed or encoded before transmission or storage. Compressed video is decompressed or decoded before displaying it to the end user. Consider a HD video at resolution 1920 × 1080 pixels at 60 frames per second. The three colors (red, green and blue) are quantized at 8 bits per pixel. The size of one second of video is equal to 1920 × 1080 × 60 × 3 × 8 bits or roughly 2.8 Gigabits per second. Transmission of such file size is almost impossible on the best home networks (LTE-A or WiFi). Thus, the volume of data has to be greatly compressed.

A CODEC has two parts: COder and DECoder. The COder is used to compress or code large data into fewer bits, using a reversible conversion. It allows efficient storage and transmission of data. The inverse process is called decompression (DECoding). Video compression or the process of compressing video signals has many stages. There are many video codecs which use a variety of stages and their variants, however, their main functions are similar.

The first step in image or video compression is frequency-domain transformation. A frequency-domain transformation is a necessary step because natural images can be compactly represented in frequency domain. In other words, it takes less words

A. Pande, J. Zambreno, *Embedded Multimedia Security Systems*,
DOI 10.1007/978-1-4471-4459-5_1, © Springer-Verlag London 2013



0.558824	0.046556	0.05045	0.003783	0.02549	-0.00808	0.001388	0.000264
-0.0283	0.007398	-0.00146	0.008132	-0.00073	-0.0002	0.002409	-0.00331
-0.00481	0.000167	0.006407	-0.0027	-0.0095	0.000378	-0.00404	0.00188
0.002057	0.007195	-0.00914	0.000741	-0.00613	-0.00218	-0.00391	-0.00084
-0.00784	0.00714	0.001592	0.000679	-0.00784	-0.00308	-0.00384	-0.00158
-0.00509	0.003444	-0.00111	0.000591	-0.00361	-0.00098	-0.00107	-0.00048
-0.0065	0.002443	-0.00012	0.001482	-0.00243	8.78E-05	-0.00053	-0.00028
0.005852	-0.00053	-0.00083	-0.00192	-0.00256	-0.00036	-0.00236	0.000683

3.896569	-0.19977	0.184919	-0.03101	0.030882	-0.06264	0.017318
0.093513	-0.3037	-0.04789	-0.10458	-0.08909	-0.03845	-0.05888
-0.04414	-0.14098	0.008405	0.115268	-0.11869	0.017876	-0.11972
0.024842	-0.00019	-0.00448	-0.09499	-0.00839	0.19905	-0.06001
0.026961	0.075704	0.034245	0.044743	-0.25833	0.054225	-0.02108
-0.12781	-0.08887	-0.06453	-0.07474	0.102632	-0.24699	0.116567
0.030488	0.033169	0.127342	-0.1143	0.083651	0.029814	-0.07409
-0.06989	0.076474	-0.05459	-0.01535	-0.0645	0.136597	0.056959

Fig. 1.1 Operation of Discrete Cosine Transform on 8 × 8 blocks of an image. The texture regions have more frequency components while the regular background (like cow skin) has less frequency components. Implementing DWT

to describe an image in frequency domain than in spatial domain (see Fig. 1.1). The most frequent operation for such a task is called Fourier Transform or Cosine Transform or Discrete Cosine Transform (DCT). These transforms are not the same—there are some differences between them but the basic idea is conversion of coefficients from time-domain to transform domain. There are also K-L transform, Hadamard transform etc. The most popular of them is DCT, used in most image and video compression algorithms. It maps real coefficients into real coefficients. The low frequency terms have most visual information while the high frequency terms are nearly zero and can be discarded to achieve compression. In lossy compression schemes such as JPEG, MPEG and related codecs, the coefficients are quantized.

Applying DCT on an entire image or frame is a memory intensive task. Most schemes apply DCT on small blocks of size 8 × 8 pixels or 16 × 16 pixels, called macroblocks.

Another popular frequency-domain transformation is called wavelet transform. Instead of waves (sine/cosine or exponential waveforms used in DCT-based transforms, ranging from $-\infty$ to $+\infty$) these transforms use wavelets or finite length waveforms, which give a time-frequency resolution. The Discrete Wavelet Transform (or DWT) can also be implemented using a set of filters (comprising a filter bank). A low and a high pass filter forms a wavelet filter bank. Some additional

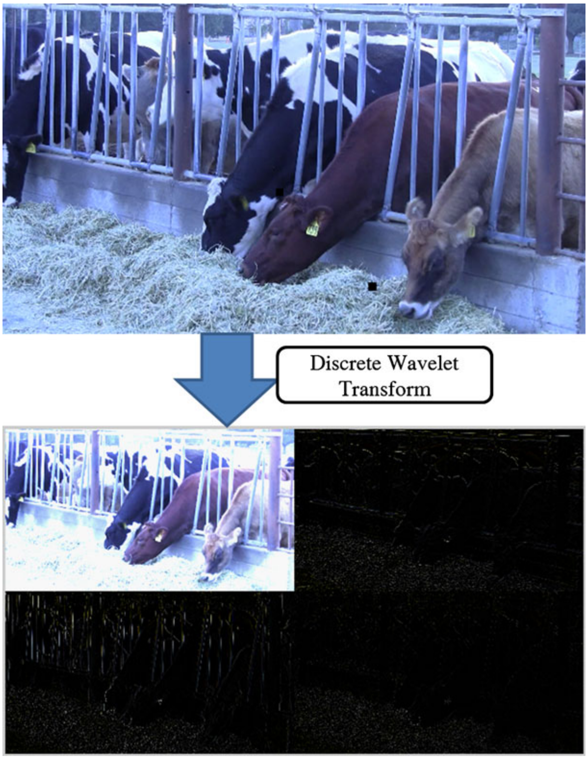


Fig. 1.2 One level decomposition of sample image using Discrete Wavelet Transform. The first sub-band has maximum information while others have higher frequency terms

properties of common wavelet filters are perfect reconstruction, bi-orthogonality, and energy separation between low and high pass coefficients. The exciting property of wavelet transform is the property of time-frequency resolution as against frequency resolution provided by DCT. It is helpful and forms the basis of multi-resolution analysis (see Fig. 1.2). The earliest wavelet-based codecs such as EZW and SPIHT use this property to achieve scalability.

The frequency-domain coefficients are quantized followed by a entropy coding technique. Many entropy coding techniques exist in literature, and they are used to lossly squeeze the output bits to maximize entropy of output bitstream, making them close to the Shannon optimality bound on information contained in a given bitstream.

Arithmetic coding is the most efficient of those techniques, reaching Shannon optimality for large strings. However, it is associated with high computational complexity and several variants such as Q coder, QM coder etc. exist. Another popular scheme is Huffman coding which is used to generate variable length codes. Huffman coding is less optimal than arithmetic coding but implementation efficiency is much higher than arithmetic coding. It is therefore used in most implementations of JPEG and MPEG codecs.

Prior to frequency-domain transformation, in the case of video coding, some degree of temporal coding is used to reduce redundancy amongst frames. With DWT,

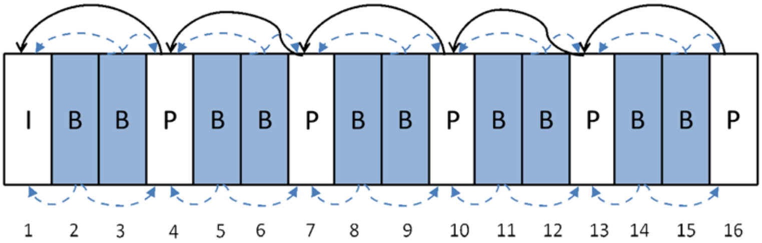


Fig. 1.3 Consecutive frames of a video are grouped into chunks called GOP or Group of Pictures. The first frame in each GOP is called I frame (intra-coded). P (Predictively coded) and B (Bi-directionally coded) frames constitute remaining frames of a GOP

such treatment can be done after transform also. However, in most DCT-based and other schemes, this task is done prior to spatial coding. Video is grouped into sets called a Group-Of-Pictures (GOP), see Fig. 1.3. The first frame in each GOP is intra-coded and called I-frame. In an I frame, prior to DCT, each macroblock is matched to its neighbors for similarity, to reduce redundancy. The difference (or residue) is then coded. Some frames are labeled as P frames and they are predictively coded. The macroblocks in P frames are coded based on past P or I frame. Some frames are optionally labeled as B frames and they are bi-directionally coded. This implies that each B frame is coded based on 1 previous and 1 next P or I frame. While searching for a match for a macroblock, the search is conducted in the vicinity of existing macroblock location in F_α. This process is also known as motion compensation.

1.2 Embedded Systems and Reconfigurable Architectures

Embedded systems refer to all computing devices beyond the traditional desktop and laptop-based computers. It includes most of the electronics devices and other things we use in our day-today life. Printers, cars, home security systems, mobile phones, digital cameras, washing machines etc. It may serve as a part of a larger unit and provides dedicated service to that unit. It is hard to precisely give a definition of embedded systems. Nearly any computing system other than a desktop computer can be classified as a embedded system. Billions of such units are produced yearly and hundreds are used per household.

Some common properties and generalities of embedded systems are enumerated as follows: They are usually single functioned—perform a single task repeatedly. They are tightly constrained in cost and power, size etc. In our context, we consider examples of digital camera, surveillance cameras and smart phones (have similar constraints although they have a more general purpose by nature).

Such system have tighter resource constraints than desktops or commodity computing machines used for image and video processing. For example—one has to consider the battery requirements, computational capabilities, memory constraints, latency of the process, thermal specifications and so on. Hardware and software

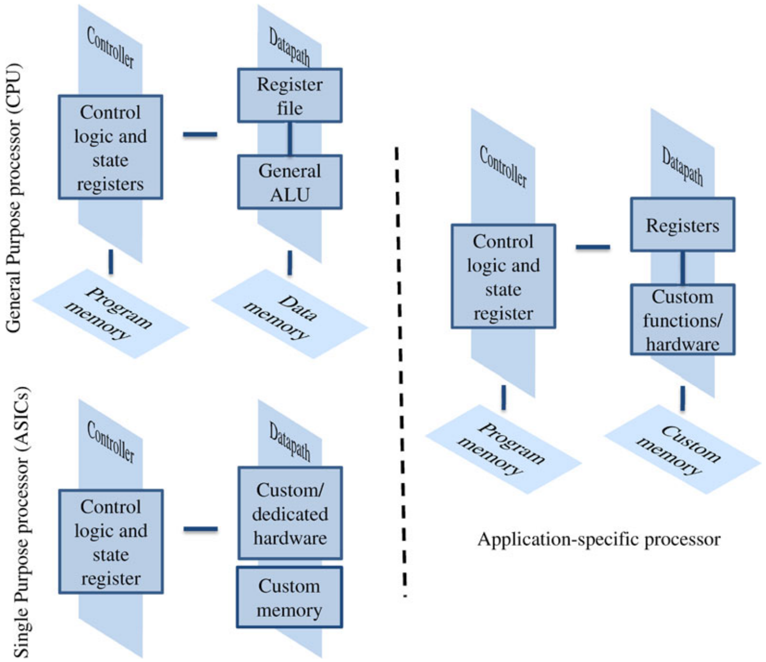

General Purpose processor (CPU)
Controller
Control logic and state registers
Datapath
Register file
General ALU
Program memory
Data memory
Single Purpose processor (ASICs)
Controller
Control logic and state register
Datapath
Custom/ dedicated hardware
Custom memory
Controller
Control logic and state register
Datapath
Registers
Custom functions/ hardware
Program memory
Custom memory
Application-specific processor

Fig. 1.4 Overview of different processor technologies used in embedded systems

must be co-designed in such scenarios to yield an efficient implementation. For example—many stages of video processing pipeline can be hard coded into VLSI chips to achieve a order of hundred speed-up over software implementations.

Processor technology refers to the architecture of the computation engine used to implement a system's desired functionality. It need not be programmable. A simple classification of processor technologies used in embedded systems is given in Fig. 1.4. A general purpose processor, also known as a microprocessor, has a program memory and a general datapath with large register files and general ALU. Although it has low time to market and NRE costs it is not very suitable for dedicated video processing devices. Desktop processors (such as the ones by Intel and AMD) fall under this category. There are application-specific processors, such as DSP or Digital Signal Processing cores, which have special functional units to optimize application-specific tasks. The fastest execution speeds, the least memory overhead, latency and power requirements are obtained by using single-purpose or dedicated hardware for tasks such as video processing. Digital cameras and smart phones mostly have a dedicated video processing core that efficiently handles such operations. Such ASIC implementations take time to ship, up to 12–24 months, and it is possible that the video codec implemented has reached maturity by that. It is quite possible that better codecs have been developed in the meantime. To mitigate such problems, and to show proof-of-concept implementations on ASIC, researchers conveniently switch to what is commonly known as FPGA.

FPGA or Field-Programmable Gate Arrays are custom hardware, which can be programmed to behave the way we like. Unlike a VLSI chip, the interconnects between computing elements and their functionalities can be dynamically defined, making it possible to implement a design on FPGA in few days, which would take months in ASICs. Moreover, modern FPGAs feature dynamic reconfiguration, which makes it possible to re-configure the hardware resources on-the-chip in real-time. Thus, FPGAs have the following uses, particularly for video processing applications:

1. Short time to implement the architecture on FPGA, which can serve as a proof-of-concept for a later-date ASIC implementation. It gives significant acceleration in run-time over microprocessors-based design.
2. Short time to implement architecture on FPGA and FPGA deployment in real-world scenarios allows for quick re-insertion of new modules, as and when they are developed. This is not possible for ASICs, where we have to design a new chip altogether.
3. Run-time reconfiguration on FPGAs can be helpful for power-efficient designs.

In this book, most prototyping of proposed architectures is done on FPGAs.

1.3 Encryption Basics

The method of disguising plaintext or regular data in a manner that its substance is hidden is called encryption. Encrypting plaintext results in unreadable gibberish called ciphertext. The unintended users can be avoided in this manner, even though they may see the cipher text. Once encrypted, no person (or machine) can discern anything about the content of the original data by reading the encrypted form of the data. The intended users can decrypt the ciphertext to get the plaintext back. Strong cryptography generally indicates the extreme difficulty of decoding back the plaintext by unintended user, even with use of best computers and resources. Such cryptography is required for security of important documents such as credit card details of a person or communications of federal government. Algorithms such as AES and DSA are used for such operations.

In the case of videos, strong cryptography is not always required. In popular consumer applications, the cost of purchasing a movie may be, say, \$5.00. An attacker will not mount attacks worth millions \$ to hack such videos. Similarly, if it takes one hour to hack an encrypted video but the video is a news feed, the provided security may be good enough. Therefore, application-specific, partial or perceptual cryptography can play a significant role in case of videos or other less sensitive information. Moreover, videos are associated with high data volumes and there is ongoing effort to reduce the latency and computational cost of strong cryptography algorithms for videos.

In encryption, some operation is performed into plaintext using a key stream which converts it into the ciphertext. The key stream is a common secret to sender and receiver. An unintended recipient has no knowledge of the key, therefore he cannot decrypt the content back to plaintext. In symmetric key encryption, the sender and receiver use the same key for encryption and decryption. Both the sender and receiver have to keep the key secret and properly protected. Asymmetric key encryption algorithms are also called public key algorithms. The main difference is that the sender and receiver share different keys to lock/unlock the encrypt/decrypt the data. For example, anyone can encrypt a message intended for A with his public key while only A, who has prior information of his private key alone can decrypt the data. Typically, asymmetric key encryption is more difficult than symmetric key schemes and requires more computational steps. Our focus here is on developing symmetric key encryption algorithms combined with compression algorithms for videos, as described in future sections.

Chapter 2
Advances in Multimedia Encryption

Abstract Rapid advances in embedded systems and mobile communications have flooded the market with a large volume of multimedia data. In this chapter, we present a summary of multimedia compression and encryption schemes, and the way they have evolved over the decades.

Chapter goals:

- Familiarize with existing deal of research approaches in the area of multimedia encryption.
- Summary of key/popular schemes proposed in research literature.

2.1 Introduction

Security is becoming an escalating concern in an increasingly multimedia defined world. The recent emergence of embedded multimedia applications such as mobile-TV, video messaging, and telemedicine have increased the impact of multimedia and its security on our personal lives. For example, a significant increase in the application of distributed video surveillance technology to monitor traffic and public places has raised concerns regarding the privacy and security of the targeted subjects.

Multimedia content encryption has attracted more and more researchers and engineers owing to the challenging nature of the problem and its interdisciplinary nature in light of challenges faced with the requirements of multimedia communications, multimedia retrieval, multimedia compression and hardware resource usage.

With the continuing development of network communications (wired and wireless), easily capturing videos and rapid advances in Internet technology and embedded computing systems multimedia data (images, videos, audios, etc.) are of importance for use more and more widely, in applications such as video-on-demand, video conferencing, broadcasting, etc. Now, it is closely related to many aspects of daily life, including education, commerce, defense, entertainment and politics. In order to maintain privacy or security, sensitive data need to be protected before transmission or distribution. The advancements in ubiquitous network environment, and rapid developments in cloud computing have promoted the rapid delivery of digital multimedia data to the users.

A. Pande, J. Zambreno, *Embedded Multimedia Security Systems*,
DOI 10.1007/978-1-4471-4459-5_2, © Springer-Verlag London 2013

Users are eager to not only enjoy the convenience of real-time video streaming but also share various media information in a rather cheap way without awareness of possibly violating copyrights. In view of these, encryption and watermarking technologies have been recognized as a helpful way in dealing with the copyright protection problem in the past decade. Encryption allows secure end–end communication of data while digital watermarking allowing still faces some challenging difficulties for practical uses; there are no other techniques that are ready to substitute it.

Within the signal processing and multimedia communities, many schemes have been proposed for protecting sensitive information while allowing certain legitimate operations to be performed. These schemes typically lack a rigorous model of privacy, and their protection becomes questionable when scaled to large datasets. The cryptography community has long developed rigorous privacy models and provably secure procedures for data manipulations. However, these procedures are primarily designed for generic data. As a result, they usually lead to a blow up in computational costs and overheads when applied to real-life multimedia applications.

There has been a great deal of effort to design algorithms and architectures for multimedia security (particularly encryption) suitable for mobile and embedded devices which have tighter constraints on computational resources.

2.2 Multimedia Encryption Problem

Multimedia encryption involves changing the multimedia datastream itself to ensure secure transmission of video data between client and server (or two nodes). It can be accomplished by means of standard symmetric key cryptography where multimedia bitstream is treated as a binary sequence and the whole data can be encrypted using conventional cryptosystem such as AES or DES [18].

In general, when the application requirements are not dynamic (not a real-time streaming) we can treat bitstream as a regular binary data stream and use the conventional encryption techniques. Encrypting the entire multimedia stream using standard encryption methods is referred to as the naive algorithm. There are many practical constraints in case of mobile multimedia which make such a scheme not practical in real-life scenario. First there are issues with available computational resources in mobile devices which combined with low battery life and limited device area limit the application of AES or DES like ciphers. Unlike desktop processors, dedicated AES co-processor will cause high power and area requirements. This can be understood with the example of GSM mobile phones which use a much lighter cryptographic cipher for data encryption. A5 is the stream cipher used to provide over-the-air communication privacy in the GSM cellular telephone standard and is used in various variants. A5/0 utilizes no encryption while A5/1 is the original A5 algorithm used in Europe. A deliberate weakening of the algorithm was proposed as A5/2, but it was cryptanalyzed the same month as it was published. The A5 algorithm is much simpler in implementation than AES, and is implemented using stream ciphers. A5/3, also known as KASUMI is a stronger encryption algorithm created as part of the 3rd Generation partnership Project (3GPP). The Secure Real-time Transport Protocol, or for short SRTP [**22RFC3711??**], is also an application of naive approach, where multimedia data are 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.

Communication encryption of multimedia content is a problem beyond the application of established encryption algorithms, such as DES or AES, to its binary sequence [20]. This is primarily due to the way multimedia is used commercially. Unlike data encryption, where we want to encrypt a complete bitstream, mobile multimedia encryption introduces several challenges. Firstly, the content providers want to ensure real-time streaming of videos. The mobile phone users will not wait for authentication and encryption of downloaded videos if they need to wait for long times. Real-time streaming of secure bitstream is a serious challenge for mobile multimedia delivery, because the wireless environment (in which mobile phones are operating) already pose serious bandwidth restrictions. First of all, the user may search (in real time) for a particular video at run-time from a digital library. Further, video compression is done in a scalable way, to allow a single compressed copy at server to be downloaded at multiple bit rates. Transcoding may be required at times. We need encryption schemes which can maintain format compatibility and not slow any of these operations.

Further, it involves careful analysis to determine and identify the optimal encryption method when dealing with audio and video media. To identify an optimal security level, we have to carefully compare the cost of the multimedia information to be protected and the cost of the protection itself. If the multimedia to be protected is not that valuable in the first place, it is sufficient to choose relatively light levels of encryption. On the other hand, if the multimedia content is highly valuable or represents a government or military secrets, the cryptographic security level must be the highest possible. For many real-world applications such as pay-per-view, the content data rate should be very high, but the monetary value of the content may not be high at all. Thus, very expensive attacks are not attractive to adversaries, and light encryption may be sufficient for distributing these videos. On the other hand, applications such as video-conferencing or videophone require much higher level of confidentiality. Maintaining such high level of security and still keeping a real-time and limited-bandwidth constraints is not easy to accomplish.

2.3 Common Approaches to Video Encryption

2.3.1 Scrambling

Scrambling is one of the simplest form of encryption that can be applied to multimedia data. It usually refers to encryption methods which perform random permutations to video data using some scheme. The histogram of image generally remains the same except for the fact that the individual positions are shuffled. Early work on signal scrambling was based on using an analog device to permute the signal in the time domain or distort the signal in the frequency domain by applying filter banks or frequency converters [22]. However, these schemes are extremely easy to crack using modern computers. With the popularization of DSP (Digital Signal Processing), in the digital signal domain focus was placed on scrambling in the domain of orthogonal transforms (DFT, DCT, wavelet transform, Hadamard transform, etc.) [22]. The security provided by scrambling alone is low. It also decreases the compression efficiency of video bitstream leading to compression losses and increased size of video file.

Scrambling is often used as an easy way to encrypt live analog/digital video signals such as surveillance camera feeds where heavy ciphers are ruled out because of computational delay. Some most common techniques include:

1. **Line Inversion Video scrambling**: In this method whole or some parts of the signal scan lines are inverted. This scheme is relatively cheap and simple to implement but the security level achieved is low.
2. **Sync Suppression Video scrambling**: The horizontal/vertical line syncs are hidden or entirely removed in this method. This provides a low-cost solution to Encryption and provides good quality video decoding. A typical disadvantage is that the level of obscurity reached by this scheme depends on video content.
3. **Line Shuffle Video scrambling**: In this scheme each signal line is re-ordered on the screen. Although this scheme provides reasonable security, it requires a lot of storage to re-order the screen.
4. **Cut and Rotate Video scrambling**: In this method, each scan line is cut into pieces and then re-assembled in a permuted manner. This scheme provides a compatible video signal, gives an excellent amount of obscurity and good decode quality and stability. However, it requires specialized scrambling equipments.

Compression algorithms have been designed for the unscrambled signals and they use the statistical characteristics of raw data. Once the signal is scrambled, these characteristics will change and the performance of the compression filter will be degraded.

In Zig-Zag permutation [19], instead of mapping the 8 × 8 block (used in DCT compression stage of video coder) to 1 × 64 vector in “Zig-Zag” order, it maps individual 8 × 8 block to 1 × 64 vector by using a random permutation list (secret key). This algorithm consists of three steps.

1. Generation of a permutation list with cardinality 64.
2. Splitting of coefficients according to permutation list, and passing the result to the entropy coding procedure.

However, this method decreases the video compression rate because the random permutation distorts the probability distribution of Discrete Cosine Transform (DCT) coefficients.

A digital image-scrambling scheme should have a relatively simple implementation, amenable to low-cost decoding and low-delay operation for real-time interactive applications. IT should be independent of compression algorithm and should

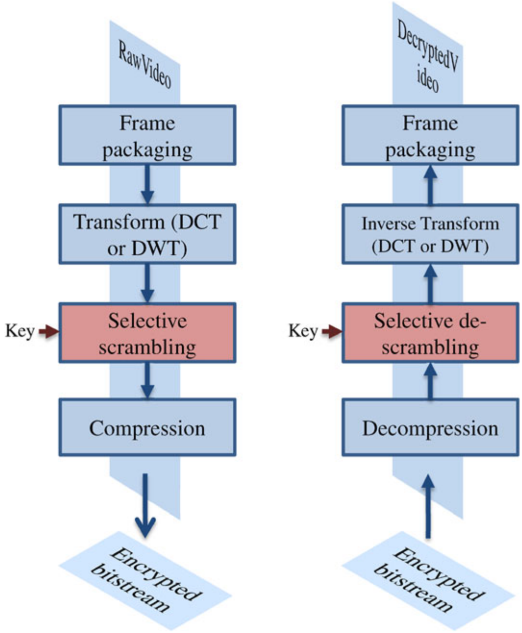


Fig. 2.1 Joint scrambling and compression framework proposed by Zeng and Liu [23]

not incur any loss to the compression operation. We present a case study of the technique presented by Zeng and Liu [23] to better understand the scrambling operations. An overview of their approach is presented in Fig. 2.1.

The authors first transform the input signal into the frequency domain using Discrete Cosine Transform (DCT) or Discrete Wavelet Transform (DWT). The transform coefficients are then divided into subsequent operations which permute their values within the image. The motion vectors are also subject to random sign changes and shuffling. A cryptographic key will be used to control the scrambling. The scrambled coefficients and motion vectors are then passed through compression block to obtain compressed bitstream. Authorized users can obtain the original content back using the same key. The scrambling operation is performed prior to compression allowing preservation of multimedia-specific compression properties such as transcoding and scalability. Frequency domain scrambling makes it easier to control transparency (i.e., what part of the video data is allowed to be freely accessed).

The encryption/decryption operations are designed to preserve, as much as possible, the transformed image properties that allow entropy coders to efficiently compress an image.

Aside from easy and secure transcoding, the joint scrambling-compression framework proposed by [23] provides some other advantages over those that perform scrambling on the compressed bitstreams.

1. **Flexibility to perform selective encryption**: In the frequency domain, it is easier to identify what parts of the data are critical for security purpose. This allows providing different levels of security and transparency.

2. **Encrypting incompressible segments**: It is also easy to identify what parts of the data are not compressible. For example, the sign bits of the coefficients are usually difficult to compress; yet they are critical for security purposes. This incompressible data segment can be selected to scramble without affecting the overall compression efficiency. Some other data segments, such as the motion vector information, are usually losslessly compressed. They therefore can be selected to encrypt without the need to consider the transcoding issue, since it does not make much sense for the transcoder to recode this part of the compressed data. Notice that the selected data can be easily located in the frequency domain without incurring any processing overhead. On the contrary, since the compressed bitstream is usually variable length coded, it is generally difficult to perform fine-scale selective encryption on the compressed bitstream without incurring processing and bit overheads.
3. **Less vulnerability to channel errors**: Encryption after compression such as using AES over MPEG is more vulnerable to channel errors because a block of 128 bits in AES are bound together so that one single bit error in a block will cause the synchronization word/bits contained in that block to be erroneous. Since the synchronization information is hidden in the encrypted video stream; it will be harder to recover from transmission errors in the network. On the other hand, spatial scrambling in the frequency domain has no adverse impact on the error resiliency.
4. **Compatibility to transform domain signal processing**: Scrambling involves changing the spatial positions of individual frequency coefficients. Watermarking and other transform domain tasks can be performed without requiring cryptographic key.

Some of the techniques for scrambling by the author are as follows:

1. *Selective Bit Scrambling*: The first basic approach scrambles selected bits in the transform coefficients to encrypt an image. Each bit of a coefficient can be viewed as one of three types. Significance bits for a coefficient are the most significant bit with a value of 1, and any preceding bits with a value of 0. These bits limit the magnitude of the coefficient to a known range. Refinement bits are the remaining magnitude bits, used to refine the coefficient within the known range. The sign bit determines whether the known range is positive or negative.
2. *Block Shuffling*: To increase the level of security, block shuffling is proposed. We divide each subband into a number of blocks of equal size. The size of the block can vary for different subbands. Within each subband, blocks of coefficients will be shuffled according to a shuffling table generated using a key. The shuffling table generally will be different for different subbands, and can vary from frame to frame.
3. *Block Rotation*: To further improve security with little impact on statistical coding, each block of coefficients can be rotated to form an encrypted block.

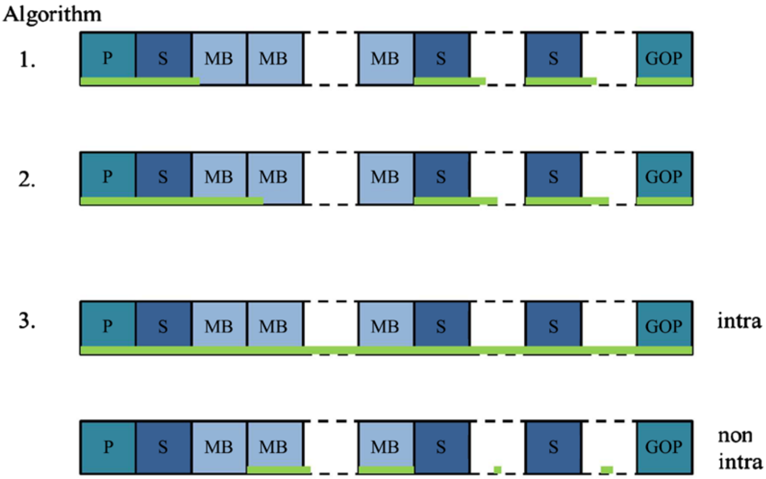


Fig. 2.2 Different levels of security offered by the SECMPEG algorithm (Meyer and Gadgegast [10]). P, S, MB and GOP refer to Primary Coding Unit (Individual Image), Slice Layer (restart points within a frame), Macro block layer (Motion Compensation Unit) and Group of Picture

2.3.2 Post-compression Encryption Algorithm

The Secure Real-time Transport Protocol, or the naive approach encrypts the compressed bitstream by packetizing multimedia data and individually encrypting every packet using AES. Although it is secure, it has huge computational overheads and it is not conducive to different desired properties of compressed bitstreams in general, owing to encryption of compressed data.

Many different algorithms have been proposed—which are format compliant, or have low computational requirements. Meyer and Gadegast [10] proposed a selective video encryption scheme called Secure MPEG or SECMPEG for the MPEG-1 video coding standard. See Fig. 2.2 for details. It offers different levels of security by encoding different parts of compressed bitstream:

- Algorithm 1: It encrypts only the headers from the sequence layer to the slice layer.
- Algorithm 2: It encrypts additionally low frequency DCT coefficients of all blocks in the I-frames.
- Algorithm 3: It encrypts all I-frames and all I-blocks in the P- and B-frames.
- Algorithm 4: It encrypts the whole MPEG-1 sequence with the naive algorithm.

The approach has some notable limitations: computations savings are not significant because I-frames constitute 30–60 % of an MPEG video. Moreover Agi and Gong [**22agi96??**] demonstrated that some scene contents are still discernible by directly playing back the selectively encrypted video stream on a conventional decoder. Maples and Spanos [17] presented a similar approach called AEGIS. All

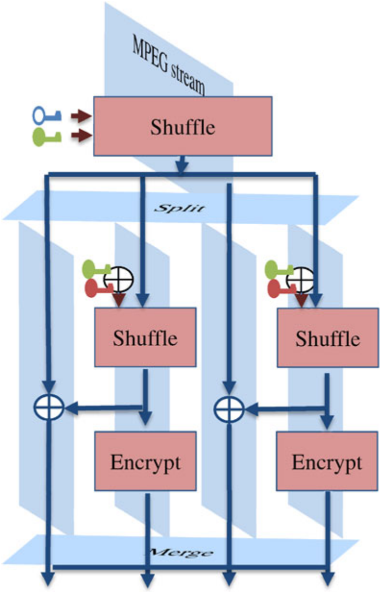


Fig. 2.3 The Video Encryption Algorithm proposed by Qiao and Nahrstedt [14]. MPEG packets are shuffled using key information for fast, efficient encryption

I-frames in an MPEG-video stream are encrypted, while P- and B-frames are left unencrypted. The AEGIS algorithm is almost same as SECMPEG level 2.

Qiao and Nahrstedt [14] introduced the Video Encryption Algorithm (VEA) which reduces the computational complexity to almost half. This video encryption algorithm is detailed in Fig. 2.3. Half of the bitstream is encrypted with a conventional encryption algorithm such as AES and is then used as key to XOR with the other half stream. The basic VEA algorithm is vulnerable to plaintext-attacks as an attacker can recover the whole frame from knowledge of either the odd or the even list. A $2n$-bits random key (KeyM) is used to split the $2n$-byte chunk randomly into two lists instead of the fixed odd-even pattern in the basic VEA. Thus, VEA also leads to increased key management issues.

Pure permutation or scrambling algorithms scramble bytes within a frame of MPEG stream by permutation. Adam J. Glagell demonstrates that pure permutation algorithm is vulnerable to known-plaintext attack, and hence its use should be carefully considered [**slagell04??**]. By comparing the ciphertext with the known frames, the adversary can recover the secret permutation list.

Chiaraluce et al. [5], Li et al. [6], Pareek et al. [11] and others propose a chaotic scheme for video encryption. Chaotic schemes are mostly based on encrypting image/ videos using chaotic maps. Logistic map is the simplest of them all and is popular choice to chaotic encryption scheme. However, simple cryptanalysis has been performed against these schemes.

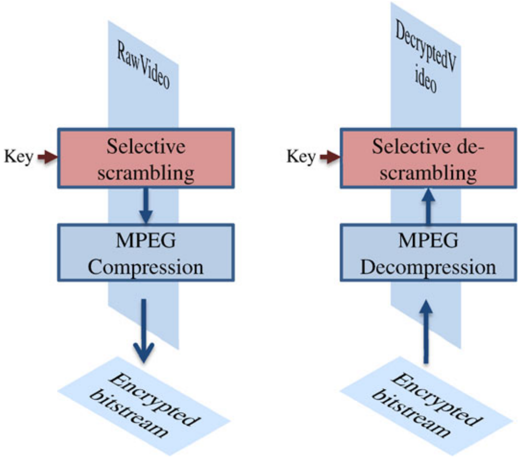


Fig. 2.4 Pre-compression encryption scheme proposed by Pazarci and Diplin [12]. The scrambler allows unauthorized user to have an arbitrarily degraded view of program, yet is totally transparent to MPEG-2 compression

2.3.3 Pre-compression Encryption Algorithm

Although it is possible to encrypt the video content before compression it has some serious limitations which are crucial for mobile devices:

1. Pre-compression encryption implies encrypting raw or uncompressed bits which will waste lot of computational resources.
2. Encryption output is generally a random bitstream with lack of redundancy, making compression operation highly inefficient for general case. For example, consider encrypting a HD video at bare resolution of 480p (852 × 480) with AES. It would require 2.3 Million AES cycles per second to encode (and to decode) that video on a mobile device (or any device)! Moreover, the compression performance will be mostly lost as the AES output bits will be nearly random with no possibility of lossless compression!

One known example is the work of Pazarci and Diplin [12]. The scrambler, shown in Fig. 2.4, is transparent to MPEG-2 compression. They encrypt the video in the RGB (red, green, blue) color space using four secret linear transformations before video coding. This scheme maintains the compression efficiency of the video codec but has been found unsafe against brute-force attacks.

2.3.4 Selective Encryption

The idea of selective encryption overlaps with post-compression approaches in some cases but it can also be applied during the compression process. A lot of research on integrating encryption with multimedia compression standards to reduce the overall computation cost is focused on using some form of selective encryption. For example, since most of the image energy in DCT domain is concentrated in the dc coefficient, Tang [19] proposed a system that encrypts dc coefficients with DES and scrambles the ac coefficients using a block-based permutation. However, the energy concentration is often unrelated to the degree of intelligibility (Wu and Kuo [21]). It was proven that the semantic content of the image is almost unaffected by removing the dc information. Therefore, the security level of Tang's system is reduced to that of the block-based permutation, making it vulnerable to various attacks. Wu and Kuo show that even encrypting some ac coefficients does not solve the problem. Shi, Wang and Bhargava [15] had proposed to encrypt every sign of DCT coefficients, but that effort was also refuted by Wu and Kuo. Pommer and Uhl [13] present a wavelet based selective encryption approach by using wavelet packet based compression instead of pyramidal compression schemes. Header information of a wavelet packet transform based image coding scheme is protected as AES is used to encrypt the subband decomposition structure. Lian et al. [8] uses Exp-Goloumb codes for the encryption operation. Cheng and Li [4] 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 Liu and Eskicioglu [9].

A syntax compliant encryption algorithm is proposed for H.264/AVC [3]. The authors allow any decoder to decode the encrypted video (although incorrectly) achieving up to 25–30 dB gains. Using the proposed method allows insertion of the encryption mechanism inside the video encoder, providing a secure transmission which does not alter the transmission process. The bits "selected for encryption" are chosen with respect to the considered video standard such that each of their encrypted configurations gives a non-desynchronized and fully standard compliant bitstream. This can in particular be done by encrypting only parts of the bitstream which have no or a negligible impact on evolution of the decoding process, and whose impact is consequently purely a visual one. For example, an encryption operation which leads to interpreting a given codeword instead of another of same size is suitable for such scheme. About 25 % of I-slices and 10–15 % of P-slices are encrypted. Since intracoded slices can represent 30–60 %, the encryption ratio is expected to be relatively high. The main drawback of this scheme is the lack of cryptographic security. Indeed, the security of the encrypted bitstream does not depend more on the AES cipher.

Lian et al. [7] use a mixture of these schemes for encrypting H.264 AVC bitstreams. During AVC encoding, such sensitive data as intra prediction mode, residue data and motion vector are encrypted partially. Among them, intra prediction mode is encrypted based on exp-Golomb entropy coding, the intra macroblocks DCs are encrypted based on context based adaptive variable length coding, and intra macroblocks ACs and the inter macroblocks MVDs are sign encrypted with a stream cipher followed with variable length coding. The encryption scheme is of high key sensitivity, which means that slight difference in the key causes great distortions in cipher video and that makes statistical or differential attack difficult. It is difficult to apply known-plaintext attack. In this encryption scheme, each slice is encrypted under the control of a 128 bit sub-key. Thus, for each slice, the brute-force space is 2^{128}; for the whole video, the brute-force space is 2^{256} (the user key is of 256 bit). According to the encryption scheme proposed here, both the texture information and the motion information are encrypted, making it difficult to recognize the texture and motion information in the video frames.

2.3.5 Joint Video Compression and Encryption (JVCE) Approaches

The main idea behind joint coding is to integrate encryption into compression operation by parameterization of the compression blocks, and (in general) not modifying the compressed bits. Two main compression blocks where these techniques have been applied are Wavelet Transform and Entropy Coding. We will present a brief summary of entropy coding-based approaches followed by a discussion of Wavelet Transform based approach proposed by the authors. Next, we will present the general rules of thumb in designing a new JVCE scheme.

Advantages JVCE approaches compression and encryption into a single operation making it feasible for mobile and embedded devices to ensure multimedia security with their low power budgets. By integrating compression and encryption operations into one, JVCE approaches reduce the latency of encryption operation which is useful for real-time video delivery. JVCE approaches typically do not change the compressed bit streams themselves but change the way compressed bitstream is obtained. This integration allows exploiting the hierarchical signal representation in a transform domain, as used by most image and video compression techniques, in order to provide the advanced functionalities required by many modern applications. The ISO/IEC JPEG 2000 Part 9 (JPSEC) standard is an example of how compression and security can coexist and take advantage of each other.

2.3.6 Future of JVCE Schemes

JVCE schemes have opened an entirely new paradigm of encryption without explicit-encryption of video content which gives them advantages in terms of computations, mobility, and friendliness to post-compression operations. However, many such schemes have been broken especially against known-plaintext attacks. To design an efficient encryption key for mobile applications, we propose the following directions: Development of JVCE algorithms for different video coding blocks and efficient integration into a common framework. An efficient integration will refute most of the cryptanalysis and the combined system will give a much greater degree of security than existing ciphers. Including some efficient scrambling operations into the design is meant to obfuscate input–output relationships at different levels.

References

1. Agi, I., Gong, L.: An empirical study of secure mpeg video transmissions. In: Proceedings of the Symposium on Network and Distributed System Security, pp. 137–144. IEEE Press, New York (1996)
2. Baugher, M., McGrew, D., Naslund, M., Carrara, E., Norrman, K.: The secure real-time transport protocol (SRTP) (2004)
3. Bergeron, C., Lamy-Bergot, C.: Complaint selective encryption for h.264/avc video streams. In: IEEE 7th Workshop on Multimedia Signal Processing, pp. 1–4 (2005). doi:10.1109/MMSP.2005.248641
4. Cheng, H., Li, X.: Partial encryption of compressed images and videos. IEEE Trans. Signal Process. **48**(8), 2439–2451 (2000). doi:10.1109/78.852023
5. Chiaraluce, F., Ciccarelli, L., Gambi, E., Pierleoni, P., Reginelli, M.: A new chaotic algorithm for video encryption. IEEE Trans. Consum. Electron. **48**(4), 838–844 (2002)
6. Li, S., Zheng, X., Mou, X., Cai, Y.: Chaotic encryption scheme for real-time digital video. In: Real-Time Imaging VI. Proceedings of SPIE, vol. 4666, pp. 149–160 (2002)
7. Lian, S., Liu, Z., Ren, Z., Wang, H.: Secure advanced video coding based on selective encryption algorithms. IEEE Trans. Consum. Electron. **52**(2), 621–629 (2006)
8. Lian, S., Liu, Z., Ren, Z., Wang, H.: Commutative encryption and watermarking in video compression. IEEE Trans. Circuits Syst. Video Technol. **17**(6), 774–778 (2007)
9. Liu, X., Eskicioglu, A.M.: Selective encryption of multimedia content in distribution networks: challenges and new directions. In: Communications, Internet, and Information Technology (CIIT 2003), pp. 276–285 (2003)
10. Meyer, J., Gadegast, F.: Security mechanisms for multimedia data with the example mpeg-1 video. Project Description of SECMPEG, Technical University of Berlin, Germany (1995)
11. Pareek, N.K., Patidar, V., Sud, K.K.: Image encryption using chaotic logistic map. Image Vis. Comput. **24**(9), 926–934 (2006)
12. Pazarci, M., Dipcin, V.: A mpeg2-transparent scrambling technique. IEEE Trans. Consum. Electron. **48**(2), 345–355 (2002)
13. Pommer, A., Uhl, A.: Selective encryption of wavelet-packet encoded image data: efficiency and security. Multimed. Syst. **9**(3), 279–287 (2003)
14. Qiao, L., Nahrstedt, K.: A new algorithm for mpeg video encryption. In: Proc. of First International Conference on Imaging Science System and Technology, pp. 21–29 (1997)
15. Shi, C., Wang, S.Y., Bhargava, B.: Mpeg video encryption in real-time using secret key cryptography. In: Proc. Int. Conf. Parallel and Distributed Processing Techniques and Applications (1999)
16. Slagell, A.J.: Known-plaintext attack against a permutation based video encryption algorithm (2004)
17. Spanos, G.A., Maples, T.B.: Performance study of a selective encryption scheme for the security of networked, real-time video. In: ICCCN, pp. 2–10. IEEE Comput. Soc., Los Alamitos (1995)
18. Stinson, D.R.: Cryptography: Theory and Practice. CRC Press, Boca Raton (2006)
19. Tang, L.: Methods for encrypting and decrypting mpeg video data efficiently. In: Proceedings of the Fourth ACM International Conference on Multimedia, pp. 219–229. ACM Press, New York (1997)
20. Wen, J., Luttrell, M., Severa, M.: Access control of standard video bitstreams. In: Proc. IEEE Intl. Conf. Media Future (2001)
21. Wu, C.-P., Kuo, C.-C.J.: Design of integrated multimedia compression and encryption systems. IEEE Trans. Multimed. **7**(5), 828–839 (2005). doi:10.1109/TMM.2005.854469
22. Wu, C.P., Kuo, C.C.J.: Efficient multimedia encryption via entropy codec design. In: Proceedings of SPIE, vol. 4314, p. 128 (2001)
23. Zeng, W., Lei, S.: Efficient frequency domain selective scrambling of digital video. IEEE Trans. Multimed. **5**(1), 118–129 (2003). doi:10.1109/TMM.2003.808817

Chapter 3
Securing Multimedia Content Using Joint Compression and Encryption

Abstract Algorithmic parameterization and hardware architectures can ensure secure transmission of multimedia data in resource-constrained environments such as wireless video surveillance networks, tele-medicine frameworks for distant health care support in rural areas, and Internet video streaming. Joint multimedia compression and encryption techniques can significantly reduce the computational requirements of video processing systems. We present an approach to reduce the computational cost of multimedia encryption, while also preserving the properties of compressed video (useful for scalability, transcoding, and retrieval), which endangers loss by naive encryption. Hardware-amenable design of proposed algorithms makes them suitable for real-time embedded multimedia systems. This approach alleviates the need of additional hardware for encryption in resource-constrained scenario, and can be otherwise used to augment existing encryption methods used for content delivery in Internet or other applications.

3.1 Introduction

In this work, we discuss design of algorithms and hardware architectures for secure transmission of multimedia data in resource-constrained environments. Some typical application scenarios include wireless video surveillance networks, telemedicine frameworks for distant health care support in rural areas, and Internet video streaming.

Wireless Video Surveillance Networks Wireless video surveillance networks have been recently deployed in different network settings such as WiFi, WiMAX, and wireless sensor networks. These networks are deployed in private and public settings, and carry sensitive visual information. For example, the live feeds from over 10,000 cameras are used by city police departments in the NYC and Chicago areas to monitor criminal activity. It is important to protect the video feeds of these cameras from eavesdropping. However, the three-fold limitations of (a) wireless network characteristics (low bandwidth, frequent packet drops), (b) QoS (real-time delivery, low jitter), and (c) the limited computational resources at the encoder end makes it difficult to provide real-time end-to-end encryption of video data using

A. Pande, J. Zambreno, *Embedded Multimedia Security Systems*,
DOI 10.1007/978-1-4471-4459-5_3, © Springer-Verlag London 2013

conventional cryptographic primitives. Apart from the need of a secure way of transmitting videos, we need computationally efficient algorithms to save on computing power and also enable easy access of visual information from encrypted videos in databases.

Tele-medicine Frameworks Tele-medicine is an application of clinical medicine where consultation, and even remote medical procedure and examinations are performed using interactive audio-visual media. Extending such services to remote locations (which lack high-speed connections and even electric power in underdeveloped countries) requires efficient low-power devices. Further, the privacy of patient information and prescriptions is an important concern for these applications, considering the vulnerability of communication channels against eavesdropping and other attacks.

Internet and Mobile Video The advent of embedded multimedia systems has already revolutionized the way we live. Video messaging, video-conferencing, video surveillance and Internet video sites such as YouTube are increasingly becoming popular and pervasive. Network traffic in next-generation cellular networks is predicted to be dominated by video [22] and it makes sense to provision for security of videos in these applications. Most mobile devices have low computational resources and limited battery resources.

Conventional encryption schemes such as those using AES and DES are not suitable for video data because of the large computational overhead. Compressed multimedia streams also exhibit well-defined hierarchical structures that can be exploited in several useful ways (e.g. scalability, random access, transcoding, rate shaping) in low and variable bandwidth scenarios—these structures would not be recognizable in traditional ciphertext.

In this work, an augmented video coding model is used for joint compression and encryption which can significantly reduce the computational requirements. We propose to build design blocks which enable security for these applications at the algorithmic level, and leave domain-specific optimization to application developers. These algorithmic optimizations map easily to fixed point hardware, allowing us to come up with efficient architectural optimizations for resource-constrained scenarios. In other application scenarios, these approaches can complement the security provided by conventional schemes such as AES.

The proposed schemes are also low-cost in the sense that the required computational hardware is considerably smaller than existing approaches, and in some configurations the hardware needed is no larger than that for conventional video compression.

3.2 Basics

Multimedia compression involves large computations and large amount of datatransfers thus requiring application-specific hardware such as ASICs and FPGAs to compress and deliver the media in real-time [29]. Video compression using hardware accelerators has gained increased attention because of the popularity of low-power embedded devices [1, 5, 25]. Recently, the authors in [3] proposed an embedded video codec with area and external access power savings to support a real-time encoding of CIF images, using a power-aware design for video coding in embedded scenarios [3]. However, their approach achieves real-time coding of only small resolutions up to 352 × 288 pixels. *Thus, an efficient architectural design of multimedia compression blocks is a must to ensure real-time video delivery.*

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 ciphertext, 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 does not change the compressed output, yet provides access and copy control for concerned media. *Thus, we need encryption of video data without affecting the properties of compressed bitstream, or affecting the compression performance.*

On one hand, compression and encryption operations require large amount of computational overhead, while on the other hand, there has been an increasing trend towards deployment of battery-driven low-power embedded systems such as portable mobile devices (iPods, mobile phones, and cameras). *Apart from optimizations in hardware architecture, we also need to reduce the computation cost for secure multimedia transactions through algorithmic improvements.*

Related Research Efforts The research in video coding for the last five decades has been commercially utilized in the form of state-of-the-art video coding standards such as MPEG-1, MPEG-2 and so on. MPEG-2-based schemes [9, 13, 20, 21, 23, 24, 35] are useful for DVD quality compression. In these scenarios, coding or power efficiency are not the constraints and security was provided using end-end encryption with AES or some variants [2]. Some work has been done for resource optimization in these schemes in cases of low bandwidth [4, 11, 18, 32], but the problem is not so acute.

Recent research in wireless networks (such as cognitive radio networks) and video surveillance [10, 14] aims at optimizing the video quality for wireless transmission and often uses the MPEG-4 format which produces a more compressed and scalable bitstream. H.264 SVC is the most recently used format in this work. Many researchers have also tried to optimize the hardware implementations of MPEG-2- [15, 17, 27, 30, 31] and MPEG-4-based video applications [6, 16, 33].

Recent research in video encryption [7, 8, 12, 34] over wireless and other scarce resource channels has identified the need for non-traditional approaches to video encryption besides the use of standard cryptographic ciphers. These approaches involve selective or partial encryption of video stream, chaotic encryption and shuffling in compressed bitstream etc. There has also been research to accelerate these video processing kernels in hardware such as ASIC or FPGA [19, 28, 36].

Thus, there has been little research which targets the three-fold goal of high compression, low computational cost, and secrecy. As shown in Fig. 3.1, there has been

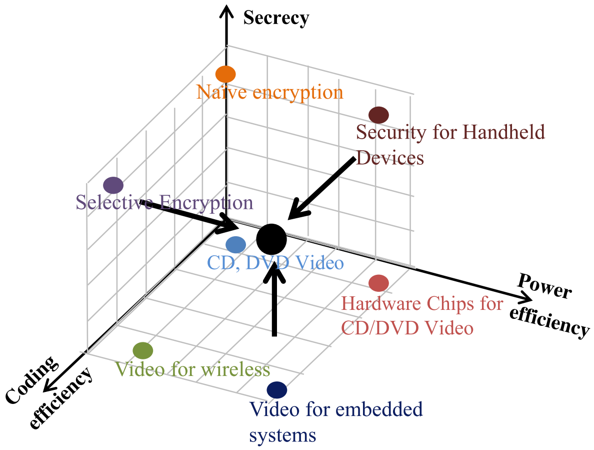


Fig. 3.1 The broad goal of proposed research is to develop algorithms and architectures to push the operating curve towards (1, 1, 1), considering all the factors (Coding Efficiency, Power Efficiency and Privacy) together during design

little research that considers the three objectives: coding efficiency (compression achieved), power efficiency (architectural efficiency) and privacy (encryption) and aim to optimize them together. With these three goals in mind, we propose our approach in the next section.

3.3 Our Approach

What we propose in this work is a redesign of the video compression blocks themselves to enable encryption and efficient mapping onto hardware. For example, if the video coders have an additional parameter which can be changed to provide encryption, we can use it as a keyspace for secret key generation. The required mixing of the input, as required by cryptographic ciphers, is automatically provided by different blocks of video coding system. Similarly, if we could design the system with the rational coefficients as a design constraint, we will obtain a hardware-amenable implementation.

The redesign of video coding blocks enables joint compression and encryption and also reduces the computational requirements of multimedia encryption algorithms. The approach modifies the compression system properties instead of the compressed bitstream itself. Moreover, the redesign is amenable to hardware acceleration over reconfigurable computing platforms. We leverage signal processing techniques to make the algorithms suitable for hardware optimizations (and encryption), and reduce the critical path of circuits using hardware-specific optimizations.

A trivial way to explain this solution (of joint encryption and compression) is to find 2^N different but similar ways to compress a video, where all ways give similar compression performance and the compressed bitstream has the same properties. For large values of 2^N, we can say that the N bit code representing the choice of

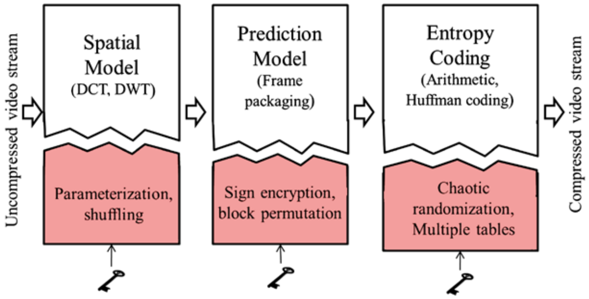

Uncompressed video stream
Spatial Model
(DCT, DWT)
Parameterization, shuffling
Prediction Model
(Frame packaging)
Sign encryption, block permutation
Entropy Coding
(Arithmetic, Huffman coding)
Chaotic randomization, Multiple tables
Compressed video stream

Fig. 3.2 Video compression system augmented with different operations to ensure real-time encryption

compression system is the encryption key of the system. In order for such a system to be secure, the combined system must follow cryptographic requirements such as good diffusion and confusion properties [26]. The output from two closely related keys should be nearly uncorrelated and there should not exist a way to reverse-engineer the N bit key except by a brute-force attack.

This proposal also meets the requirements of property-preserving encryption because essentially we are trying to shuffle the compression parameters using the key and not modifying the input bitstream itself. Each of the 2^N compression systems provides ‘property-preserving’ compression.

How Exactly Can We Augment Encryption to Video Coding System? This is achieved by redesign of individual video coding blocks followed by integration into a single prototype and hardware implementation (Fig. 3.2). These modifications are briefly described below.

(a) **Augmented Prediction Model**: We propose to use a fuzzy prediction model, which selects from several past and future frames and uses multiple streams, based on a key-dependent fuzzy logic instead of the traditional use of immediate neighbors. Similarly, the sign bits of the motion vectors can be encoded and/or a key-based non-linear mapping of motion vectors can be performed.
(b) **Augmented Spatial Model**: We propose to parameterize the transform filter (DWT), so that the choice of filter depends on key value. Different filters give different output coefficients while the compression efficiency of each is similar. The output sub-bands of DWT (or sub-blocks for DCT) can be re-oriented and permuted according to a key.
(c) **Augmented Entropy Coding**: Modified entropy coders can be used with multiple statistical models so that the exact choice of model is governed by a key. Similarly, arithmetic coding can be implemented using a key-based chaotic random map. The re-iterations of chaotic map make the output appear random, while the choice of map itself is governed by a key.
(d) **All-on-a-Chip**: Hardware-specific optimizations on augmented modules will enable us to fit the prototype on a single chip.

References

1. Bahari, A., Arslan, T., Erdogan, A.T.: Low-power h.264 video compression architectures for mobile communication. IEEE Trans. Circuits Syst. Video Technol. **19**(9), 1251–1261 (2009). doi:10.1109/TCSVT.2009.2022779
2. Baugher, M., McGrew, D., Naslund, M., Carrara, E., Norrman, K.: The Secure Real-time Transport Protocol (SRTP). RFC Editor, United States (2004)
3. Cheng, C.-C., Tseng, P.-C., Chen, L.-G.: Multimode embedded compression codec engine for power-aware video coding system. IEEE Trans. Circuits Syst. Video Technol. **19**(2), 141–150 (2009). doi:10.1109/TCSVT.2008.2009250
4. Conti, M., Gregori, E.: Traffic characterization and bandwidth allocation for mpeg-2 vbr video traffic. In: IEEE, Global Telecommunications Conference, GLOBECOM '97, vol. 1, pp. 443–4471 (1997). doi:10.1109/GLOCOM.1997.632586
5. Dandalis, A., Prasanna, V.K.: Configuration compression for FPGA-based embedded systems. In: FPGA '01: Proceedings of the 2001 ACM/SIGDA Ninth International Symposium on Field Programmable Gate Arrays, pp. 173–182. ACM Press, New York (2001). doi:10.1145/360276.360342
6. De Vleeschouwer, C., Nilsson, T., Denolf, K., Bormans, J.: Algorithmic and architectural co-design of a motion-estimation engine for low-power video devices. IEEE Trans. Circuits Syst. Video Technol. **12**(12), 1093–1105 (2002). doi:10.1109/TCSVT.2002.806810
7. Dufaux, F., Ebrahimi, T.: Scrambling for privacy protection in video surveillance systems. IEEE Trans. Circuits Syst. Video Technol. **18**(8), 1168–1174 (2008). doi:10.1109/TCSVT.2008.928225
8. Elkilani, W.S., Abdul-Kader, H.M.: Performance of encryption techniques for real time video streaming. In: International Conference on Networking and Media Convergence, ICNM 2009, pp. 130–134 (2009). doi:10.1109/ICNM.2009.4907203
9. Fossel, S., Fottinger, G., Mohr, J.: Motion jpeg2000 for high quality video systems. IEEE Trans. Consum. Electron. **49**(4), 787–791 (2003). doi:10.1109/TCE.2003.1261156
10. Gualdi, G., Prati, A., Cucchiara, R.: Video streaming for mobile video surveillance. IEEE Trans. Multimed. **10**(6), 1142–1154 (2008). doi:10.1109/TMM.2008.2001378
11. Guo, Z., Ni, J., Yang, T.: Traffic smoothing and bandwidth allocation for VBR MPEG-2 video connections in ATM network. In: Proceedings of Fourth International Conference on Computer Communications and Networks, pp. 554–561 (1995). doi:10.1109/ICCCN.1995.540173
12. Hamdi, M., Boudriga, N.: A progressive chaotic mpeg-4 video encryption scheme for wireless networks. In: IEEE International Conference on Communications, ICC '09, pp. 1–5 (2009). doi:10.1109/ICC.2009.5305955
13. Haskell, B.G., Puri, A., Netravali, A.N.: Digital Video: An Introduction to MPEG-2. Kluwer Academic, Dordrecht (1997)
14. Hu, D., Mao, S., Hou, Y.T., Reed, J.H.: Scalable video multicast in cognitive radio networks. IEEE J. Sel. Areas Commun. **28**(3), 334–344 (2010). doi:10.1109/JSAC.2010.100414
15. Irfan, M., Khan, A.K., Jamal, H.: FPGA based implementation of MPEG-2 compression algorithm. In: The 17th International Conference on Microelectronics, 2005. ICM 2005, pp. 240–244 (2005). doi:10.1109/ICM.2005.1590075
16. Kamat, S.P.: Low bandwidth YCbCr data processing technique for video applications in handheld devices. IEEE Trans. Consum. Electron. **56**(3), 1770–1774 (2010). doi:10.1109/TCE.2010.5606324
17. Kinsman, A.B., Nicolici, N.: A VLSI architecture and the FPGA prototype for MPEG-2 audio/video decoding. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **PP**(99), 1–5 (2009). doi:10.1109/TVLSI.2009.2034168

18. Kotra, A., Fohler, G.: Resource aware real-time stream adaptation for mpeg-2 transport streams in constrained bandwidth networks. In: 2010 IEEE International Conference on Multimedia and Expo (ICME), pp. 729–730 (2010). doi:10.1109/ICME.2010.5583196
19. Ladis, E., Papaefstathiou, I., Marchesani, R., Tuinenbreijer, K., Langendorfer, P., Zahariadis, T., Leligou, H., Redondo, L., Riesgo, T., Kannegiesser, P., et al.: Secure, mobile visual sensor networks architecture. In: 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks Workshops, SECON Workshops' 09, pp. 1–3. IEEE Press, New York (2009)
20. Martins, B., Forchhammer, S.: A unified approach to restoration, deinterlacing and resolution enhancement in decoding mpeg-2 video. IEEE Trans. Circuits Syst. Video Technol. **12**(9), 803–811 (2002). doi:10.1109/TCSVT.2002.803227
21. Matsumoto, S., Hamada, T., Koga, K., Murakami, H.: Development of high compression HDTV digital codec. In: Fifth IEE Conference on Telecommunications, pp. 100–104 (1995). doi:10.1049/cp:19950121
22. Oyman, O., Foerster, J., Tcha, Y.j., Lee, S.-C.: Toward enhanced mobile video services over wimax and lte [wimax/lte update]. IEEE Commun. Mag. **48**(8), 68–76 (2010). doi:10.1109/MCOM.2010.5534589
23. Petajan, E.: The HDTV grand alliance system. IEEE Commun. Mag. **34**(6), 126–132 (1996). doi:10.1109/35.506821
24. Pettersson, P.E.: Digital HDTV using MPEG-2. In: IEE Colloquium on Advanced, Widescreen and High Definition Television Systems—Where Are They Now? pp. 4–142 (1996). doi:10.1049/ic:19960102
25. Schoner, B., Villasenor, J., Molloy, S., Jain, R.: Techniques for fpga implementation of video compression systems. In: FPGA '95: Proceedings of the 1995 ACM Third International Symposium on Field-Programmable Gate Arrays, pp. 154–159. ACM Press, New York (1995). doi:10.1145/201310.201334
26. Shannon, C.E.: Communication theory of secrecy systems. Bell Syst. Tech. J. **28**, 656–715 (1949)
27. Tanougast, C., Berviller, Y., Mannino, C., Rabah, H., Janiaut, M., Weber, S.: SystemC model of a MPEG-2 DVB-T bit-rate measurement architecture for FPGA implementation. In: Proceedings of 15th IEEE International Workshop on Rapid System Prototyping, pp. 157–163 (2004). doi:10.1109/IWRSP.2004.1311111
28. Thomas, T., Emmanuel, S., Das, A., Kankanhalli, M.S.: A crt based watermark for multiparty multilevel drm architecture. In: IEEE International Conference on Multimedia and Expo, ICME 2009, pp. 1010–1013 (2009). doi:10.1109/ICME.2009.5202668
29. Tseng, P., Chang, Y., Huang, Y., Fang, H., Huang, C., Chen, L.: Advances in hardware architectures for image and video coding—a survey. Proc. IEEE **93**(1), 184–197 (2005). doi:10.1109/JPROC.2004.839622
30. Verderber, M., Zemva, A., Lampret, D.: HW/SW partitioned optimization and VLSI-FPGA implementation of the MPEG-2 video decoder. In: Design, Automation and Test in Europe Conference and Exhibition, pp. 238–243 (2003). doi:10.1109/DATE.2003.1253835
31. Verderber, M., Zemva, A., Trost, A.: Hw/sw codesign of the mpeg-2 video decoder. In: Proceedings of International Parallel and Distributed Processing Symposium, p. 7 (2003). doi:10.1109/IPDPS.2003.1213330
32. Verscheure, O., Frossard, P., Hamdi, M.: User-oriented QoS analysis in MPEG-2 video delivery. Real-Time Imaging **5**(5), 305–314 (1999)
33. Wang, S.-H., Tai, S.-H., Chiang, T.: A low-power and bandwidth-efficient motion estimation ip core design using binary search. IEEE Trans. Circuits Syst. Video Technol. **19**(5), 760–765 (2009). doi:10.1109/TCSVT.2009.2017416
34. Wen, J., Severa, M., Zeng, W., Luttrell, M.H., Jin, W.: A format-compliant configurable encryption framework for access control of video. IEEE Trans. Circuits Syst. Video Technol. **12**(6), 545–557 (2002). doi:10.1109/TCSVT.2002.800321

35. Xiong, H., Yu, S., Ye, W.: Implementation of digital HDTV encoder with parallel sub-picture encoding modules and its joint bit-allocation strategy. IEEE Trans. Consum. Electron. **48**(4), 898–907 (2002). doi:10.1109/TCE.2003.1196418
36. Zhang, Y., Liu, Z., Zheng, X.: A chaos-based image encryption ASIC using reconfigurable logic. In: IEEE Asia Pacific Conference on Circuits and Systems, APCCAS 2008, pp. 1782–1785 (2008). doi:10.1109/APCCAS.2008.4746387

Part II
Examples

This is part two of the book. It discusses some specific examples of building secure embedded multimedia systems, implementation details, and results giving a hands-on feel to the reader.

Chapter 4
Polymorphic Wavelet Transform

Abstract 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 chapter, 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. Poly-DWT serves as an example of joint design of algorithms and architectures for multimedia compression.

4.1 Introduction

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 [30, 32, 36]. The work on using Embedded Zero-tree Wavelet (EZW) structures [32] for image compression was a milestone research that introduced subband coding to achieve high compression performance. Said and Pearl-

A. Pande, J. Zambreno, *Embedded Multimedia Security Systems*,
DOI 10.1007/978-1-4471-4459-5_4, © Springer-Verlag London 2013

man [30] introduced a more efficient DWT-based Set Partitioning in Hierarchical Trees (SPIHT) encoding to improve the performance of Shapiro's EZW coding. Taubman [36] proposed the DWT-based Embedded Block Coding with Optimal Truncation (EBCOT) coding algorithm which was accepted for scalable encoding in JPEG2000 [8]. 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 [27]; 3-D, 4-D subband coding [7, 44]; and MPEG-4 SVC (Scalable Video Coding extension, released in July 2007) [31]. Of the 14 proposals for SVC received by the MPEG committee, 12 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 work. Implementations such as those using ASICs or FPGAs are capable of accelerating these computations by exploiting the inherent algorithmic parallelism. Stroobandt et al. [35] discuss the performance requirements of a reconfigurable hardware architecture for a scalable wavelet-based video decoder. In [12], the authors present a complete video deliver 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 [2, 4, 5, 17, 20, 29, 43]. 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 [40, 41]. 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 work, 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 run-time reconfigurability of the design over commodity FPGA platforms and ASIC designs.

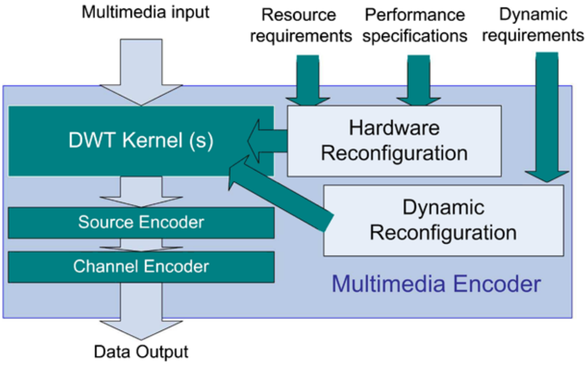


Fig. 4.1 Conceptual overview of the Polymorphic Wavelet Architecture

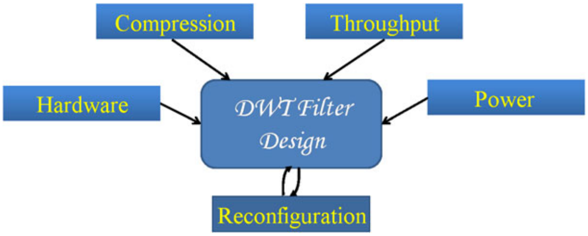


Fig. 4.2 Conceptual overview of the DWT filter design constraints and desired features

Figure 4.1 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 [18], remote laboratories [22], or educational video streaming [24] may benefit enormously from the polymorphism of DWT kernels.

We summarize the requirements of embedded multimedia system design in Fig. 4.2 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 work 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 [23, 25]. 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 work 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 work is organized as follows. In Sect. 4.2, we give a working knowledge of DWT filters used in image compression. Section 4.3 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 Sect. 4.4. 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 4.5 gives an insight into hardware re-allocation by changing the internal word width representation of registers by 'bit-width' switching scheme. Section 4.6 introduces the 'on-the-fly' switching scheme for filter coefficients. Section 4.7 also gives details of other experiments both in ModelSim and Xilinx ISE for hardware performance and over MATLAB for rig-

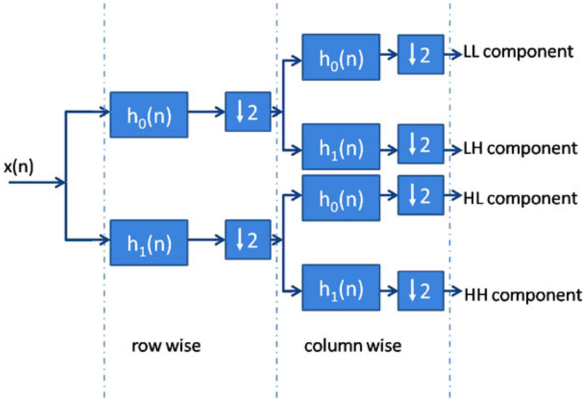


Fig. 4.3 Basic stages of a one level 2-D wavelet transform operation

orous image processing performance measurements. In Sect. 4.8 we conclude the work with a look towards planned future work.

4.2 Motivation and Insight

Prior work in signal processing explains 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 Fig. 4.3. 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 [8]. 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 work develops the Poly-DWT architecture to serve as a backbone for real-time multimedia applications to address their dynamic demands and constraints. In this work 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 [22, 24] 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 work 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.

4.2.1 Daubechies 9/7-Tap Bi-orthogonal Filter

The bi-orthogonal Daubechies $9/7$ filter is the most widely used filter for DWT operation. It is also commonly called CDF filter. 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 and the idea behind their construction is same as orthogonal Daubechies wavelets.

For Daubechies filters, solutions are derived by solving the following factorization for polynomial $Q_I(X)$ where I is an integer:

$$(1 - X/2)^I Q_I(X) + (X/2)^I Q_I(2 - X) = 1$$

To get the series of solutions for Daubechies filters, this equation is solved using spectral factorization. However, the following factorization is used in CDF $9/7$ filter:

$$Q_I(X) = q_{\text{primal}}(X) q_{\text{dual}}(X)$$

A simple factorization is done with $q_{\text{primal}}(X) = 1$ and $q_{\text{dual}}(X) = Q_I(X)$. Solving this equation for $I = 1$ gives famous Haar wavelets while $I = 4$ gives the coefficients for CDF filter. The bi-orthogonal pair of sequences for wavelet filter are calculated as follows:

$$G_0(Z) = 2Z^p \left(\frac{1+Z}{2}\right)^2 (1 - cX) \tag{4.1}$$

$$H_0(Z) = 2Z^p \left(\frac{1+Z}{2}\right)^2 Q_4\left(1 - \frac{Z + Z^{-1}}{2}\right) \tag{4.2}$$

where

$$Q_4(X) = 1 + 2X + \frac{5}{2}X^2 + \frac{5}{2}X^3 \tag{4.3}$$

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.

$$H_1(z) = z^{-1} G_0(-z), G_1(z) = z H_0(-z) \tag{4.4}$$

Table 4.1 Coefficients for the CDF 9/7 filter

i	$h_0(i)$	$h_1(i)$	$g_0(i)$	$g_1(i)$
±4	0.026748757411	0	0	0.026748757411
±3	−0.016864118443	0.091271763114	−0.091271763114	0.016864118443
±2	−0.078223266529	−0.057543526229	−0.057543526229	−0.078223266529
±1	0.266864118443	−0.591271763114	0.591271763114	−0.266864118443
0	0.602949018236	1.11508705	1.11508705	0.602949018236

Table 4.2 Coefficients for Le Gall 5/3 filter

i	$h_0(i)$	$h_1(i)$	$g_0(i)$	$g_1(i)$
±2	−1/8	0	0	−1/8
±1	2/8	−1/2	1/2	−2/8
0	6/8	1	1	6/8

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

$$D(z) + D(-z) = 2 \tag{4.5}$$

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

$$D(z) = L_k(z) = z^k \left(\frac{1+z^{-1}}{2}\right)^{2k} \alpha(z) \tag{4.6}$$

$$\alpha(z) = \sum_{n=0}^{k-1} \left({}^{k+n-1}C_n\right) \left(\frac{2-(z+z^{-1})}{4}\right)^n \tag{4.7}$$

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 4.1. Ansari et al. [3] discuss the derivation in detailed steps.

4.2.2 Le Gall's 5/3 Filter

Le Gall and Tabatabai [14] solved the PR condition by substituting $D(z) = a_0 + a_2z^{-2} + a_3z^{-3} + a_2z^{-4} + a_0z^{-6}$ 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 4.2.

$$G_0(Z) = 2Z^p \left(\frac{1+Z}{2}\right)^2 \tag{4.8}$$

$$H_0(Z) = 2Z^p\left(\frac{1+Z}{2}\right)^2 Q_2\left(1 - \frac{Z+Z^{-1}}{2}\right) \tag{4.9}$$

This filter has lower latency than the ones studied earlier but provides lesser image compression capabilities.

$$h_0(i) \quad \text{or} \quad low_{53}(i) = \frac{3}{4}x(i) + \frac{1}{4}\big(x(i-1) + x(i+1)\big) - \frac{1}{8}\big(x(i-2) + x(i+2)\big) \tag{4.10}$$

$$h_1(i) \quad \text{or} \quad high_{53}(i) = x(i) - \frac{1}{2}\big(x(i-1) + x(i+1)\big) \tag{4.11}$$

4.3 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 work in partial reconfiguration of FPGAs provides insight into the state-of-the-art. Claus et al. [10] give 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 [16] simplifies the generation of dynamically reconfigurable systems to almost a push button process. The work also describes a communication infrastructure for dynamically reconfigurable systems. Claus et al. [11] present an IP core that enables fast on-chip dynamic partial reconfiguration close to the maximum achievable speed. Paulsson et al. [26] 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 work 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 4.3.1 discusses the development of the theory of wavelet transform, and its efficient image processing capabilities. Section 4.3.2 describes some recent attempts at implementing DWT on reconfigurable platforms.

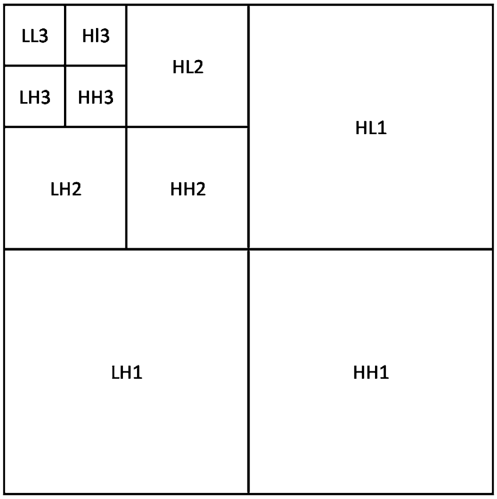


Fig. 4.4 Result of three level 2-D wavelet transform operation on an image

4.3.1 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 Fig. 4.4. The coarse information is preserved in the LL3 image and this operation forms the basis of Multi-Resolution Analysis for DWT [42].

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 (BWFBs), 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 implementation like that achieved from lifting by using a folding scheme.

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 [21, 28]. These multiplier-free implementations involve image reconstruction quality trade-offs. Many other researchers have also faced the problem of reducing DWT complexity [2, 20, 29]. 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.

4.3.2 Hardware Implementation of DWT

Much research has been done in the development of DWT architectures for image processing [4, 5, 17, 21, 29]. A good survey on architectures on DWT coding is given by [39]. The work gives insight on hardware implementations for JPEG2000 scheme which is based on DWT computations. The computational complexity analysis of JPEG2000 by [1, 19] 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 [8, 33].

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 [15]. 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 work, 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 and Hauck [6] 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 work 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. [5] 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.

Martina and Masera [21] 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 [20]. 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 better image reconstruction results over the original filter. As will be described in Sect. 4.4, our folded implementation reduces the number of adders to just nine.

In [37], 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 work to facilitate the development of a polymorphic structure.

4.4 Poly-DWT Filter

A look at Table 4.1 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.

4.4.1 Parameterized Filter Design

A parameterized design alleviates the problem of irrational coefficients. Equation (4.13) can be re-written in the following form, with $Z = 1/2 \cdot (z + z^{-1})$:

$$D(Z) = 2^K (1 + Z)^K \alpha(Z) \tag{4.12}$$

$$\alpha(Z) = \sum_{n=0}^{k-1} \left({}^{k+n-1}C_n\right) \left(\frac{1 - Z}{2}\right)^n \tag{4.13}$$

Tay [37] poses this constraint on $D(z)$ to derive the binary rational coefficients and achieve new sets of 9/7 filters by adding more degrees of freedom to the original

LHBF equation (by introducing a free parameter α). Essentially, $D(Z)$ is re-written as follows:

$$D(Z) = 2^K(1+Z)^{k-1}(\beta+Z)\alpha(Z) \tag{4.14}$$

$$\alpha(Z) = \sum_{n=0}^{k-1}\left({}^{k+n-1}C_n\right)\left(\frac{1-Z}{2}\right)^n \tag{4.15}$$

where β signifies free factor. The solution is then obtained for different values of K. The values for $K = 4$ allow the following break-up of analysis and synthesis low-pass filters:

$$H_0(Z) = K_h(Z+1)\left(Z^3 + AZ^2 + VZ + C\right) \tag{4.16}$$

$$G_0(Z) = K_g(Z+1)^2(Z+\alpha) \tag{4.17}$$

$$D(Z) = K_hK_g(Z+1)^3(Z+\alpha)\left(Z^3 + AZ^2 + BZ + C\right) \tag{4.18}$$

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 α:

$$A = -(3+\alpha) \tag{4.19}$$

$$B = \frac{9\alpha^3 + 35\alpha^2 + 48\alpha + 24}{3\alpha^2 + 9\alpha + 8} \tag{4.20}$$

$$C = \frac{8(1+\alpha)^3}{3\alpha^2 + 9\alpha + 8} \tag{4.21}$$

Setting α to -1.6848 gives back the original 9/7 filter pair.

4.4.2 Numerical Study

The parameter α 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 quantization error for the filter coefficients with α varying from -1.6 to -2 (in vicinity of the $\alpha = -1.6848$ value). The results are presented in Fig. 4.5. 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 4.3 and 4.4.

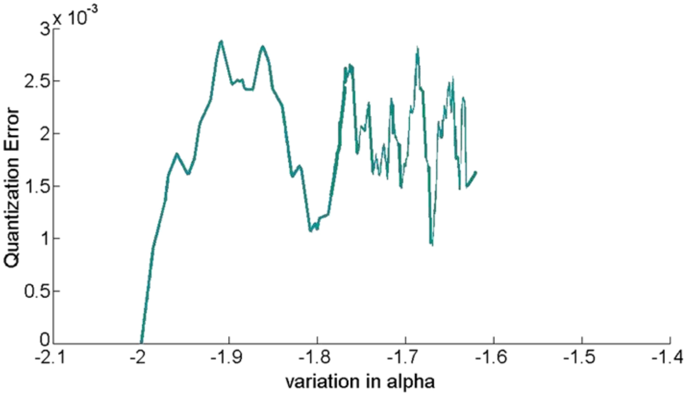


Fig. 4.5 Numerical analysis of quantization error for seven bit finite representation of filter coefficients

Table 4.3 Analysis high-pass filter coefficients (H_1) for the bi-orthogonal 9/7 tap filter

i	α			
	1.6848	−1.667	−1.8	−2
±3	0.091271763114	1/16	1/16	1/16
±2	−0.057543526229	−1/16	−1/16	0
±1	−0.591271763114	−9/16	−9/16	−9/16
0	1.11508705	9/8	9/8	1

Table 4.4 Analysis low-pass filter (H_0) coefficients for the bi-orthogonal 9/7 tap filter

i	α			
	1.6848	−1.667	−1.8	2
±4	0.026748757411	1/32	1/32	1/64
±3	−0.016864118443	−1/32	0	0
±2	−0.078223266529	−1/16	−3/32	−1/8
±1	0.266864118443	9/32	1/4	1/4
0	0.602949018236	19/32	5/8	23/32

4.4.3 Candidate Filters

Let us consider an input signal $x(i)$. The low- and high-pass output 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:

$$low(i) = \sum_{k=-4}^{k=4} h_0(k) \cdot x(i-k) \tag{4.22}$$

$$high(i) = \sum_{k=-3}^{k=3} h_1(k) \cdot x(i-k) \tag{4.23}$$

Substituting the values of filter coefficients from Tables 4.2, 4.3, and 4.4 we can factorize our 9/7 filter coefficients in terms of 5/3 filter output. The subscripts A, B, and C are used to denote the filters obtained with $\alpha = -1.67$, -1.8, and -2, respectively:

$$\begin{aligned} low_A(i) &= \frac{19}{32}x(i) + \frac{9}{32}\big(x(i-1)+x(i+1)\big) - \frac{1}{16}\big(x(i-2)+x(i+2)\big) \\ &\quad - \frac{1}{32}\big(x(i-3)+x(i+3)\big) + \frac{1}{32}\big(x(i-4)+x(i+4)\big) \end{aligned} \tag{4.24}$$

$$\begin{aligned} high_A(i) &= \frac{9}{8}x(i) - \frac{9}{16}\big(x(i-1)+x(i+1)\big) - \frac{1}{16}\big(x(i-2)+x(i+2)\big) \\ &\quad + \frac{1}{16}\big(x(i-3)+x(i+3)\big) \end{aligned} \tag{4.25}$$

$$\begin{aligned} low_B(i) &= \frac{5}{8}x(i) + \frac{1}{4}\big(x(i-1)+x(i+1)\big) - \frac{3}{32}\big(x(i-2)+x(i+2)\big) \\ &\quad + \frac{1}{32}\big(x(i-4)+x(i+4)\big) \end{aligned} \tag{4.26}$$

$$\begin{aligned} high_B(i) &= \frac{9}{8}x(i) - \frac{9}{16}\big(x(i-1)+x(i+1)\big) - \frac{1}{16}\big(x(i-2)+x(i+2)\big) \\ &\quad + \frac{1}{16}\big(x(i-3)+x(i+3)\big) \end{aligned} \tag{4.27}$$

$$\begin{aligned} low_C(i) &= \frac{23}{32}x(i) + \frac{1}{4}\big(x(i-1)+x(i+1)\big) - \frac{1}{8}\big(x(i-2)+x(i+2)\big) \\ &\quad + \frac{1}{64}\big(x(i-4)+x(i+4)\big) \end{aligned} \tag{4.28}$$

$$high_C(i) = x(i) - \frac{9}{16}\big(x(i-1)+x(i+1)\big) + \frac{1}{16}\big(x(i-3)+x(i+3)\big) \tag{4.29}$$

The original 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.

4.4.4 Hardware Architectures

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

- Observe in Tables 4.2, 4.3, and 4.4 that the coefficients of $x(i \pm k)$ are the same. Thus they can be grouped together to reduce the hardware complexity:

$$w_0 = x(0) \tag{4.30}$$

$$w_1 = x(i-1) + x(i+1) \tag{4.31}$$

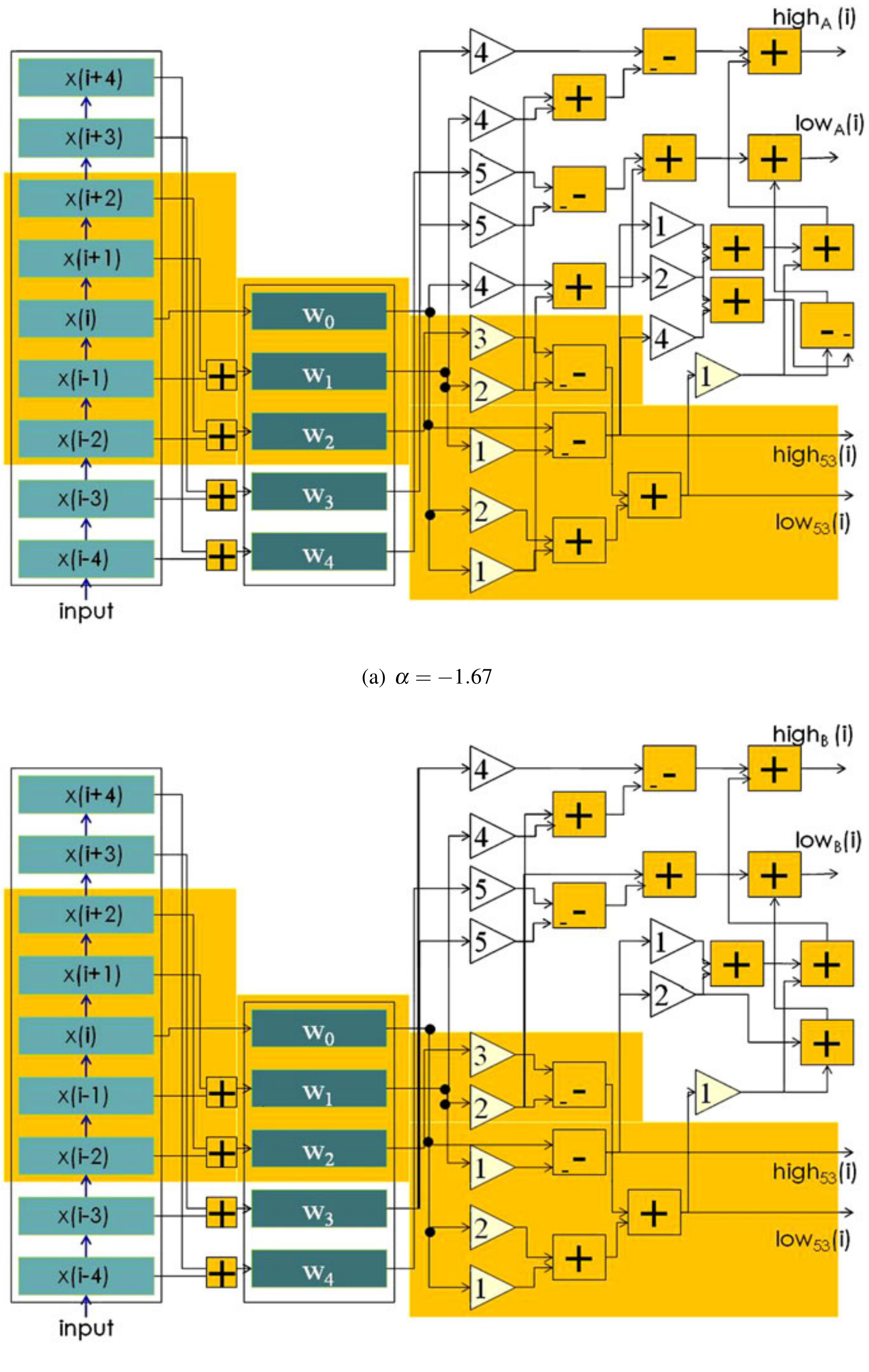
$$w_2 = x(i-2) + x(i+2) \tag{4.32}$$

$$w_3 = x(i-3) + x(i+3) \tag{4.33}$$

  The Daubechies 9/7 filter requires nine 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 (4.12) and (4.13) 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 4.3 and 4.4 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.
- The input stream was pipelined. Thus, as shown in Fig. 4.6 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 help 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.

Figures 4.6(a)–(c) provide 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. 4.6, our Le Gall's 5/3 filter implementation requires only six adder/subtracter units. Our 9/7 filter implementations for $\alpha = -1.67$ required 19 adders. For $\alpha = -1.8$, our design requires 17 adder/subtracter units. But we observe that the design for $\alpha = -2$ requires only 12 adder/subtracter units. This is a significant improvement over any reported existing work as reported in the experiment section.

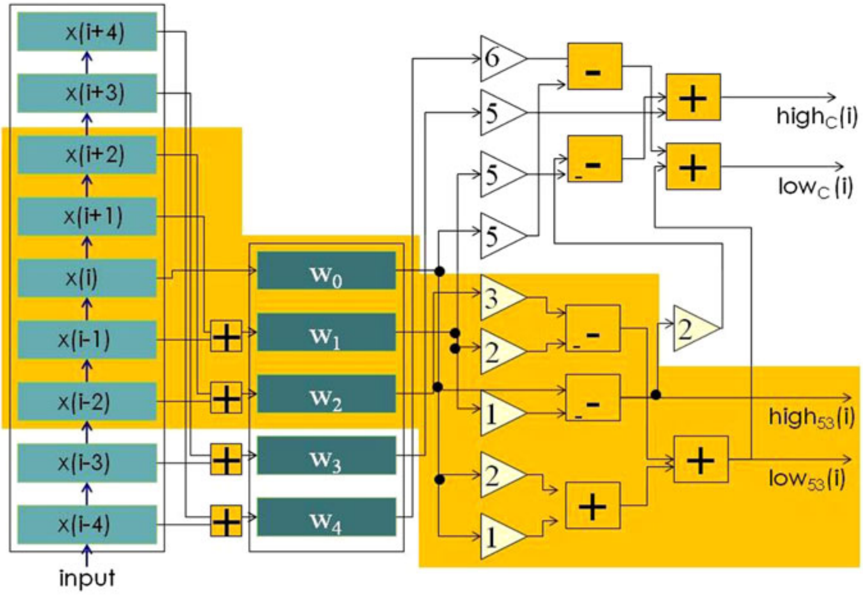
As described in Fig. 4.1, the reconfigurable implementation must allow dynamic switching between wavelet filters. Our implementation allows for easy enabling and disabling of the extra hardware to obtain the choice between a more power-efficient binary 5/3 filter versus a more compression-efficient 9/7 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 output of this filter ($low(i)$ and $high(i)$, respectively) are obtained by convolution of $x(i)$ with $h_0(i)$ and $h_1(i)$,



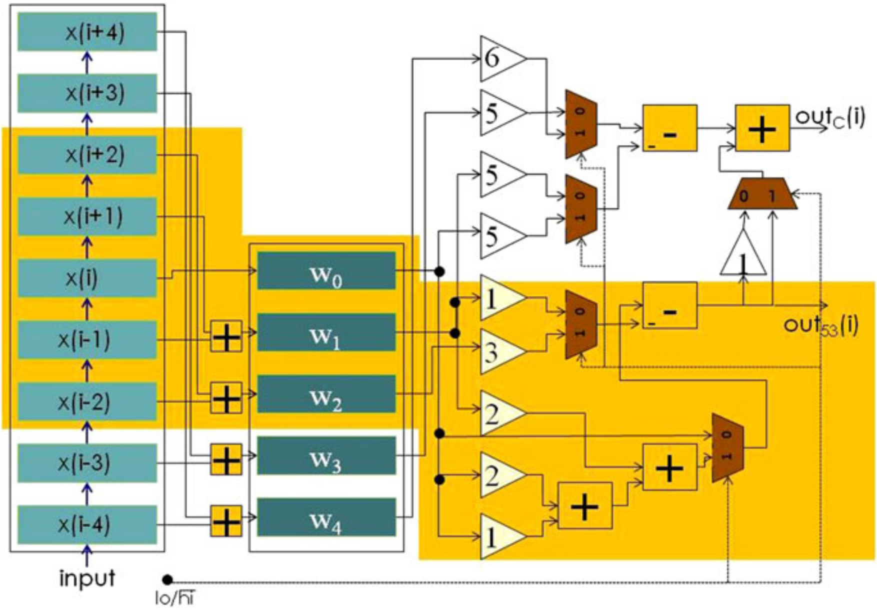
(a) $\alpha = -1.67$

(b) $\alpha = -1.8$

Fig. 4.6 Hardware architectures for bi-orthogonal 9/7 filter



(c) $\alpha = -2$



(d) $\alpha = -2$ (folded)

Fig. 4.6 (Continued)

respectively:

$$low(i) = \sum_{k=-4}^{k=4} h_0(k) \cdot x(i-k) \tag{4.34}$$

$$high(i) = \sum_{k=-3}^{k=3} h_1(k) \cdot x(i-k) \tag{4.35}$$

Substituting the values of filter coefficients from Tables 4.2, 4.3, and 4.4, we can factorize our 9/7 filter coefficients in terms of 5/3 filter output:

$$\begin{aligned} low_A(i) &= 1/2 \cdot low_{53}(i) - (1/4 + 1/16) \cdot high_{53}(i) \\ &\quad + (1/2 + 1/32) \cdot w_0 + 1/32 \cdot (w_4 - w_3) \end{aligned} \tag{4.36}$$

$$\begin{aligned} high_A(i) &= 1/2 \cdot low_{53}(i) + (1/2 + 1/4) \cdot high_{53}(i) \\ &\quad - (1/4 + 1/16) \cdot w_1 + 1/16 \cdot w_3 \end{aligned} \tag{4.37}$$

$$\begin{aligned} low_B(i) &= 1/2 \cdot low_{53}(i) + 1/4 \cdot high_{53}(i) \\ &\quad + 1/32 \cdot (w_4 - w_3) \end{aligned} \tag{4.38}$$

$$\begin{aligned} high_B(i) &= 1/2 \cdot low_{53}(i) + (1/2 + 1/4) \cdot high_{53}(i) \\ &\quad - (1/4 + 1/16) \cdot w_1 + 1/16 \cdot w_3 \end{aligned} \tag{4.39}$$

$$low_C(i) = low_{53}(i) - 1/32 \cdot w_0 + 1/64 \cdot w_4 \tag{4.40}$$

$$\begin{aligned} high_C(i) &= 1/2 \cdot high_{53}(i) - 1/32 \cdot w_1 \\ &\quad + 1/32 \cdot w_3 \end{aligned} \tag{4.41}$$

Figures 4.6(a)–(c) provide 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 output can be $low_{A/B/C}(i)$ and $high_{A/B/C}(i)$, or $low_{53}(i)$ and $high_{53}(i)$ 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. 4.6(d). This design requires only nine adders in the circuit.

4.5 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 and Hauck [6] 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 Chang and Hauck [6] 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. Fry and Hauck [13] 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

(eight bits to represent integer and zero 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. Chang and Hauck [6] 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.

4.6 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 a 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 work, 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 4.7 gives the architecture design of Poly-DWT kernel to achieve these trade-offs.

4.6.1 'On-the-Fly' Switching

We first consider 'on-the-fly' switching of filter designs from 9/7 to 5/3 architectures. The $switch_{\mathrm{hw}}$ signal in Fig. 4.7 is used to switch between 5/3 and 9/7 architectures. The two multiplexers (unshaded in Fig. 4.7) 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:

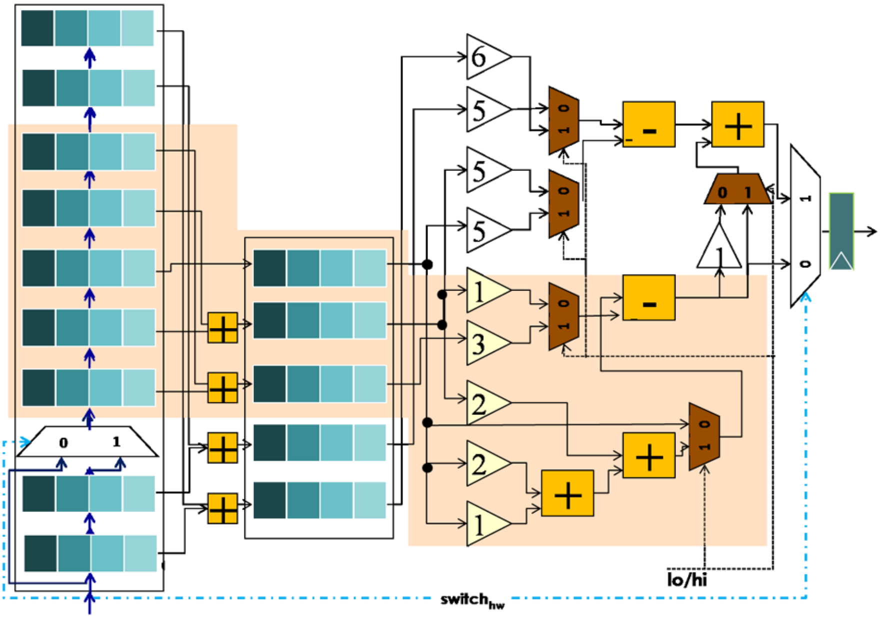


Fig. 4.7 Architectural details of Poly-DWT to facilitate ‘reconfiguration’

1. *Type A hardware*. The hardware common to both 5/3 and 9/7 filter architectures is called 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 Fig. 4.7.
2. *Type B hardware*. The hardware used by 9/7 filter architecture which is obsolete to 5/3 filter is called type B hardware. This hardware is switched off when $switch_{\mathrm{hw}}$ signal is changed to 0. This corresponds to the registers and adders in the unshaded portion of design in Fig. 4.7.

The following steps are involved in switching from 9/7 to 5/3 filters (the 5/3 filter hardware is shaded in Fig. 4.6):

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 input. 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 $switch_{\mathrm{hw}}$ from 1 to 0 (shown in Fig. 4.8(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

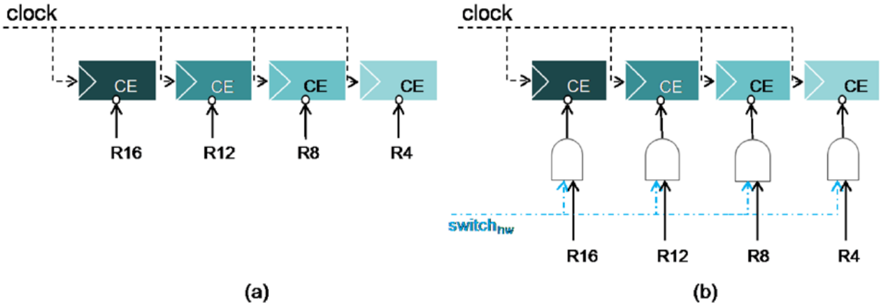


Fig. 4.8 Register level details to enable reconfiguration (**a**) type A architecture and (**b**) type B architecture

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 nanoseconds, 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 one 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.

4.6.2 '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, an $N = 16$ bits register is represented as four 4-bit registers. As shown in Fig. 4.7, the registers are represented as four 4-bit registers. Figure 4.8 explains the working of 'bit-width' switching scheme with individual registers. The four signals R4, R8, R12 and R16 are used to switch the registers on or off at run-time. When R4, R8, and R12 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 Fig. 4.8. Similar changes can be made to the design of adders to partially switch off the LUTs corresponding to an adder hardware. Figure 4.8(a) explains the bit-width switching of type A hardware. The two inputs $switch_{\mathrm{hw}}$ and the register select input (R4/R8/R12/R16) 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 Fig. 4.8(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.

4.7 Experiments

This section presents quantitative results for the performance of Poly-DWT architecture presented in this work. We evaluate our approach on the Xilinx Virtex-V XC5VLX30 FPGA by generating the different DWT architectures. The polymorphic architecture presented in this work 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 three channels, with eight 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.

4.7.1 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 5/3 filter as illustrated in Fig. 4.9(a). It can be seen in the figure that Poly-DWT 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.

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 Fig. 4.9 illustrates 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

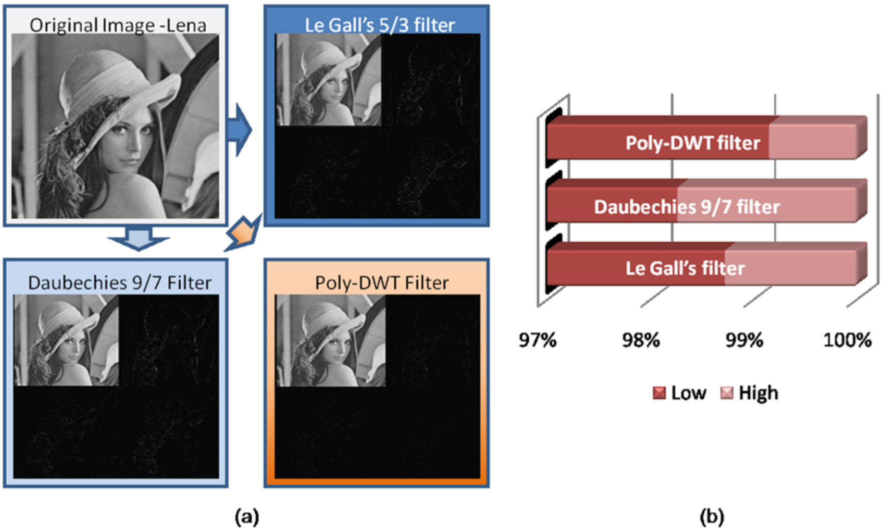


Fig. 4.9 (**a**) Results of one level of DWT and (**b**) energy decomposition by respective filters

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 [30]. We tested the performance on an open-source filter bank-based implementation provided by [38]. 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 4.5. 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 [21] illustrates that our design requires a fewer number of adders and gives a higher compression performance as evident by higher PSNR values.

4.7.2 *Hardware vs. Software Performance*

The hardware performance of DWT kernels proposed in the work was compared with a software-based implementation on the PC platform. Table 4.6 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

Table 4.5 Image compression performance on SPIHT coder (PSNR values)

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

Table 4.6 Hardware acceleration on a Virtex-5 XC5VLX30 FPGA (time in µs)

Image	Le Gall 5/3 filter			Daubechies 9/7 filter			Poly-DWT 9/7 filter	
	SW	HW	Speedup	SW	HW	Speedup	HW	Speedup
CIF	1420	197	7.06×	2980	330	9.03×	288	10.35×
Q-CIF	370	68	5.45×	790	91	8.68×	77	10.26×

times for one level of DWT for different image sizes is presented in Table 4.6. 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 [9] is used for data I/O operations.

The results as summarized in Table 4.6 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 four bits word length). The actual speedup achieved by the Poly-DWT kernel (Table 4.7) over Daubechies filter is greater (three times more) than the results indicated in Table 4.5 because of memory access computations involved in image compression results.

4.7.3 Hardware Comparison

Direct implementation of the CDF-9/7 filter gave a clock frequency of 107 MHz, while requiring nine multiplier units. A clock frequency of 110 MHz was reported when we forced the design to map the constant multiplications into Lookup Tables. Martina and Masera [21] implement Daubechies 9/7 filter with approximate coefficients and reports a clock frequency of about 200 MHz through a multiplier-free implementation, targeting 0.13 µm VLSI technology.

Table 4.7 Comparison of binary filter features and hardware resources requirements

Features	Daub. 9/7	$\alpha = -2$ folded	$\alpha = -2$	$\alpha = -1.8$	$\alpha = -1.67$	Tay, 2001	Kotteri, 2005	Huang, 2001	Martina, 2007	Martina, 2005
Adders	15	9	12	17	19	19	15	8	19	21
Multipliers	9	0	0	0	0	0	0	4	0	0
PSNR	A	B	B	A	A	C	C	A	B	B
Reconf.	N	Y	Y	Y	Y	N	N	N	Y	N
Registers	144	208	213	253	294	–	–	–	–	–
LUTs	80	175	194	217	289	–	–	–	–	–
Bit slices	210	245	259	311	375	–	–	–	–	–
Clock (MHz)	107	389	317	311	310	–	–	–	200	–

Table 4.7 summarizes the performance of our Xilinx Virtex-V implementation, and compares our results with other recent work. 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 critical path. The critical path of the circuit lies from the w_i registers to the final output $low_C(i)$ or $high_C(i)$, passing through signals $low_C(i)$ or $high_C(i)$. 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 $\alpha = -1.8$ architecture 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 × 1080) in an estimated 5 ms time. As previously mentioned, the shaded (yellow) regions in Fig. 4.6 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 eight 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 4.7. Here, a comparison of hardware resources utilization is provided against existing work. Martina and Masera [21] 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

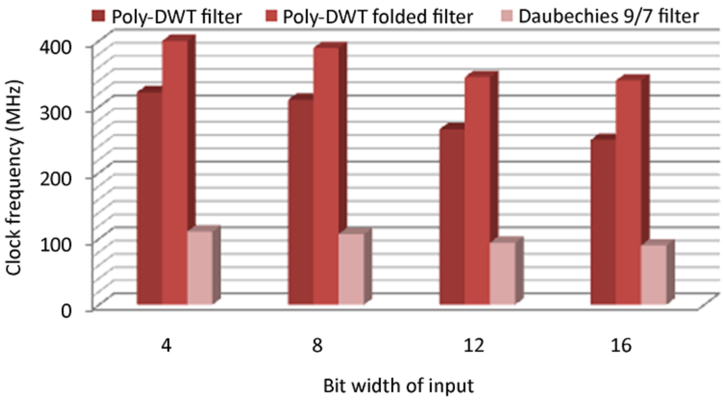


Fig. 4.10 Change in FPGA clock frequency (MHz) for variable word widths for various filters

leads to its poor PSNR performance. Our architecture provides both more efficient hardware usage and better compression performance.

Figure 4.10 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 Fig. 4.10 leads to reduction of clock frequency and hence reduced throughput.

Zhang et al. [45] 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 two frames (at CCIR resolution of 720 × 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).

4.7.3.1 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 45 nm cell library [34]. 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 4.8). We were able to achieve a clock speed of 500 MHz for the folded 9/7 filter design.

Table 4.8 Performance comparison using standard cell libraries

	Poly-DWT	Le Gall's	Daub. 9/7	[21][a]
Area	2135	1370	6693	–
Cells	544	194	1022	–
Clock frequency	500	500	300	200

[a] Martina and Masera [21] report a gate count of 2.68 K using a 130 nm cell library

4.7.4 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 were kept at 8.0 format and the word width of internal registers was changed. Figure 4.11 shows the change in reconstruction quality of the images depending on changes in hardware resources (single bit registers or flip-flops). The x axis here refers to the total number of bits given to an internal processing register. Figure 4.12 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.

4.7.5 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 on the bridge.

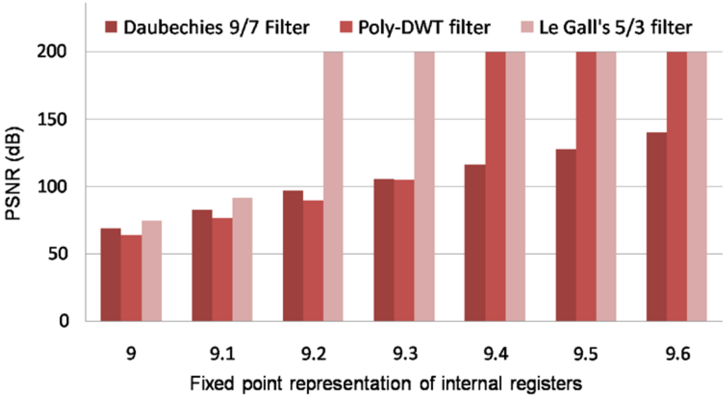


Fig. 4.11 Plot of PSNR vs. the number of bits alloted for internal registers

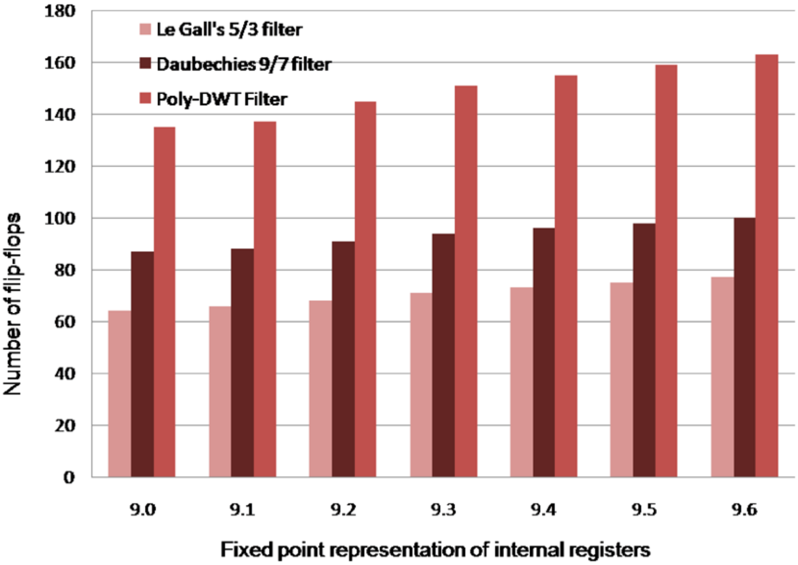


Fig. 4.12 Comparison of register usage for the binary filter implementations

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 0.34 W for the 5/3 filter and 0.46 W 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 0.12 W power. The respective values were 0.0477 W for 5/34 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 $\frac{0.46}{0.46\cdot 0.99+0.34\cdot 0.01} = 1.348$ (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 those data are 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.

4.8 Conclusions and Future Work

This work 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 output 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.

References

1. Adams, M.D., Kossentini, F.: JasPer: a software-based JPEG-2000 codec implementation. In: Proc. IEEE Intl. Conf. Image Processing, ICIP 2000, vol. 2, pp. 53–56 (2000). doi:10.1109/ICIP.2000.899223

2. Alam, M., Rahman, C., Badawy, W., Jullien, G.: Efficient distributed arithmetic based DWT architecture for multimedia applications. In: Proc. Int. Work. SoC for Real Time Applications, pp. 333–336 (2003)
3. Ansari, R., Guillemot, C., Kaiser, J.: Wavelet construction using Lagrange halfband filters. IEEE Trans. Circuits Syst. **38**(9), 1116–1118 (1991)
4. Benkrid, A., Benkrid, K., Crookes, D.: Design and implementation of a generic 2D orthogonal discrete wavelet transform on FPGA. In: Proc. IEEE Symp. Field-Programmable Custom Computing Machines (FCCM), pp. 162–172 (2003)
5. Benkrid, A., Crookes, D., Benkrid, K.: Design and implementation of a generic 2D biorthogonal discrete wavelet transform on an FPGA. In: Proc. IEEE Symp. Field-Programmable Custom Computing Machines (FCCM), pp. 190–198 (2001)
6. Chang, M., Hauck, S.: Automated least-significant bit datapath optimization for FPGAs. In: Proc. IEEE Symp. Field-Programmable Custom Computing Machines (FCCM), pp. 59–67 (2004)
7. Choi, S.-J., Woods, J.W.: Motion-compensated 3-d subband coding of video. IEEE Trans. Image Process. **8**(2), 155–167 (1999). doi:10.1109/83.743851
8. Christopoulos, C., Skodras, A., Ebrahimi, T.: The JPEG2000 still image coding system: an overview. IEEE Trans. Consum. Electron. **46**(4), 1103–1127 (2000)
9. Chrysafis, C., Ortega, A.: Line based reduced memory, wavelet image compression. In: Proc. of Data Compression Conference, DCC'98, pp. 398–407 (1998). doi:10.1109/DCC.1998.672177
10. Claus, C., Stechele, W., Kovatsch, M., Angermeier, J., Teich, J.: A comparison of embedded reconfigurable video-processing architectures. In: Proc. IEEE Int. Conf. on Field Programmable Logic and Applications, FPL 2008, pp. 587–590 (2008). doi:10.1109/FPL.2008.4630015
11. Claus, C., Zhang, B., Stechele, W., Braun, L., Hubner, M., Becker, J.: A multi-platform controller allowing for maximum Dynamic Partial Reconfiguration throughput. In: Proc. IEEE Int. Conf. on Field Programmable Logic and Applications, FPL 2008, pp. 535–538 (2008). doi:10.1109/FPL.2008.4630002
12. Eeckhaut, H., Devos, H., Lambert, P., de Schrijver, D., van Lancker, W., Nollet, V., Avasare, P., Clerckx, T., Verdicchio, F., Christiaens, M., Schelkens, P., van de Walle, R., Stroobandt, D.: Scalable, wavelet-based video: from server to hardware-accelerated client. IEEE Trans. Multimed. **9**(7), 1508–1519 (2007)
13. Fry, T.W., Hauck, S.A.: SPIHT image compression on FPGAs. IEEE Trans. Circuits Syst. Video Technol. **15**(9), 1138–1147 (2005). doi:10.1109/TCSVT.2005.852625
14. Le Gall, D., Tabatabai, A.: Sub-band coding of digital images using symmetric short kernel filters and arithmetic coding techniques. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP), pp. 761–764 (1988)
15. Huang, C.-T., Tseng, P.-C., Chen, L.-G.: VLSI architecture for discrete wavelet transform based on B-spline factorization. In: Proc. IEEE Work. Signal Processing Systems, SIPS 2003, pp. 346–350 (2003)
16. Koch, D., Beckhoff, C., Teich, J.: ReCoBus-Builder—a novel tool and technique to build statically and dynamically reconfigurable systems for FPGAs. In: Proc. IEEE Int. Conf. on Field Programmable Logic and Applications, FPL 2008, pp. 119–124 (2008). doi:10.1109/FPL.2008.4629918
17. Kotteri, K., Barua, S., Bell, A., Carletta, J.: A comparison of hardware implementations of the biorthogonal 9/7 DWT: convolution versus lifting. IEEE Trans. Circuits Syst. II **52**(5), 256–260 (2005)
18. Leeser, M., Miller, S., Haiqian, Y.: Smart camera based on reconfigurable hardware enables diverse real-time applications. In: Proc. IEEE Symp. Field-Programmable Custom Computing Machines (FCCM), pp. 147–155 (2004)
19. Lian, C.-J., Chen, K.-F., Chen, H.-H., Chen, L.-G.: Analysis and architecture design of block-coding engine for EBCOT in JPEG 2000. IEEE Trans. Circuits Syst. Video Technol. **13**(3), 219–230 (2003). doi:10.1109/TCSVT.2003.809833

20. Martina, M., Masera, G.: Low-complexity, efficient 9/7 wavelet filters implementation. In: Proc. IEEE Int. Conf. Image Processing (ICIP) (2005)
21. Martina, M., Masera, G.: Multiplierless, folded 9/7–5/3 wavelet VLSI architecture. IEEE Trans. Circuits Syst. II **54**(9), 770–774 (2007)
22. Mittal, A., Pande, A., Verma, P.K.: Content-based network resource allocation for mobile engineering laboratory applications. In: Proc. Int. Conf. Mobile Learning, pp. 146–152 (2007)
23. Pande, A., Zambreno, J.: Polymorphic wavelet architecture over reconfigurable hardware. In: IEEE Int. Conf. on Field Programmable Logic and Applications, pp. 471–474 (2008)
24. Pande, A., Verma, A., Mittal, A., Agrawal, A.: Network aware efficient resource allocation for mobile-learning video systems. In: Proc. Int. Conf. Mobile Learning, pp. 189–196 (2007)
25. Pande, A., Zambreno, J.: Design and analysis of efficient reconfigurable wavelet filters. In: Proc. IEEE Int. Conf. Electro Information Technology, pp. 337–342 (2008)
26. Paulsson, K., Hubner, M., Becker, J.: Exploitation of dynamic and partial hardware reconfiguration for on-line power/performance optimization. In: Proc. IEEE Int. Conf. Field Programmable Logic and Applications, FPL 2008, pp. 699–700 (2008). doi:10.1109/FPL.2008.4630044
27. Qiu, R., Yu, W.: An efficient quality scalable motion-JPEG2000 transmission scheme. Technical Report WUCS-01-37, Department of Computer Science, Washington University in St. Louis (November 2001). citeseer.ist.psu.edu/qiu01efficient.html
28. Redmill, D., Bull, D., Martin, R.: Design of multiplier free linear phase perfect reconstruction filter banks using transformations and genetic algorithms. In: Proc. Int. Conf. Image Processing and Its Applications (1997)
29. Ritter, J., Molitor, P.: A pipelined architecture for partitioned DWT based lossy image compression using FPGAs. In: Proc. Int. Symposium on Field Programmable Gate Arrays (FPGA), pp. 201–206 (2001)
30. Said, A., Pearlman, W.: An image multiresolution representation for lossless and lossy image compression. IEEE Trans. Image Process. **5**, 1303–1310 (1996)
31. Schwarz, H., Marpe, D., Wiegand, T.: Overview of the scalable video coding extension of the H.264/AVC Standard. IEEE Trans. Circuits Syst. Video Technol. **17**(9), 1103–1120 (2007). doi:10.1109/TCSVT.2007.905532
32. Shapiro, J.: Embedded image coding using zerotrees of wavelet coefficients. IEEE Trans. Signal Process. **41**(12), 3445–3462 (1993)
33. Skodras, A., Christopoulos, C., Ebrahimi, T.: The JPEG 2000 still image compression standard. IEEE Signal Process. Mag. **18**(5), 36–58 (2001). doi:10.1109/79.952804
34. Stine, J., Castellanos, I., Wood, M., Henson, J., Love, F., Davis, W.R., Franzon, P.D., Bucher, M., Basavarajaiah, S., Oh, J., Jenkal, R.: FreePDK: an open-source variation-aware design kit. In: IEEE Int. Conf. on Microelectronic Systems Education, MSE 2007, pp. 173–174 (2007). doi:10.1109/MSE.2007.3
35. Stroobandt, D., Eeckhaut, H., Devos, H., Christiaens, M., Verdicchio, F., Schelkens, P.: Reconfigurable hardware for a scalable wavelet video decoder and its performance requirements. In: Computer Systems: Architectures, Modeling, and Simulation. Lecture Notes in Computer Science, vol. 3133, pp. 203–212. Springer, Berlin (2004)
36. Taubman, D.: High performance scalable image compression with EBCOT. IEEE Trans. Image Process. **9**(7), 1158–1170 (2000)
37. Tay, D.: Rationalizing the coefficients of popular biorthogonal wavelet filters. IEEE Trans. Circuits Syst. Video Technol. **10**(6), 998–1005 (2000)
38. Tian, J.: SPIHT coder. http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=4808&objectType=file (2008)
39. Tseng, P., Chang, Y., Huang, Y., Fang, H., Huang, C., Chen, L.: Advances in hardware architectures for image and video coding—a survey. Proc. IEEE **93**(1), 184–197 (2005). doi:10.1109/JPROC.2004.839622
40. Verma, P.K., Pande, A., Mittal, A., Kumar, P.: Content-based network adaptive wireless transmission of remote surveillance video. In: National Conf. on Communications, India (2008)

41. Verma, P.K., Mittal, A., Kumar, P.: Fusion of thermal infrared and visible spectrum video for robust surveillance. In: ICVGIP, pp. 528–539 (2006)
42. Vetterli, M., Kovačevic, J.: Wavelets and Subband Coding. Prentice-Hall, Upper Saddle River (1995)
43. Villasenor, J., Belzer, B., Liao, J.: Wavelet filter evaluation for image compression. IEEE Trans. Image Process. **4**(8), 1053–1060 (1995)
44. Yang, W., Lu, Y., Wu, F., Cai, J., Ngan, K.N., Li, S.: 4-D wavelet-based multiview video coding. IEEE Trans. Circuits Syst. Video Technol. **16**(11), 1385–1396 (2006). doi:10.1109/TCSVT.2006.884571
45. Zhang, X., Rabah, H., Weber, S.: Auto-adaptive reconfigurable architecture for scalable multimedia applications. In: Second NASA/ESA Conference on Adaptive Hardware and Systems, AHS 2007, pp. 139–145 (2007). doi:10.1109/AHS.2007.34

Chapter 5
The Secure Wavelet Transform

Abstract 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 work, we propose a lightweight 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 an 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 45 nm CMOS technology. SWT introduces parametrization of a video compression operation for video encryption by generating a parametric key for encryption. Implementations over FPGA and VLSI technology are both presented.

5.1 Introduction

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).

A. Pande, J. Zambreno, *Embedded Multimedia Security Systems*,
DOI 10.1007/978-1-4471-4459-5_5, © Springer-Verlag London 2013

The computer security protocols (e.g. SSL [36], TLS [4]) and cryptographic ciphers (e.g. AES [10], DES [11], IDEA [18]) 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 for short SRTP [2], is an application of the naive approach. In SRTP, multimedia data are 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 [34]) 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 [14]). The partial or selective encryption schemes [21] encode only the important parts of multimedia data to reduce these computational overheads.

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 [3, 9, 19, 21, 26] to reduce the computational requirements of full encryption schemes. Lian et al. [19] present a scheme for encryption of Discrete Cosine Transform (DCT) coefficients' signs and watermarking of DCT coefficients. Lian et al. [19] use Exp-Goloumb codes for the encryption operation. Cheng and Li [6] 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 [21].

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 [5]. The authors in [5] 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 352×288 pixels). Adding a sequential or pipelined encryption stage to the system in [5] 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. Grangetto et al. [12] 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. Kim et al. [17] present a variation of [12] that improves the security performance of parameterized arithmetic coding scheme but increases the complexity in hardware implementation.

Mao and Wu [24] 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. The scheme gives good results, however, it requires extra computations (and hence extra hardware resources) to implement such a scheme. Lian and Wang [20] present 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 overhead. 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 realtime multimedia encryption [38]. 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 [37].

This work 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 work. The initial results regarding parameterized construction of the DWT were presented in [30].

Section 5.2 gives the theory and mathematical preliminaries of the proposed SWT architecture. Section 5.3 discusses the image security provided by the SWT. In Sect. 5.4 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 5.5 concludes the work with insight of future work.

5.2 Preliminaries

Prior work in signal processing establishes that the 1-D DWT can be viewed as a signal decomposition using specific low-pass and high-pass filters [33]. 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 2 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 [7], 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.

5.2.1 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. We have

$$G_1(z) = z^{-1} H_2(-z), \qquad G_2(z) = z H_1(-z)$$

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

$$H_1(z) H_2(z) + H_1(-z) H_2(-z) = 2$$

Liu and Zheng [22] present a parameterized construction of Bi-orthogonal Wavelets Filter Banks (typically used for image compression). For an even number of vanishing moments, $H_1(z)$ and $H_2(z)$ are represented as follows:

$$\begin{aligned} H_1(z) &= \left(z^{-\frac{1}{2}} + z^{\frac{1}{2}}\right)^{2l_1} \left(\alpha + (1-\alpha)\left(z^{\frac{1}{2}} + z^{\frac{1}{2}}\right)^2\right) \\ H_2(z) &= \left(z^{-\frac{1}{2}} + z^{\frac{1}{2}}\right)^{2l_2} Q(z) \end{aligned}$$

where

$$Q(z) = \sum_{n=0}^{3} q_n \left(z^{\frac{1}{2}} + z^{\frac{1}{2}}\right)^{2n}, \quad l_1, l_2 \geq 0, \ \{l_1, l_2\} \in Z$$

and α is the free parameter introduced in the design. The values q_n are calculated by the following expression:

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

and

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

with $L = l_1 + l_2$.

Substituting these values, the perfect reconstruction condition can be shown to form a Bezout's identity. In the theory of polynomial rings, this is one specific case of Diophantine equation. Since $\alpha \notin 0, 1$, this equation is found to have infinite number of polynomial solutions (corresponding to infinite construction of wavelets). For low taps filters, the construction is restricted to least degree [22]. For the 9/7 filter, $l_1 = 2$, $l_2 = 1$ and $L = 3$ the values of q_n were approximated using Taylor's series expansion and obtained as follows:

$$q_0 = 1, \qquad q_1 = 5 - 2\alpha, \qquad q_2 = 4\alpha^2 - 14\alpha + 16$$

$$q_3 = 36\alpha - 8\alpha^2 - 60 + 32/\alpha$$

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

$$\begin{aligned} H_1(z) = {} & \left(-9\alpha/64 + \alpha^2/32 + 15/64 - 1/(8\alpha)\right)\left(z^4 + 1/z^4\right) \\ & + \left(-\alpha^2/16 + 11\alpha/32 - 11/16 + 1/(2\alpha)\right)\left(z^3 + 1/z^3\right) \\ & + \left(1/8 - 1/(2\alpha)\right)\left(z^2 + 1/z^2\right) \\ & + \left(-11\alpha/32 + \alpha^2/16 + 15/16 - 1/(2\alpha)\right)(z + 1/z) \\ & + \left(9\alpha/32 - \alpha^2/16 - 7/32 + 5/(4\alpha)\right) \end{aligned}$$

$$\begin{aligned} H_2(z) = {} & (1/32 - \alpha/32)\left(z^3 + 1/z^3\right) + (1/8 - \alpha/16)\left(z^2 + 1/z^2\right) \\ & + (7/32 + \alpha/32)(z + 1/z) + (1/4 + \alpha/8) \end{aligned}$$

There are several useful features of parameterized DWT construction that make it suitable for being a part of the SWT.

5.2.1.1 Binary Coefficients

Although not a subject of our discussion in this section, we would briefly mention generating binary filters using this approach. A number of families of DWT filters can be generated by varying the values l_1 and l_2. By setting the parameter α to be of the form $k/2^n$ (k and n are integers), it turns out that we can generate a number of binary coefficients filters amenable to efficient hardware implementation, as we discussed in Poly-DWT chapter.

The focus here is to find parametric expression for DWT to enable generation of key for encryption and to map it properly to custom hardware. The discussion continues as follows.

5.2.1.2 Rational Coefficients

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

5.2.1.3 Feasible Range of Parameter α

The numerical value of free parameter α can be varied over a wide range while retaining the perfect reconstruction property of the wavelet transform. However, as we vary the value of α 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 α varies in the range 1 to 3. When α 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 5.1 illustrates the significant decline in PSNR values (in dB) for $\alpha > 3$.

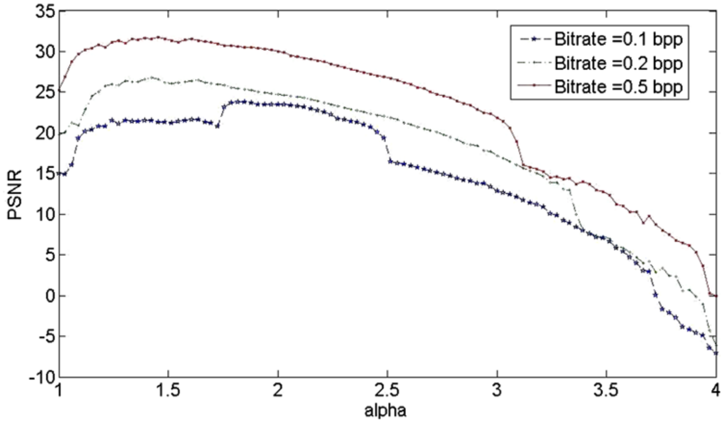


Fig. 5.1 PSNR values (in dB) for image reconstruction using SPIHT coder at different bitrates (in bpp or bits per pixel)

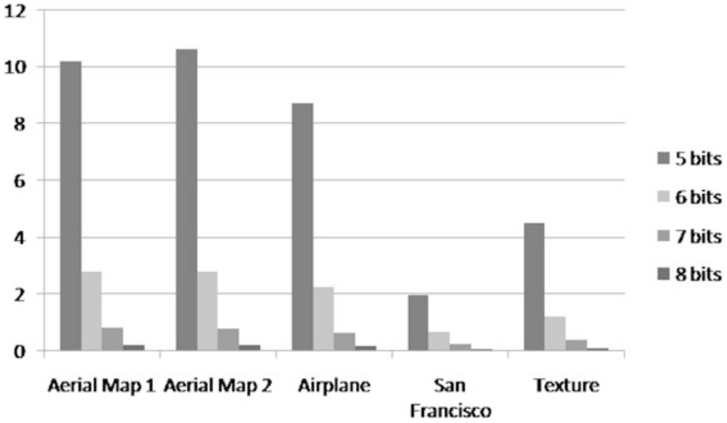


Fig. 5.2 MSE values for sample images with the change in number of bits assigned to one α parameter. The image was encoded with one α value and decoded with adjacent alpha values for various bit-widths of α. 1000 simulations were run to obtain an average value

5.2.1.4 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 α values for all $2N$ 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 a low number of bits. Figure 5.2 shows the MSE (Mean Square Values) for image encoded with one α value and reconstructed with the adjacent α value for various bit-width. It can be seen that five or less bits give a large MSE (MSE > 8) while some applications may allow $m = 8$.

Figure 5.3 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 α parameter set to 2.0 and decryption (or image reconstruction) was performed with different α values. We can see that the images decrypted with the wrong key values (Figs. 5.3(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 seem that wrong guesses for DWT parameter α leads to high reconstruction errors in images. However, we need further dimensions to increase the key

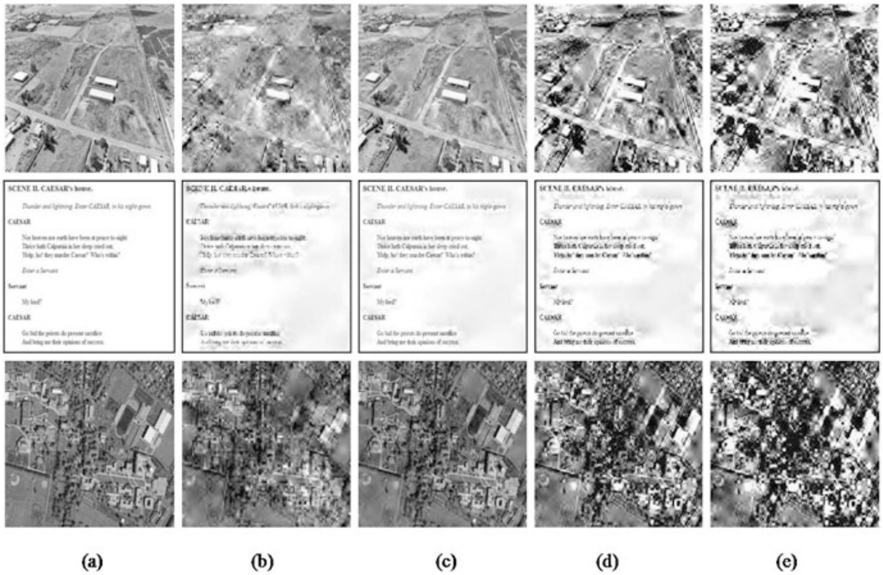


Fig. 5.3 Image reconstruction with different keys. (**a**) Original images which are then encrypted with $\alpha = 2$, (**b**)–(**e**) show reconstruction with $\alpha = 1, 2, 3$ and 3.5, respectively

space and make image reconstruction more obscure in case of wrong guesses for the key value.

5.2.2 Subband Re-orientation

The parent-child coding gain in the DWT-based coders was quantified by [25]. 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.

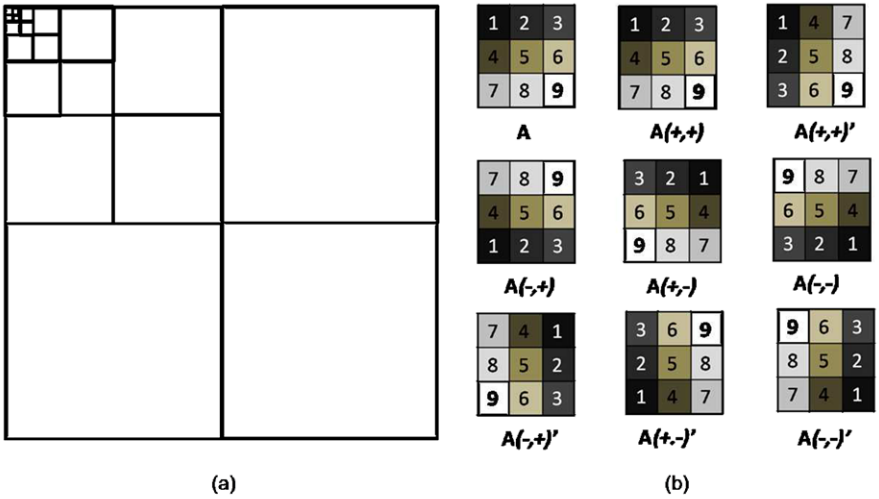


Fig. 5.4 (**a**) Image decomposition with DWT (six levels) leading to 19 subbands. Three bits are assigned for each subband's re-orientation information. (**b**) Possible transpose relationships for subbands. A is the original matrix. The eight permutations are achieved using transpose relationship ('), and reverse-ordering of the subbands ($-$ for reverse, $+$ for forward read access) along both rows and columns

5.2.2.1 Zero Computational Overhead

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

5.2.2.2 Feasible Subband Re-orientations

In Fig. 5.4(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 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.

5.2.2.3 Key Space

Figure 5.4(a) shows the 19 different subbands obtained by a 6-level wavelet decomposition. In general, we obtain $3N + 1$ subbands for a N level wavelet decomposition, each requiring three bits. Thus, we get a key space of $9N + 3$ bits using subband re-orientation.

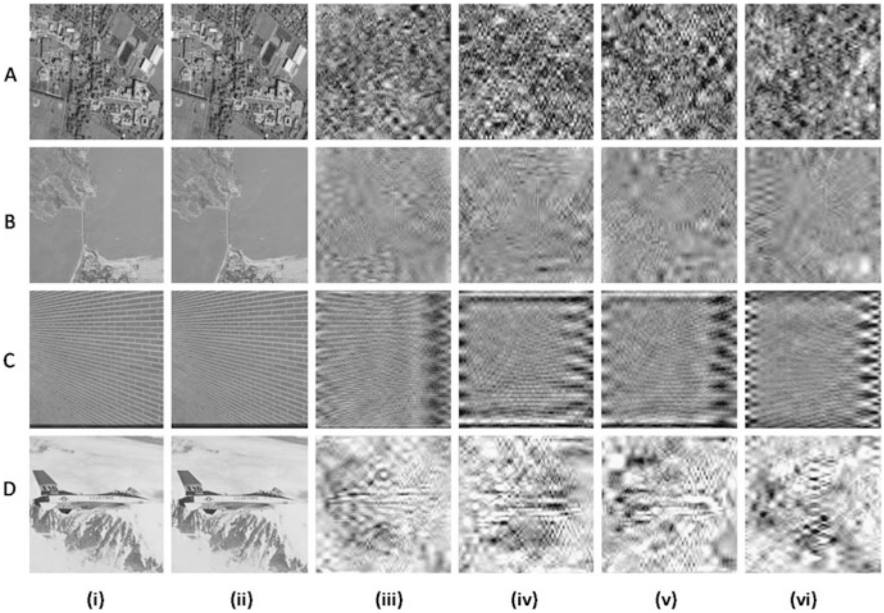


Fig. 5.5 Image reconstruction with different keys. **A** aerial map image, **B** San Francisco Golden gate aerial image, **C** brick wall (texture) image and **D** airplane image. (*i*) Original image encrypted with key0, (*ii*) image decrypted with same key, (*iii*)–(*vi*) image decrypted with randomly generated keys

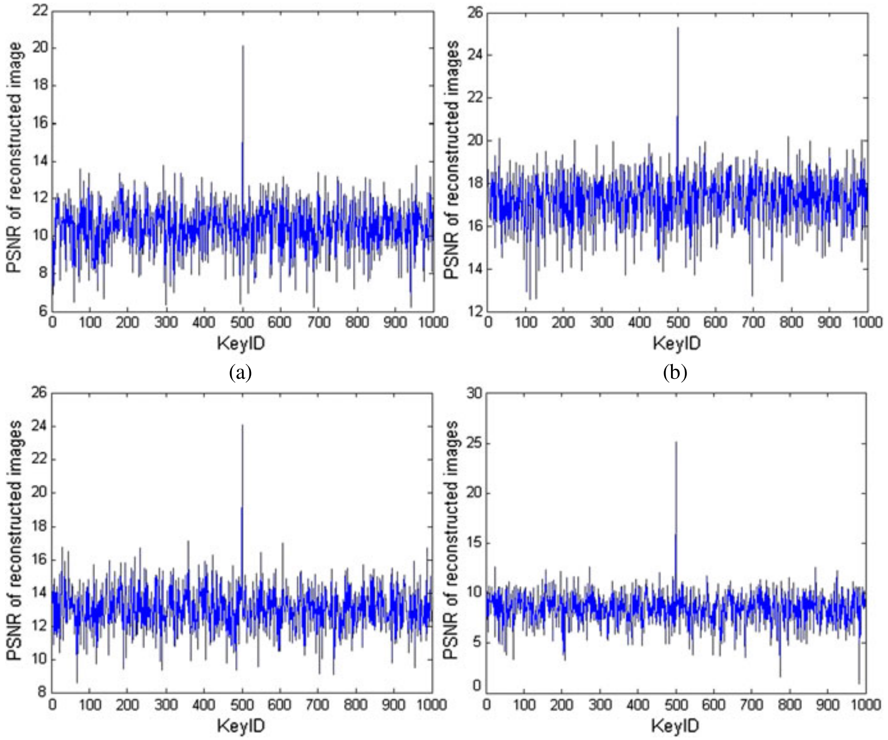
5.3 Security

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 × 512 pixels this upper limit of N ($N_{\max}$) is 9. However, choosing N close to $N_{\max}$ will lead to the innermost subband size being very small.

We selected wavelet decomposition level of $N = 6$ for images of dimension 512 × 512 pixels to allow a standard block size of 8 × 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 work [31], which serves as the foundational treatment of modern cryptography, calls this property the 'confusion' property. Ideally, change in one bit of the key should change the cipher text completely.

Figure 5.5 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 5.1). Figures 5.6(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.



(a) (b)

(c) (d)

Fig. 5.6 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

Table 5.1 PSNR values (in dB) for image reconstruction with various random keys (encoded with key0)

	Key0	Key1	Key2	Key3	Key4
Aerial	∞	12.36	11.17	11.67	11.77
San Francisco	∞	18.40	17.34	18.21	18.46
Brick Wall	∞	14.75	13.39	14.34	13.58
Airplane	∞	13.19	11.26	11.63	12.43

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 such as AES or DES which allow enough mixing between bit values in multiple rounds

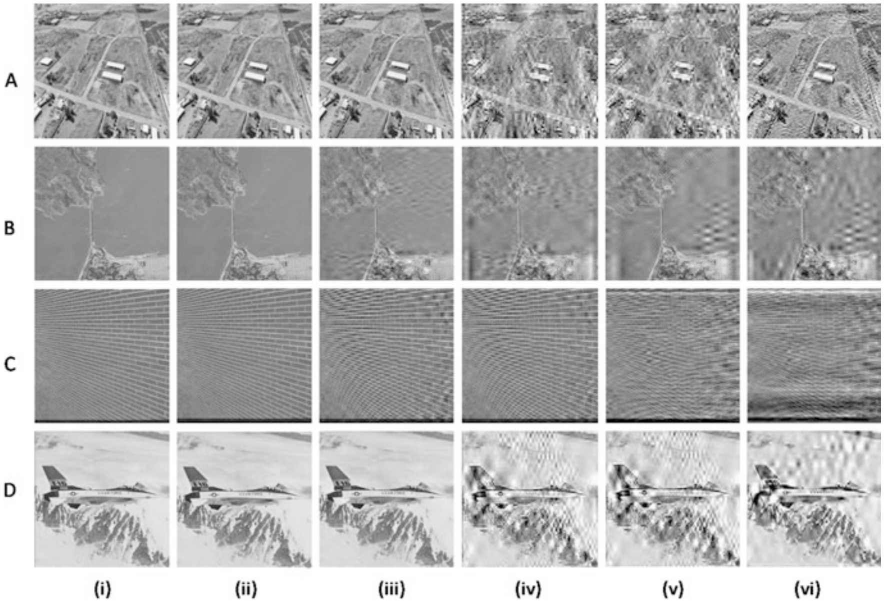


Fig. 5.7 Image reconstruction with different keys. **A** aerial map image, **B** San Francisco Golden gate aerial image, **C** brick wall (texture) image and **D** airplane image. (*i*) Original image encrypted with key0, (*ii*) image decrypted with same key, (*iii*)–(*vi*) image decrypted with Hamming distance of 1, 4, 6 and 8

Table 5.2 Variations in image reconstruction quality (PSNR values) with Hamming distance

Hamming distance	0	1	4	6	8
Aerial	∞	50.3	23	16.04	13.18
San Francisco	∞	36.27	30.98	22.61	21.09
Brick Wall	∞	50.27	37.5895	25.9	23.2
Airplane	∞	44.28	21.64	21.43	16.16

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 5.2. 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. 5.7. Different bit positions in the key have different effects on the image quality degradation. This is attributed to the fact that changing different bit positions in value of α will lead to different degrees of distortions. This attributes to the fact that Fig. 5.7(D)(vi) has

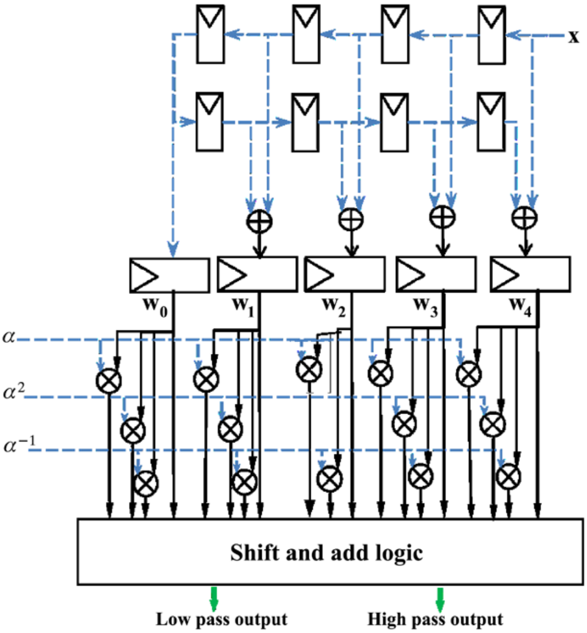


Fig. 5.8 Hardware architecture for THE 1-D SWT filter

less quality degradation compared to Fig. 5.7(D)(v). To quantify the image degradation with increasing $h.d.$, we ran 1000 simulations and recorded the average values in Table 5.2.

5.4 Hardware Implementation

Figure 5.8 gives an overview of the 1-D SWT hardware architecture. The input data (one pixel input per cycle) x are 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 to simplify further computations. In Fig. 5.8, eight of the nine inputs are passed through four adders to reduce the number of inputs to five. These values (labeled w_0, w_1, w_2, w_3 and w_4) are multiplied with α, α^2 and α^{-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.

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 14.

The initial work was presented in [29, 30] and there is a mistake in the count of multipliers reported in them.

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 [32] 45 nm cell library. Under the timing constraints of 500 MHz, the design required 4259 cells and a chip area of 22066 μm^2.

The design used 14 10 × 9 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 5.3. 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 four pipelining stages we obtained a clock frequency of 242 MHz.

5.4.1 Reconfigurable Constant Multiplier (RCM)

The expression for low- and high-pass filter coefficients were obtained in Sect. 5.2.1. It was implemented in Fig. 5.8 using several multiplier units. The $w_i, i \in \{0, 1, 2, 3, 4\}$ values are obtained by summing the inputs for symmetric taps in the SWT implementation as shown in Fig. 5.8. 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

$$H_1(k) = \sum_{i=0}^{4} K_i(a) \cdot w_i(k)$$

and

$$H_0(k) = \sum_{i=0}^{3} \hat{K}_i(a) \cdot w_i(k)$$

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

Table 5.3 Hardware utilization of DWT architecture on Xilinx Virtex XCVLX330 FPGA

	SWT (RCM)	SWT (VLSI)	SWT (FPGA)	[28]	[27]	Daubechies 9/7	[16]	[35]	[15]
Slices flip flops	5580	–	649	245	–	210	–	–	–
Multipliers	0	14	14	0	0	16	12	36	12
Adders	11	41	41	9	19	15	16	36	16
Registers	120	–	92	208	–	144	–	–	–
Critical path	$4T_a + T_l$	$T_m + 5T_a$	$T_m + 5T_a$	$3T_a$	$5T_a$	$T_m + 4T_a$	$T_m + 2T_a$	$T_m + 4T_a$	$4T_m + 8T_a$
Clock frequency (MHz)	114 (np) –	– 500 (p)	60 (np) 243 (p)	120 (np)	200	107	–	–	–
Security	Yes	Yes	Yes	No	No	No	No	No	No

Note: T_m and T_a are the time delay in multiplier and adder circuits, respectively

np: not pipelined

p: pipelined

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. 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 4 × 4 bit RCM to explain the LUT mappings.

5.4.1.1 4 x 4 Bits Multiplier Using LUTs

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

$$P_i = A_i \oplus B_i, \qquad G_i = A_i B_i$$

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 [23], we get

$$\begin{aligned} C_1 &= \text{initial carry} \\ C_2 &= G_1 + P_1 C_1 = A_1 B_1 + (A_1 \oplus B_1) A_0 B_0 \\ C_3 &= G_2 + P_2 G_1 + P_2 P_1 C_1 \\ C_4 &= G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_1 \end{aligned}$$

and

$$\begin{aligned} S_1 &= A_1 \oplus B_1 \oplus C_1 = A_1 \oplus B_1 \oplus A_0 B_0 \\ S_2 &= A_2 \oplus B_2 \oplus C_2 = A_2 \oplus B_2 \oplus \big(A_1 B_1 + (A_1 \oplus B_1) A_0 B_0\big) \ldots \end{aligned}$$

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.

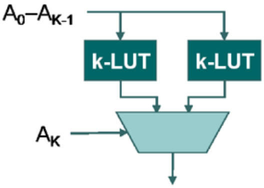


Fig. 5.9 Building a $(K+1)$-LUT from K-LUT

- 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 4×4 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 [1]. 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.

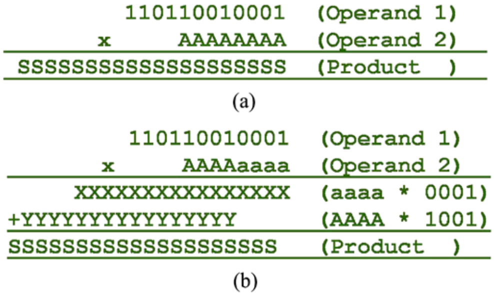
5.4.1.2 Mapping a Generic RCM into LUTs

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 4×4 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 therefore be 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 two K-input LUTs (as shown in Fig. 5.9). For example, we can build a 8-LUT from two 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 5.10 gives an example of multiplication of 8-bit number with 12-bit constant ($M = 8$, $K = 12$). Figure 5.10(a) gives an implementation using 8-LUTs. 20 8-LUTs or equivalently 128 4-LUTs are used in the design.

Figure 5.10(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.



```
      110110010001   (Operand 1)
x         AAAAAAAA   (Operand 2)
SSSSSSSSSSSSSSSSSSSS (Product  )
```

(a)

```
      110110010001   (aaaa * 0001)
x         AAAAaaaa   (Operand 2)
    XXXXXXXXXXXXXXXX (aaaa * 0001)
+YYYYYYYYYYYYYYYY    (AAAA * 1001)
SSSSSSSSSSSSSSSSSSSS (Product  )
```

(b)

Fig. 5.10 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 two 4-bit values

5.4.2 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 5.3 shows the performance comparison of SWT architecture with existing work and amongst different configurations. A direct implementation of a Daubechies 9/7 DWT filter requires 16 multipliers and 12 adders in the design. The critical path is $T_m + 4T_a$, where T_m is the time latency of the multiplier and T_a is the time latency of the adder. Several optimizations were proposed including those by [15, 16, 35]. Our earlier work [28] 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 ($4T_a + T_l$). (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 frequencies except the VLSI implementation represent implementation on Vitex-4 FPGA. These FPGAs are built on a 90 nm process technology. To test the speed of VLSI implementation of proposed architecture, we used freepdk 45 nm cell library [32]. 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.

5.5 Parameterized Lifting

DWT can also be implemented using lifting scheme. This scheme was designed by Wim Sweldens and is used for designing and performing the wavelet transform. The

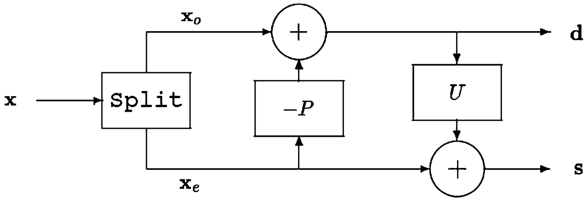


Fig. 5.11 Illustration of lifting-based implementation of wavelet transform

filter bank scheme, as discussed earlier performs a convolution of low- and high-pass filter with the signal. On the other hand, lifting scheme consists of alternating lifts. Once the low pass is fixed, the high pass is changed and vice versa.

The basic idea of lifting is as follows: Consider a signal $x = (x_k)_{k \in Z}$. Let us split it into two polyphase components: the even indexed samples x_e and the odd indexed samples x_o:

$$x_e = (x_{2k})_{k \in Z}, \qquad x_o = (x_{2k+1})_{k \in Z}$$

These two sets are correlated as they sample the same real-life event. Therefore, one set can be used as a predictor of the other, so that we need not record the details of other but only the difference between actual value and predictor (d),

$$d = x_o - P(x_e)$$

A simple predictor for odd samples, for example would take average of its two neighbors. In that case, the above equation will look like this:

$$d_k = x_{2k+1} - \frac{x_{2k} + x_{2k+2}}{2}$$

A good predictor will lead to perfect reconstruction of values (x_{2k+1}) from d_k values. However, d will be sparse, have lower entropy and hence lead to compression. This step is called predict step. Another step, namely the update step is used to remove the problems of aliasing and restores the correct running average. This is used to smoothen the values to correct dc average:

$$s = x_e + U(d)$$

A simple example of this is as follows:

$$s_k = x_{2k} + \frac{d_{k-1} + d_k}{4}$$

These steps can be repeated multiple times. Figure 5.11 gives a brief illustration of the lifting scheme. A more elaborate description of lifting procedure is given in [8]. Lifting is also referred to as in-place computation and reduces the memory complexity of implementation. Successive steps of same direction can be merged. The decimation at the beginning results in a speed up of 2 and all the transforms implemented using lifting scheme are a perfect reconstruction.

The lifting-based implementation of a 9/7 filter is presented by Guangjun et al. [13]. The lifting decomposition for CDF 9/7 leads to following expression:

$$s_k^{(0)} = x_{2k} \tag{5.1}$$

$$d_k^{(0)} = x_{2k+1} \tag{5.2}$$

$$d_k^{(1)} = d_k^{(0)} + \alpha\big(s_k^{(0)} + s_{k+1}^{(0)}\big) \tag{5.3}$$

$$s_k^{(1)} = s_k^{(0)} + \beta\big(d_k^{(1)} + d_{k-1}^{(1)}\big) \tag{5.4}$$

$$d_k^{(2)} = d_k^{(1)} + \gamma\big(s_k^{(1)} + s_{k+1}^{(1)}\big) \tag{5.5}$$

$$s_k^{(2)} = s_k^{(1)} + \theta\big(d_k^{(2)} + d_{k-1}^{(2)}\big) \tag{5.6}$$

$$s_k = \delta s_k^{(2)} \tag{5.7}$$

$$d_k = d_k^{(2)}/\delta \tag{5.8}$$

The normalization condition (namely, $H_0(\omega) = \sqrt{2},\ H_1(\omega) = \sqrt{2}$), at $\omega = 0$ leads to

$$h_0 + 2\sum_{n=1}^{4} h_0(n) = \sqrt{2} \tag{5.9}$$

$$h_1 + 2\sum_{n=1}^{4} h_1(n) = \sqrt{2} \tag{5.10}$$

$$\Rightarrow h_0 + 2\sum_{n=1}^{4} (-1)^n h_0(n) = 0 \tag{5.11}$$

$$h_1 + 2\sum_{n=1}^{4} (-1)^n h_1(n) = 0 \tag{5.12}$$

$$2\sum_{n=1}^{3} (-1)^n h_1(n) = 0 \tag{5.13}$$

They present the above simplification after assuming $N_1 = 2$, $N_2 = 4$, and calculating derivatives for expressions of $H_1(\omega)$ and $H_2(\omega)$. The solution gives the values of all other variables in terms of α:

$$\beta = \frac{-1}{4(1 + 2\alpha^2)} \tag{5.14}$$

$$\gamma = \frac{-1 - 14\alpha - 4\alpha^2}{1 + 4\alpha} \tag{5.15}$$

$$\theta = \frac{1}{16}\left(4 - \frac{2+4\alpha}{(1+2\alpha)^4} + \frac{1-8\alpha}{(1+2\alpha)^2}\right) \tag{5.16}$$

$$\delta = \frac{2\sqrt{2}(1+2\alpha)}{1+4\alpha} \tag{5.17}$$

Like the earlier work, the free parameter α can be used to parameterize the lifting-based wavelet transform. Engel and Uhl [9] present such a parameterization. The range of α which gives good compression performance is $\{-\inf, -1.4] \cup [0.2, \inf\}$.

5.6 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 work.
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 work 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.

References

1. Ahmed, E., Rose, J.: The effect of LUT and cluster size on deep-submicron FPGA performance and density. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **12**(3), 288–298 (2004)
2. Baugher, M., McGrew, D., Naslund, M., Carrara, E., Norrman, K.: The secure real-time transport protocol (SRTP) (2004)
3. Brachtl, M., Uhl, A., Dietl, W.: Key-dependency for a wavelet-based blind watermarking algorithm. In: Proc. ACM Workshop on Multimedia and Security (MM&Sec), pp. 175–179. ACM Press, New York (2004). doi:10.1145/1022431.1022462
4. Canvel, B., Hiltgen, A., Vaudenay, S., Vuagnoux, M.: Password interception in a SSL/TLS channel. In: The 23rd Annual International Cryptology Conference, CRYPTO'03, vol. 2729, pp. 583–599 (2003)

5. Cheng, C.-C., Tseng, P.-C., Chen, L.-G.: Multimode embedded compression Codec engine for power-aware video coding system. IEEE Trans. Circuits Syst. Video Technol. **19**(2), 141–150 (2009). doi:10.1109/TCSVT.2008.2009250
6. Cheng, H., Li, X.: Partial encryption of compressed images and videos. IEEE Trans. Signal Process. **48**(8), 2439–2451 (2000). doi:10.1109/78.852023
7. Christopoulos, C., Skodras, A., Ebrahimi, T.: The JPEG2000 still image coding system: an overview. IEEE Trans. Consum. Electron. **46**(4), 1103–1127 (2000)
8. Daubechies, I., Sweldens, W.: Factoring wavelet transforms into lifting steps. Technical report (1998)
9. Engel, D., Uhl, A.: Parameterized biorthogonal wavelet lifting for lightweight JPEG 2000 transparent encryption. In: Proc. ACM Workshop on Multimedia and Security (MM&Sec), pp. 63–70. ACM Press, New York (2005). doi:10.1145/1073170.1073183
10. FIPS 197: Announcing the Advanced Encryption Standard (2001)
11. FIPS 46-2: Announcing the standard for Data Encryption Standard (1993)
12. Grangetto, M., Magli, E., Olmo, G.: Multimedia selective encryption by means of randomized arithmetic coding. IEEE Trans. Multimed. **8**(5), 905–917 (2006)
13. Guangjun, Z., Lizhi, C., Huowang, C.: A simple 9/7-tap wavelet filter based on lifting scheme. In: Proceedings of 2001 International Conference on Image Processing, vol. 2, pp. 249–252. IEEE Press, New York (2001)
14. Hodjat, A., Verbauwhede, I.: A 21.54 Gbit/s fully pipelined AES processor on FPGA. In: Proc. IEEE Symp. Field-Programmable Custom Computing Machines (FCCM), pp. 308–309 (2004). doi:10.1109/FCCM.2004.1
15. Huang, C., Tseng, P., Chen, L.: Flipping structure: an efficient VLSI architecture for lifting-based discrete wavelet transform. IEEE Trans. Signal Process. **52**(4), 1080–1089 (2004)
16. Jou, J.M., Shiau, Y.-H., Liu, C.-C.: Efficient VLSI architectures for the biorthogonal wavelet transform by filter bank and lifting scheme. In: IEEE Intl. Symp. Circuits and Systems (ISCAS 2001), vol. 2, pp. 529–532 (2001). doi:10.1109/ISCAS.2001.921124
17. Kim, H., Wen, J., Villasenor, J.D.: Secure arithmetic coding. IEEE Trans. Signal Process. **55**(5), 2263–2272 (2007). doi:10.1109/TSP.2007.892710
18. Lai, X., Massey, J.L.: A proposal for a new Block Encryption Standard. In: EUROCRYPT'90, pp. 389–404. Springer, New York (1991)
19. Lian, S., Liu, Z., Ren, Z., Wang, H.: Commutative encryption and watermarking in video compression. IEEE Trans. Circuits Syst. Video Technol. **17**(6), 774–778 (2007)
20. Lian, S., Wang, Z.: Comparison of several wavelet coefficient confusion methods applied in multimedia encryption. In: Intl. Conf. Computer Networks and Mobile Computing, pp. 372–376 (2003)
21. Liu, X., Eskicioglu, A.M.: Selective encryption of multimedia content in distribution networks: challenges and new directions. In: Communications, Internet, and Information Technology (CIIT 2003), pp. 276–285 (2003)
22. Liu, Z., Zheng, N.: Parametrization construction of biorthogonal wavelet filter banks for image coding. Signal Image Video Process. **1**(1), 63–76 (2007)
23. Mano, M.M., Ciletti, M.D.: Digital Design, 4th edn. Prentice-Hall, Upper Saddle River (2006)
24. Mao, Y., Wu, M.: A joint signal processing and cryptographic approach to multimedia encryption. IEEE Trans. Image Process. **15**(7), 2061–2075 (2006)
25. Marcellin, M.W., Bilgin, A.: Quantifying the parent-child coding gain in zero-tree-based coders. IEEE Signal Process. Lett. **8**(3), 67–69 (2001). doi:10.1109/97.905942
26. Martin, K., Plataniotis, K.N.: Privacy protected surveillance using secure visual object coding. IEEE Trans. Circuits Syst. Video Technol. **18**(8), 1152–1162 (2008)
27. Martina, M., Masera, G.: Multiplierless, folded 9/7–5/3 wavelet VLSI architecture. IEEE Trans. Circuits Syst. II **54**(9), 770–774 (2007)
28. Pande, A., Zambreno, J.: Poly-dwt: polymorphic wavelet hardware support for dynamic image compression. ACM Trans. Embed. Comput. Syst. **11**(1), 6 (2010)
29. Pande, A., Zambreno, J.: A reconfigurable architecture for secure multimedia delivery. In: 23rd Intl. Conf. VLSI Design, pp. 258–263 (2010). doi:10.1109/VLSI.Design.2010.50

30. Pande, A., Zambreno, J.: An efficient hardware architecture for multimedia encryption and authentication using discrete wavelet transform. In: IEEE CS Intl. Symp. VLSI, pp. 85–90 (2009)
31. Shannon, C.E.: Communication theory of secrecy systems. Bell Syst. Tech. J. **28**, 656–715 (1949)
32. Stine, J., Castellanos, I., Wood, M., Henson, J., Love, F., Davis, W.R., Franzon, P.D., Bucher, M., Basavarajaiah, S., Oh, J., Jenkal, R.: FreePDK: an open-source variation-aware design kit. In: IEEE Int. Conf. on Microelectronic Systems Education, MSE 2007, pp. 173–174 (2007). doi:10.1109/MSE.2007.3
33. Strang, G., Nguyen, T.: Wavelets and Filter Bank. Wellesley/Cambridge University Press, Cambridge (1996)
34. Tseng, P., Chang, Y., Huang, Y., Fang, H., Huang, C., Chen, L.: Advances in hardware architectures for image and video coding—a survey. Proc. IEEE **93**(1), 184–197 (2005). doi:10.1109/JPROC.2004.839622
35. Vishwanath, M., Owens, R.M., Irwin, M.J.: VLSI architectures for the Discrete Wavelet Transform. IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process. **42**(5), 305–316 (1995). doi:10.1109/82.386170
36. Wagner, D., Schneier, B.: Analysis of the SSL 3.0 protocol. In: Proc. Second UNIX Workshop on Electronic Commerce, pp. 29–40. USENIX Association, Berkeley (1996)
37. Wang, S.-J., Chen, H.-H., Chen, P.-Y., Tsai, Y.-R.: Security cryptanalysis in high-order improved fast encryption algorithm for multimedia. In: Future Generation Communication and Networking (FGCN 2007), vol. 1, pp. 328–331 (2007). doi:10.1109/FGCN.2007.199
38. Yi, X., Tan, C.H., Slew, C.K., Rahman Syed, M.: Fast encryption for multimedia. IEEE Trans. Consum. Electron. **47**(1), 101–107 (2001). doi:10.1109/30.920426

Chapter 6
Chaotic Filter Banks

Abstract 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 work, 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.

6.1 Introduction

6.1.1 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 [26]. 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 [6, 15, 24]. However, these schemes need a synchronization procedure. On the other hand, discrete-time chaotic systems behave like private-key encryption algorithms [25] and are amenable to implementation on fixed-point hardware.

Many chaotic block ciphers [2, 9, 12, 17, 22] have been proposed in research literature. For example, Baptista [2] builds a block cipher based on chaotic encryption. Each character of the message is encoded as the integer number of iterations

A. Pande, J. Zambreno, *Embedded Multimedia Security Systems*,
DOI 10.1007/978-1-4471-4459-5_6, © Springer-Verlag London 2013

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 can be explained as follows.

Firstly, the distribution of the cipher text 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 cipher text is at least twice that of plain text, 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 Habutsu et al. [10]. Its cryptanalysis was presented in the same conference [5]. Chen et al. [9, 17] constructed a block cipher based on three-dimensional maps while Pichler and Scharinger [22] proposed a cipher by direct discretization of two-dimensional Baker map. A good survey and introductory tutorial on these schemes is found in [11, 29]. Masuda and Aihara [19] present a crypto-system based on a discretization of the skew tent map. Masuda et al. [20] present 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.

6.1.2 Wavelets and Chaotic Filter Banks

Chaotic Filter Banks-based cipher was first proposed in 2007 by Ling et al. [16]. 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 selected 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 easily be changed by the designers and perfect reconstruction is still guaranteed no matter how small these ratios are.
4. The proposed cryptographic system can easily be adapted to the international multimedia standards, such as JPEG 2000 and MPEG4 [16].

The encryption procedure is carried out by decomposing the input plain text signal into two different subbands and masking each of them with a pseudo-random number sequence generated by iterating the chaotic logistic map. Ling et al. [16] use the Discrete Wavelet Transform (DWT)-based filter banks in their approach to maintain compatibility with existing image compression standards such as JPEG2000 [7].

Arroyo et al. [1] present a cryptanalysis of [16] 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.

6.1.3 Scope and Organization

In this work 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 work 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 the authors’ knowledge, 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 work is organized as follows. Section 6.2 gives a brief overview of the wavelet transform. Section 6.3 gives details of the chaotic filter bank scheme proposed earlier. In Sect. 6.4, we discuss the MCFB Scheme and subsequently discuss its distinguishing features in Sects. 6.5 and 6.6. Section 6.5 explains the Improved Chaotic Oscillator and Sect. 6.6 gives an overview of wavelet parameterization. Section 6.7 gives the details of hardware implementation over Xilinx Virtex-5 FPGA and the proposed optimizations, while Sect. 6.8 concludes the work with directions of future work.

6.1.4 Reconfigurable Hardware Implementation of DWT

Much research has been done in the development of DWT architectures for image processing [3, 4, 14, 18, 23]. A good survey on architectures on DWT coding is given by Tseng et al. [27].

Recent work in partial reconfiguration of FPGAs implements DWT in a reconfigurable fashion. Claus et al. [8] give a comparison of embedded reconfigurable video-processing architectures. They propose a hybrid of two hardware platforms:

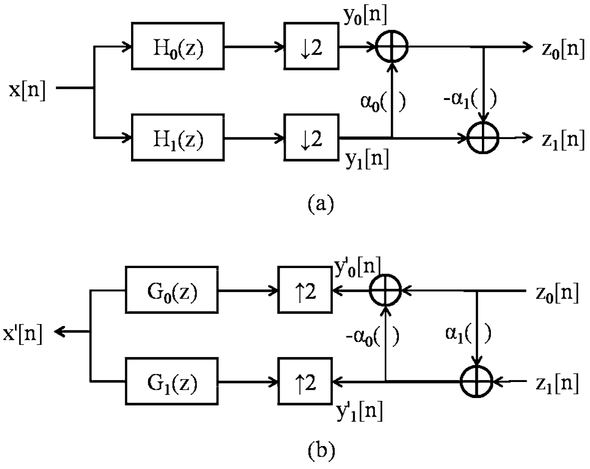


Fig. 6.1 Block Diagram representation of the Chaotic Filter Bank Scheme. (**a**) The encryption module and (**b**) the decryption module

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 [13] simplifies the generation of dynamically reconfigurable systems to almost a push button process. The work also describes a communication infrastructure for dynamically reconfigurable systems.

6.2 Chaotic Filter Bank Scheme

The chaotic filter bank scheme is illustrated in Fig. 6.1. A chaotic function $\alpha_i(\ )$ is used to create a chaotic response to the system:

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

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

The various signals in Fig. 6.1 are expressed as follows:

$$\begin{aligned}
y_0[n] &= \sum_{\forall m} x[m]h_0[2n-m] \\
y_1[n] &= \sum_{\forall m} x[m]h_1[2n-m] \\
z_0[n] &= y_0[n] + \alpha_0\big(y_1[n]\big) \quad \text{and} \\
z_1[n] &= y_1[n] - \alpha_1\big(y_0[n]\big) \\
\Rightarrow z_0[n] &= y_0[n] + y_1[n] + s_0[n] \quad \text{and} \\
z_1[n] &= y_1[n] + y_0[n] - s_1[n]
\end{aligned}$$

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.

$$
\begin{aligned}
y_1^{'}[n] &= z_1[n] + \alpha_1\big(z_0[n]\big),\\
y_0^{'}[n] &= z_0[n] - \alpha_0\big(z_1[n]\big),\\
x^{'}[n] &= \sum_{\forall m} y_0^{'}[m]g_0[n-2m] + \sum_{\forall m} y_1^{'}[m]g_1[n-2m]
\end{aligned}
$$

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 for correct recovery of the plaintext signal.

6.2.1 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

$$x_{n+1} = \lambda_{\mathrm{LM}} \cdot x_n(1 - x_n)$$

where λ_{LM} is a positive number.

The behavior of logistic map is dependent on the value of λ_{LM}. At $\lambda_{\mathrm{LM}} \approx 3.57$ 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 chaotic behavior, but certain isolated values of λ_{LM} appear to show non-chaotic behavior and are called islands of stability. Beyond $\lambda_{\mathrm{LM}} = 4$, 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 λ 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.

6.2.2 Key Space

Ling et al. [16] suggest using the initial values of logistic map and the value of parameter λ_{LM} to build the key space.

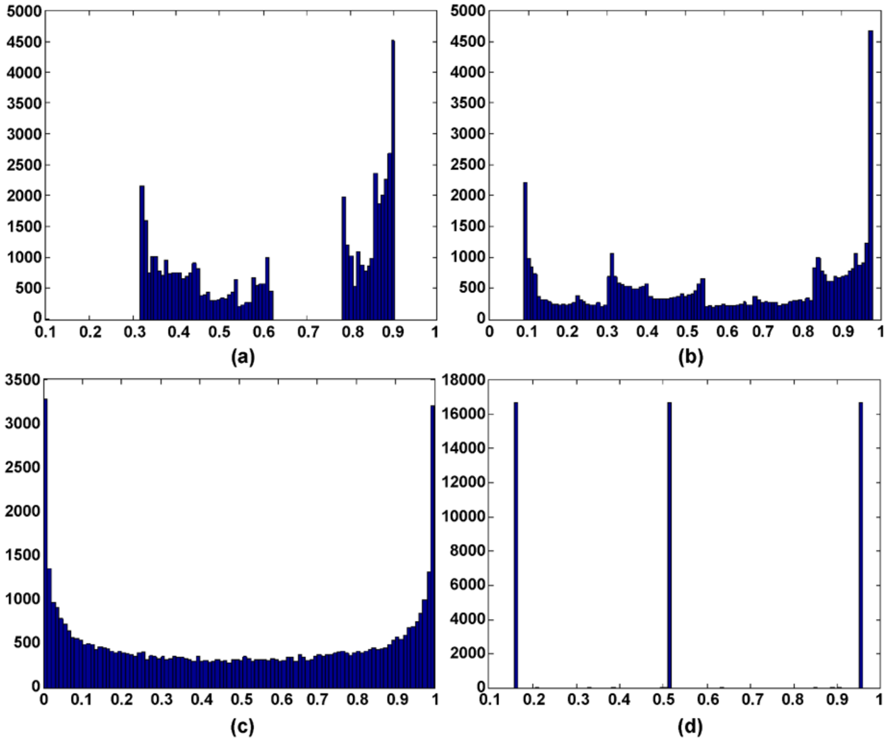


Fig. 6.2 Histogram for 50,000 samples obtained using logistic map with initial seed 0.100010 and (**a**) $\lambda_{\text{LM}} = 3.61$, (**b**) $\lambda_{\text{LM}} = 3.91$, (**c**) $\lambda_{\text{LM}} = 4$, and (**d**) $\lambda_{\text{LM}} = 3.83$

Arroyo et al. [1] 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**. Ling et al. [16] propose to use the entire range $[3,4]$ as the key space. The values of λ_{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 Fig. 6.2(d)). Another issue is the non-uniform distribution of output values (as shown in Figs. 6.2(a)–(b)).
2. **Vulnerablity to known plain-text attack**. The value of λ_{LM} can be calculated very accurately from two successive iterations of the logistic map leading to successful plain-text attacks on the scheme.

6.3 The MCFB Scheme

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

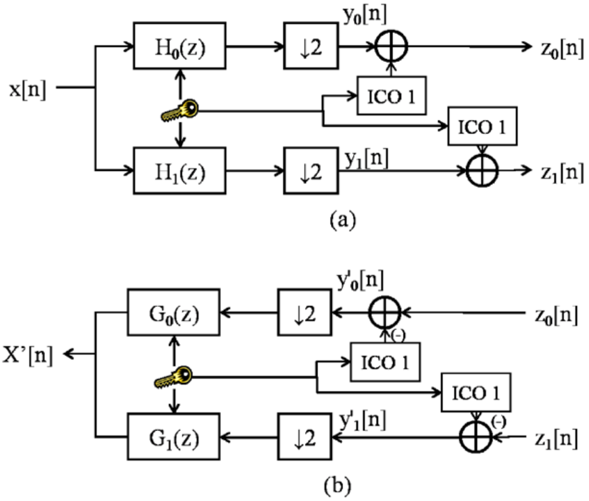


Fig. 6.3 Block Diagram representation of the MCFB scheme. (**a**) The encryption module and (**b**) the decryption module

1. The Chaotic Filter Bank scheme [16] 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 the Discrete Wavelet Transform.

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

$$z_0[n] = y_0[n] + s_0[n] \quad \text{and}$$
$$z_1[n] = y_1[n] + s_1[n]$$

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 Fig. 6.3.

The improved chaotic oscillator and parameterized wavelet transform are explained in following two sections.

6.4 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 [16]. The proposed scheme is robust to the choice of initial conditions (due to lack of any unsuitable λ values), achieves real-time encryption speed and is resistant to known attacks.

6.4.1 The Modified Logistic Map (MLM)

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

1. The histogram obtained for different λ_{LM} values (with 50000 samples) is skewed and not uniform or flat. This is illustrated for $\lambda_{\mathrm{LM}} = 3.61$ and $\lambda_{\mathrm{LM}} = 3.91$ values in Figs. 6.2(a)–(b). The distribution for $\lambda_{\mathrm{LM}} = 4$ is most flat and symmetric (see Fig. 6.2(c)). It is desirable to have a flatter distribution of samples drawn from the logistic map in order to increase its randomness.
2. For $\lambda_{\mathrm{LM}} = 4$, the logistic map equation $x_{n+1} = \lambda_{\mathrm{LM}} \cdot x_n(1 - x_n)$ has the same domain and range intervals $(0, 1)$. For $\lambda_{\mathrm{LM}} < 4$ and input x_n in range $(0, 1)$, the range of x_{n+1} in the expression is $(0, \lambda_{\mathrm{LM}}/4]$ and the distribution of random numbers is biased towards 0 or 1 (as seen in distributions in Figs. 6.2(a)–(b)). It is desirable to have a distribution of random numbers symmetric around 0.5.
3. There are certain isolated values of λ_{LM} that appear to show non-chaotic behavior and are called islands of stability. For example: $\lambda_{\mathrm{LM}} = 1 + \sqrt{8} \approx 3.83$ shows oscillation between three values.
4. $\lambda_{\mathrm{LM}} = 4.0$ has most flat, uniform and symmetric histogram than other λ_{LM} values.

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

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

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 $\lambda \cdot \alpha(1 - \alpha) + \mu$. Equating the maximum and minimum values to the range $[\alpha, (1 - \alpha)]$ leads to the following equations:

$$\alpha = \lambda\alpha(1 - \alpha) + \mu$$

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

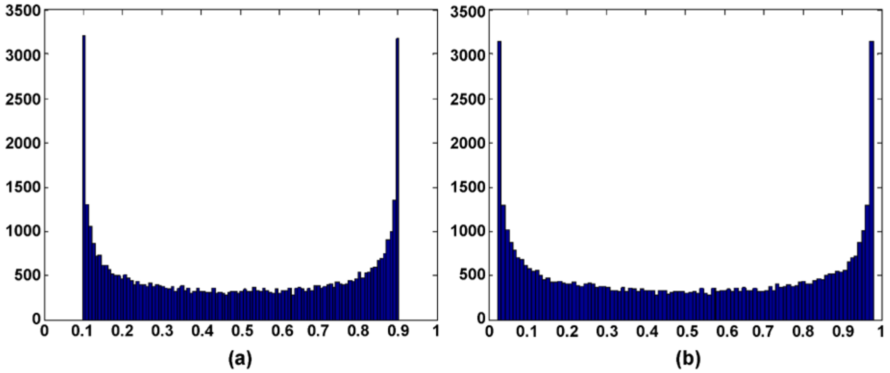


Fig. 6.4 Histogram for 50,000 samples obtained using Modified Logistic map with α values corresponding to (**a**) $\lambda_{\mathrm{LM}} = 3.61$ and (**b**) $\lambda_{\mathrm{LM}} = 3.91$

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 Fig. 6.4. 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 α.

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:

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

where a_j are individual bit values.

Thus, p_n is given by

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

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 decorrelation makes it difficult to back-track p_n from s_n.

6.5 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 the previous chapter). We get the following expression for $H_1(z)$ and $H_2(z)$:

$$\begin{aligned} H_1(z) &= \big(-9/64a + 1/32a^2 + 15/64 - 1/8/a\big)\big(z^4 + 1/z^4\big) \\ &\quad + \big(-1/16a^2 + 11/32a - 11/16 + 1/2/a\big)\big(z^3 + 1/z^3\big) \\ &\quad + (1/8 - 1/2/a)\big(z^2 + 1/z^2\big) \\ &\quad + \big(-11/32a + 1/16a^2 + 15/16 - 1/2/a\big)(z + 1/z) \\ &\quad + \big(9/32a - 1/16a^2 - 7/32 + 5/4/a\big) \\ H_2(z) &= (1/32 - 1/32a)\big(z^3 + 1/z^3\big) \\ &\quad + (1/8 - 1/16a)\big(z^2 + 1/z^2\big) \\ &\quad + (7/32 + 1/32a)(z + 1/z) + (1/4 + 1/8a) \end{aligned}$$

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.

6.6 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 [21].

Shannon [26] explains that a good crypto-system must show diffusion and confusion properties. Confusion refers to making the relationship between the key and the cipher text 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 (α) that make the key. We perform some statistical tests to test the pseudo-random nature of the key obtained.

6.6.1 Randomness Tests

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

6.6.1.1 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 10,000 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 10,000.

6.6.1.2 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 10,000 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).

6.6.1.3 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

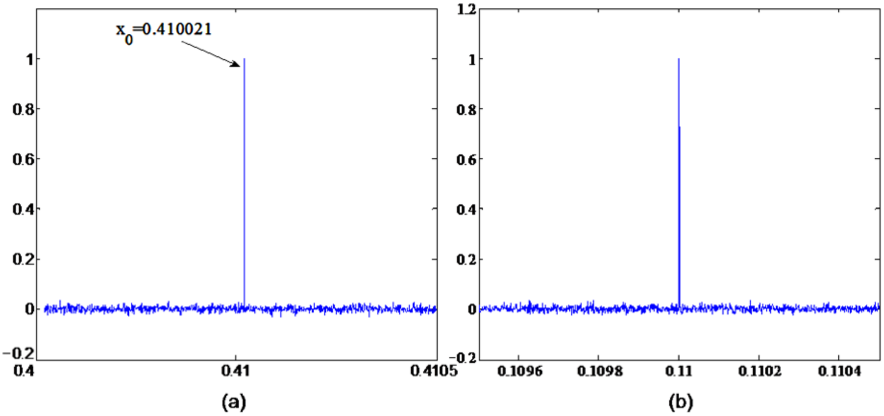


Fig. 6.5 Correlation test of the pseudo-random sequence. (**a**) Generated using different initial values x_0 and (**b**) different initial parameter α. The plots are measured against initial value $\alpha = 0.110000$ and $x_0 = 0.410021$

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 10,000 simulations with different initial values and parameter settings, giving 8916 successful simulations with $H = 0$.

6.6.1.4 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 [25]. A long period is ensured by taking a large value of N (say 64). Figures 6.5(a) and (b) show the low correlation between sequences obtained using slightly different (a) initial value x_0 and (b) parameter λ. It can be seen that a very poor correlation is obtained amongst sequences generated using slightly different initial conditions or parameters. The probability of zero in generated sequences is close to 0.5 (see Table 6.1).

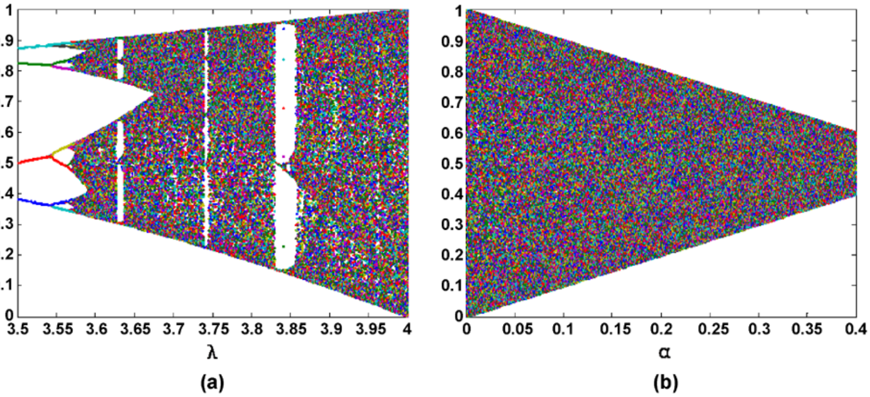


Fig. 6.6 Bifurcation diagram for (**a**) logistic map showing the white spaces (islands of stability) and asymmetricity and (**b**) modified logistic map with symmetric and flatter distribution

Table 6.1 Statistical performance of Generated Sequence b_n (results based on 1000 sequences of length 10,000 each)

Probabilities of zero	0.4993
Probabilities of one	0.5007

6.6.2 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 λ_{LM}.

The bifurcation map of logistic map is shown in Fig. 6.6(a). It is observed that for some value of λ_{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 the logistic map.

Figure 6.6(b) shows the bifurcation map of MLM as a function of free parameter α. 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 α can be used to build the key space.

6.6.3 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:

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

where $\delta Z(t)$ is the separation at time t and δZ_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 discrete-time equivalent expression to find Lyapunov exponent of MLM will be

$$\begin{aligned}\Lambda &= \lim_{n\to\infty} \frac{1}{n} \ln \frac{|\delta x_n|}{|\delta x_0|}\\ &= \lim_{n\to\infty} \frac{1}{n} \ln \frac{|\delta x_n|}{|\delta x_{n-1}|} \frac{|\delta x_{n-1}|}{|\delta x_{n-2}|} \cdots \frac{|\delta x_1|}{|\delta x_0|}\end{aligned}$$

An analysis similar to logistic map [28] can be performed to prove the positive Lyapunov exponent for logistic maps:

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

Hence,

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

Therefore, we can express Λ as follows:

$$\begin{aligned}\Lambda &= \lim_{n\to\infty} \frac{1}{n} \left(\sum_{j=1}^{j=n} \ln \left| \frac{\delta x_j}{\delta x_{j-1}} \right| \right)\\ &= \lim_{n\to\infty} \frac{1}{n} \left(\sum_{j=1}^{j=n} \ln \left| \lambda(1 - 2x_j) \right| \right)\end{aligned}$$

The value of Λ 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 Figs. 6.7(a) and (b). This value was found to be $\ln 2$ for MLM which is the same as the value for logistic map with $\lambda_{\mathrm{LM}} = 4$. 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 α indicating the invariance of confusion properties with the change in parameter α.

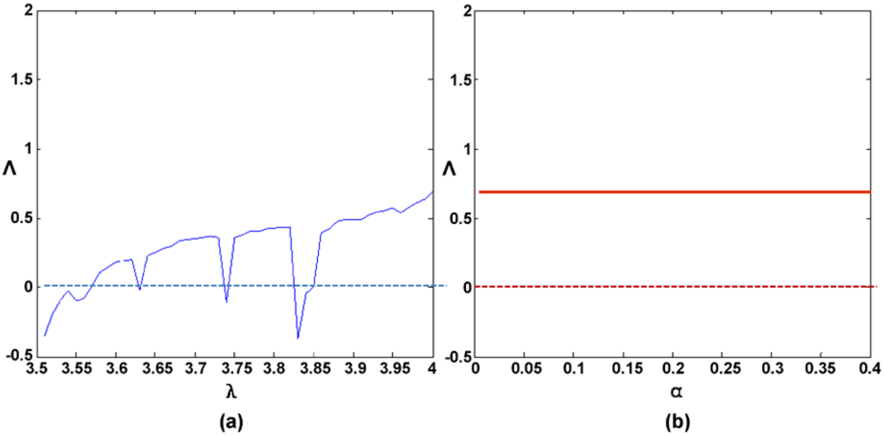


Fig. 6.7 Plot of Lyapunov coefficient (Λ—*solid line*) for (**a**) logistic map as a function of parameter λ_{LM} indicating regions of non-chaotic behavior and (**b**) modified logistic map showing higher divergence than logistic map and independence of Λ from parameter α

6.7 Security Enhancement

A serious drawback of chaotic crypto-systems is that they are weak against known-plain-text 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 plain text. Our proposed scheme has many advantages over the logistic map.

- The Modified Logistic Map has better security properties than the logistic map. Figure 6.5 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 maps for LM and MLM are shown in Fig. 6.6. The absence of any white space in the keyspace of MLM allows us to build a continuous key space. Figure 6.7 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 plain text.
- 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

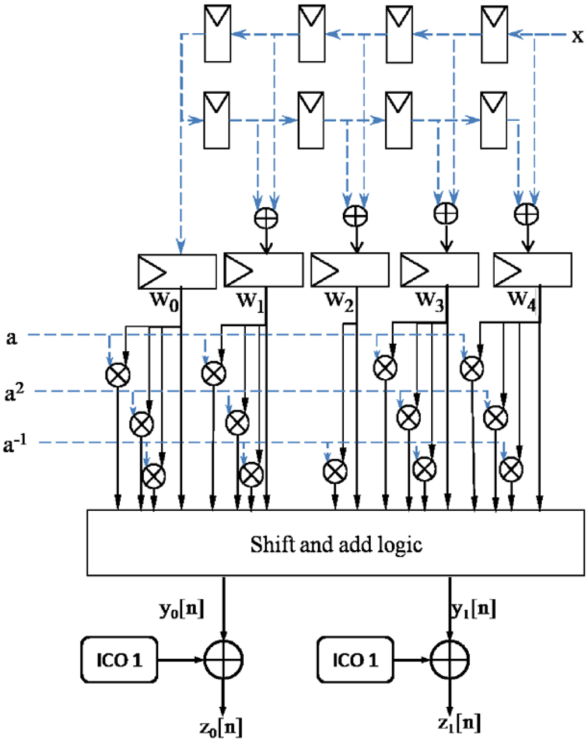


Fig. 6.8 Hardware architecture for the Modified Chaotic Filter Bank scheme

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 helps 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.

6.8 Hardware Implementation

Figure 6.8 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 output of two Improved Chaotic Oscillators is then added to these two signals to get $z_0[n]$ and $z_1[n]$, respectively.

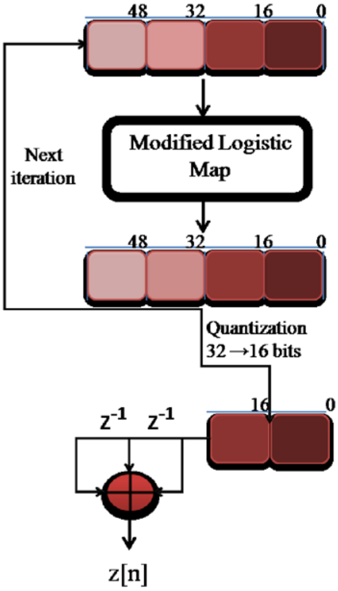


Fig. 6.9 Hardware architecture for Improved Chaotic Oscillator

The hardware architecture of ICOs is shown in Fig. 6.9. Two instances of ICOs are required in the design.

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 Fig. 6.8, 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 eight 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 is added to the output $y_i[n]$ 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 λ and μ are both implemented with 64 bits fixed-point precision. The permissible range of parameter α 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 λ is then calculated to be $(4, 16)$ which is implemented with 5.59 I.F. format. The range for μ is $(-3, -15.0975)$ which is represented using 5.59 I.F. format. Thus, the multiplication $\lambda \cdot x(i) \cdot (1 - x(i))$ is truncated to 5.59 I.F. format and then added to μ 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.

6.8.1 Hardware Optimizations for ICO

A single DSP48E slice can perform a maximum of 25 × 18 bits multiplication and hence 12 slices are required for a 64 × 64 bits multiplication. Two multiplications 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 $X_a X_b X_c$ where X_a, X_b and X_c are each 24 bit long sequences. We have

$$\{X_{\text{SE}}\}_0^{71} = \{X_a\}_{48}^{71}\{X_b\}_{24}^{47}\{X_c\}_0^{23}$$

Similarly, we can represent Y as combination of four 16 bit numbers $Y_w Y_x Y_y Y_z$,

$$\{Y\}_0^{71} = \{Y_w\}_{48}^{63}\{Y_x\}_{32}^{47}\{Y_y\}_{17}^{31}\{Y_z\}_0^{15}$$

Numerically,

$$X = X_{\text{SE}} = X_a \cdot 2^{48} + X_b \cdot 2^{24} + X_c$$

and

$$Y = Y_w \cdot 2^{48} + Y_x \cdot 2^{32} + Y_y \cdot 2^{16} + Y_z$$

.

The product $X \cdot Y$ can then be represented as

$$\begin{aligned} X \cdot Y &= \left(X_a \cdot 2^{48} + X_b \cdot 2^{24} + X_c\right) \cdot \left(Y_w \cdot 2^{48} + Y_x \cdot 2^{32} + Y_y \cdot 2^{16} + Y_z\right) \\ \Rightarrow X \cdot Y &= 2^{96} \cdot X_a Y_w + 2^{72} \cdot X_b Y_w + 2^{48} \cdot X_c Y_w \\ &\quad + 2^{80} \cdot X_a Y_x + 2^{56} \cdot X_b Y_x + 2^{32} \cdot X_c Y_x \\ &\quad + 2^{64} \cdot X_a Y_y + 2^{40} \cdot X_b Y_y + 2^{16} \cdot X_c Y_y \\ &\quad + 2^{48} \cdot X_a Y_z + 2^{24} \cdot X_b Y_z + 2^{0} \cdot X_c Y_z \end{aligned}$$

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 the lower 72 bits of the product. Each of the products $X_\alpha Y_\beta$, 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,

$$\begin{aligned} X \cdot Y = 2^{96} \cdot X_a Y_w &+ 2^{72} \cdot X_b Y_w + 2^{48} \cdot X_c Y_w \\ + 2^{80} \cdot X_a Y_x &+ 2^{56} \cdot X_b Y_x \\ + 2^{64} \cdot X_a Y_y &+ 2^{40} \cdot X_b Y_y \\ + 2^{48} \cdot X_a Y_z & \end{aligned}$$

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 10 × 9 bits multipliers and 4 64 × 64 bits multiplier which required 48 DSP48E slices and Look Up Tables for implementation. The optimized implementation uses only 32 24 × 16 bits multiplier which are implemented in 32 DSP48E slices. Moreover, the achievable clock frequency increases by 30 % from 67.8 MHz to 88.3 MHz.

6.9 Conclusions

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

This work also presents a reconfigurable hardware implementation of the proposed scheme. Use of reconfigurable hardware allows partial removal of hardmultipliers 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.

References

1. Arroyo, D., Li, C., Li, S., Alvarez, G.: Cryptanalysis of a computer cryptography scheme based on a filter bank. Chaos Solitons Fractals **41**(1), 410–413 (2009). doi:10.1016/j.chaos.2008.01.020

2. Baptista, M.S.: Cryptography with chaos. Phys. Lett. **240**(1–2), 50–54 (1998)
3. Benkrid, A., Benkrid, K., Crookes, D.: Design and implementation of a generic 2D orthogonal discrete wavelet transform on FPGA. In: Proc. IEEE Symp. Field-Programmable Custom Computing Machines (FCCM), pp. 162–172 (2003)
4. Benkrid, A., Crookes, D., Benkrid, K.: Design and implementation of a generic 2D biorthogonal discrete wavelet transform on an FPGA. In: Proc. IEEE Symp. Field-Programmable Custom Computing Machines (FCCM), pp. 190–198 (2001)
5. Biham, E.: Cryptanalysis of the chaotic-map cryptosystem suggested at EUROCRYPT'91. In: Advances in Cryptology—EUROCRYPT'91, Lecture Notes in Computer Science, pp. 532–534. Springer, Berlin (1991)
6. Carroll, T.L., Pecora, L.M.: Synchronizing chaotic circuits. IEEE Trans. Circuits Syst. **38**(4), 453–456 (1991). doi:10.1109/31.75404
7. Christopoulos, C., Skodras, A., Ebrahimi, T.: The JPEG2000 still image coding system: an overview. IEEE Trans. Consum. Electron. **46**(4), 1103–1127 (2000)
8. Claus, C., Stechele, W., Kovatsch, M., Angermeier, J., Teich, J.: A comparison of embedded reconfigurable video-processing architectures. In: Proc. IEEE Intl. Conf. Field Programmable Logic and Applications, FPL 2008, pp. 587–590 (2008). doi:10.1109/FPL.2008.4630015
9. Guanrong Chen, Y.M., Chui, C.K.: A symmetric image encryption scheme based on 3d chaotic cat maps. Chaos Solitons Fractals **21**(3), 749–761 (2004)
10. Habutsu, T., Nishio, Y., Sasase, I., Mori, S.: A secret key cryptosystem by iterating a chaotic map. In: Advances in Cryptology—EUROCRYPT'91, Lecture Notes in Computer Science, pp. 127–140 (1991)
11. Kocarev, L.: Chaos-based cryptography: a brief overview. IEEE Circuits Syst. Mag. **1**(3), 6–21 (2001). doi:10.1109/7384.963463
12. Kocarev, L., Jakimoski, G., Stojanovski, T., Parlitz, U.: From chaotic maps to encryption schemes. In: IEEE Intl. Symp. Circuits and Systems, vol. 4, pp. 514–517 (1998). doi:10.1109/ISCAS.1998.698968
13. Koch, D., Beckhoff, C., Teich, J.: ReCoBus-builder—a novel tool and technique to build statically and dynamically reconfigurable systems for FPGAs. In: Proc. IEEE Intl. Conf. Field Programmable Logic and Applications, FPL 2008, Heidelberg, Germany (2008)
14. Kotteri, K., Barua, S., Bell, A., Carletta, J.: A comparison of hardware implementations of the biorthogonal 9/7 DWT: convolution versus lifting. IEEE Trans. Circuits Syst. II **52**(5), 256–260 (2005)
15. Liang, X., Zhang, J., Xia, X.: Improving the security of chaotic synchronization with a delta-modulated cryptographic technique. IEEE Trans. Circuits Syst. II **55**(7), 680–684 (2008). doi:10.1109/TCSII.2008.921585
16. Ling, B.W.-K., Ho, C.Y.-F., Tam, P.K.-S.: Chaotic filter bank for computer cryptography. Chaos Solitons Fractals **34**(3), 817–824 (2007). doi:10.1016/j.chaos.2006.03.105
17. Mao, Y.B., Chen, G.R., Lian, S.G.: A symmetric image encryption scheme based on 3D chaotic baker maps. Int. J. Bifurc. Chaos **14**(10), 3613–3624 (2004)
18. Martina, M., Masera, G.: Multiplierless, folded 9/7–5/3 wavelet VLSI architecture. IEEE Trans. Circuits Syst. II **54**(9), 770–774 (2007)
19. Masuda, N., Aihara, K.: Cryptosystems with discretized chaotic maps. IEEE Trans. Circuits Syst. I, Fundam. Theory Appl. **49**(1), 28–40 (2002). doi:10.1109/81.974872
20. Masuda, N., Jakimoski, G., Aihara, K., Kocarev, L.: Chaotic block ciphers: from theory to practical algorithms. IEEE Trans. Circuits Syst. I **53**(6), 1341–1352 (2006). doi:10.1109/TCSI.2006.874182
21. May, R.M.: Simple mathematical models with very complicated dynamics. Nature **261**, 459–467 (1976)
22. Pichler, F., Scharinger, J.: Finite dimensional generalized baker dynamical systems for cryptographic applications. In: EUROCAST'95: Select. Papers Fifth Intl. Work. Computer Aided Systems Theory, pp. 465–476. Springer, London (1996)

23. Ritter, J., Molitor, P.: A pipelined architecture for partitioned DWT based lossy image compression using FPGAs. In: Proc. Intl. Symposium on Field Programmable Gate Arrays (FPGA), pp. 201–206 (2001)
24. Robilliard, C., Huntington, E.H., Webb, J.G.: Enhancing the security of delayed differential chaotic systems with programmable feedback. IEEE Trans. Circuits Syst. II **53**(8), 722–726 (2006). doi:10.1109/TCSII.2006.876405
25. Rueppel, R.A.: Analysis and Design of Stream Ciphers. Springer, Berlin (1986)
26. Shannon, C.E.: Communication theory of secrecy systems. Bell Syst. Tech. J. **28**, 656–715 (1949)
27. Tseng, P., Chang, Y., Huang, Y., Fang, H., Huang, C., Chen, L.: Advances in hardware architectures for image and video coding—a survey. Proc. IEEE **93**(1), 184–197 (2005). doi:10.1109/JPROC.2004.839622
28. Wolf, A.: Quantifying Chaos with Lyapunov Exponents. Princeton University Press, Princeton (1986)
29. Yang, T.: A survey of chaotic secure communication systems. Int. J. Comput. Cogn. **2**(2), 81–130 (2004)

Chapter 7
Chaotic Arithmetic Coding

Abstract 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 work, we present a data (image or video) encryption scheme based on arithmetic coding, which we refer to as Chaotic Arithmetic Coding (CAC). In CAC, a large number of chaotic maps can be used to perform coding, each achieving Shannon-optimal compression performance. The exact choice of map is governed by a key. CAC 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. We next describe Binary CAC (BCAC) with some simple Security Enhancement (SE) modes which can alleviate the security of a scheme against known cryptanalysis against AC-based encryption techniques. These modes, namely Plaintext Modulation (PM), Pair-Wise-Independent Keys (PWIK), and Key and ciphertext Mixing (MIX) modes have insignificant computational overhead, while BCAC decoder has lower hardware requirements than BAC coder itself, making BCAC with SE as excellent choice for deployment in secure embedded multimedia systems. A bit sensitivity analysis for key and plaintext is presented along with experimental tests for compression performance.

7.1 Introduction

The issue of real-time multimedia delivery has gained an increased importance in a world dominated by portable multimedia-capable devices, as well as with the emergence of the cloud computing paradigm. The technical challenges involved in such scenarios include providing a delivery mechanism that is highly scalable, **secure**, easily search-able and index-able, all without losing the important **compression** properties. Providing security in a video communication context is especially challenging, as the security requirements tend to be application- and platform-specific, and the input data is characterized by large storage requirements, real-time processing latencies, and the use of standardized video codecs.

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 compressed using arithmetic coder, frequently used characters are stored

A. Pande, J. Zambreno, *Embedded Multimedia Security Systems*,
DOI 10.1007/978-1-4471-4459-5_7, © Springer-Verlag London 2013

with fewer bits and not-so-frequently occurring characters are stored with more bits, resulting in fewer bits used in total [16]. 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 as well as video standards, including H.264/AVC to provide lossless entropy coding.

Arithmetic coding is extremely efficient for compression efficiency in large datasizes 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 [18] are lost. These approaches encrypt the output of compression system into ciphertext, which is completely random and uncorrelated to the compressed bits. This makes it impossible to retain the desired properties of compressed bitstream into the encrypted bitstream. The scheme presented in this work overcomes these limitation because it does not modify any properties of compressed bitstream.

Recently, Grangetto et al. [9] 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 et al. [13] presented a generalization of this procedure, referred to as Secure Arithmetic Coding (SAC). The SAC coder is constructed over a Key-Splitting Arithmetic Coding (KSAC) [30], 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.

7.1.1 Weakness of SAC Coder

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. The SAC encoder may have to work with multiple sub-intervals thereby significantly increasing the computational cost of encoder. Successful attacks have been demonstrated against SAC scheme [11, 28, 37, 38].

We present a joint video encryption and compression scheme based on piecewise linear chaotic maps, referred to as Chaotic Arithmetic Coding (CAC). The idea of using chaotic maps for encryption was presented in [24]. However, the authors use a skew version for data encryption which is prone to known-plaintext attacks, and increases the computational complexity of coder exponentially.

The contributions of this work are as follows:

1. The work presents a generalized framework for video encryption using chaotic maps called Chaotic Arithmetic Coding (CAC). Chaotic maps are highly sensitive to initial conditions, so that slight wrong guess of initial conditions will

lead to imperfect reconstruction. For introduction on chaos theory, please refer to [14]. We discuss the general case with N alphabets and the specific case of binary alphabets.

2. The known weaknesses of arithmetic coding-based encryption schemes are alleviated by security enhancements (SE) proposed in this work.
3. CAC provides goals of encryption without any computational overhead. In fact, the decoding complexity of CAC is less than a normal Binary Arithmetic Coder (BAC) without encryption (one multiplication and one addition per iteration vs. one multiplication and two additions per iteration for BAC).
4. CAC provides low-cost encryption without compression losses for multimedia data, which is suitable for application in embedded systems scenarios where computational resources and power budget is limited.

7.2 Arithmetic Coding with Piece-wise Linear Chaotic Maps

Let us consider a scenario where we have a string $S = x_1 x_2 \ldots x_N$ consisting of N symbols to be encoded. The probability of occurrence of a symbol $s_i, \ i \in 1, 2, \ldots, n$ 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 (ρ) with the following properties:

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

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

- The map can be decomposed into N piece-wise linear parts ρ_k i.e.

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

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

$$\rho_k : [beg_k, end_k) \to [0, 1]$$

The last two propositions lead to

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

- The map ρ_k is one-one and onto i.e.

$$\forall x \in [beg_k, end_k) \ \exists y \in [0, 1): \quad y = \rho_k(x), \quad \text{and}$$
$$\forall y \in [0, 1) \ \exists x \in [beg_k, end_k): \quad \rho_k(x) = y$$

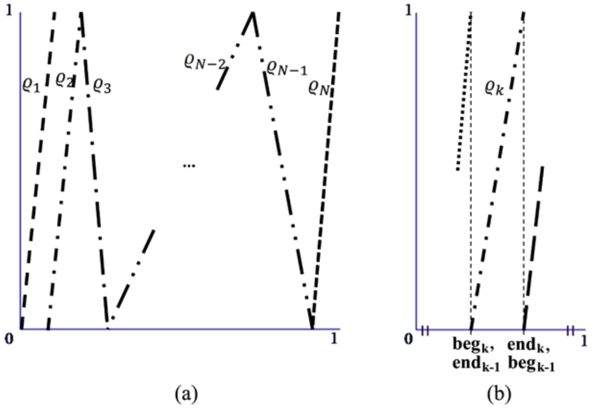


Fig. 7.1 A sample piece-wise linear map for arithmetic coding like compression. (**a**) The entire map is shown (ρ). (**b**) A single linear part of the map (ρ_k) is zoomed. It can have a positive or negative slope depending on choice

- ρ is a many-one mapping from [0, 1) to [0, 1). This implies that the decomposed linear maps (ρ_k) do not intersect each other i.e.

$$\forall(k \neq j): \quad [beg_k, end_k) \cap [beg_j, end_j) = 0$$

- Each linear map ρ_k is associated uniquely with one symbol s_i. The mapping $\rho_k \rightarrow s_i$ is defined arbitrarily but one-one relationship must hold.
- The valid-input width of each map (ρ_k), given by ($end_k - beg_k$) is proportional to a probability of occurrence of symbol s_i:

$$end_k - beg_k \propto p_i$$
$$\Rightarrow \quad end_k - beg_k = C \cdot p_i$$

We recall that $\sum_{k=1}^{N}(end_k - beg_k)$ is same as the input width of $\bigcup_{k=1}^{N} \rho_k = \rho$, which is 1. Also, $\sum_{i=1}^{N} p_i = 1$. Thus, we get the value of constant C to be 1:

$$\Rightarrow end_k - beg_k = p_i$$

Figure 7.1 shows a sample map fulfilling these properties. Figure 7.1(a) shows the full map with different parts $\rho_1, \rho_2, \ldots, \rho_N$ present while Fig. 7.1(b) zooms into individual linear part ρ_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 ρ fulfilling the properties mentioned above is given by $N! = N \cdot (N-1) \cdot (N-2) \cdots 3 \cdot 2 \cdot 1$, 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 $N!2^N$ which is independent of unique symbol probability. 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 eight mappings. If we increase N to 4, this value increases to 384.

The equation for individual maps can be derived as follows:

$$y' = \rho_k(x') = \left(\frac{x' - beg_k}{end_k - beg_k}\right) \quad \text{or} \quad \left(1 - \frac{x' - beg_k}{end_k - beg_k}\right)$$

The equation for the full map is given by

$$y = \rho(x) = \rho_k(x): \quad beg_k \le x < end_k$$

The coding procedure, correspondence to arithmetic coding and compression efficiency of basic chaotic coding is explained in [27].

7.2.1 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 ith 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 $end_j - beg_j$ (width of ρ_j). Thus, the width δ of final interval would be given by

$$\delta = \prod_{j=1}^{N} (end_j - beg_j)^{N_j}$$

We have the relation $end_j - beg_j = p_j = \frac{N_j}{N}$, 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 δ of the same size is $\lceil -\log_2(\delta) \rceil$.

$$\begin{aligned} B &= \left\lceil -\log_2(\delta) \right\rceil \\ &= \left\lceil -\log_2\left(\prod_{j=1}^{N} \left(\frac{N_j}{N}\right)^{N_j}\right) \right\rceil \end{aligned}$$

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

$$= \left\lceil -\sum_{j=1}^{N} N_j \log_2\left(\frac{N_j}{N}\right)\right\rceil \tag{7.1}$$

The average number of bits required per symbol (B_{av}) is given by

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

This relation is derived by averaging the total number of bits which is given by logarithm of δ. According to Shannon's entropy equation, the number of bits needed to encode a string of symbols is given by

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

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

$$\begin{aligned} B_{\text{sh}} &= -\sum_{j=1}^{N} \frac{N_i}{N} \log_2 \frac{N_i}{N} \\ B_{\text{av}} &= \frac{1}{N}\lceil N \cdot B_{\text{sh}} \rceil \leq \frac{1}{N}(N \cdot B_{\text{sh}} + 1) \\ &\Rightarrow B_{\text{av}} \leq B_{\text{sh}} + \frac{1}{N} \end{aligned}$$

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

7.2.2 Binary Chaotic Arithmetic Coding (BCAC)

In the previous section we explained how arithmetic coding can be viewed as reiteration on chaotic maps. AC is more commonly implemented in binary mode to reduce the computational requirements of video coders. For same considerations, we discuss implementation and security issues with BCAC in this work after introducing CAC in last section. The Binary CAC (or BCAC) uses either of the eight equivalent skewed binary maps (shown in Fig. 7.2) based on an input key. These maps differ from each other in the way input is mapped into the chaotic orbit—differ in the interval in which the arithmetic code must lie for a symbol '0' or '1' but

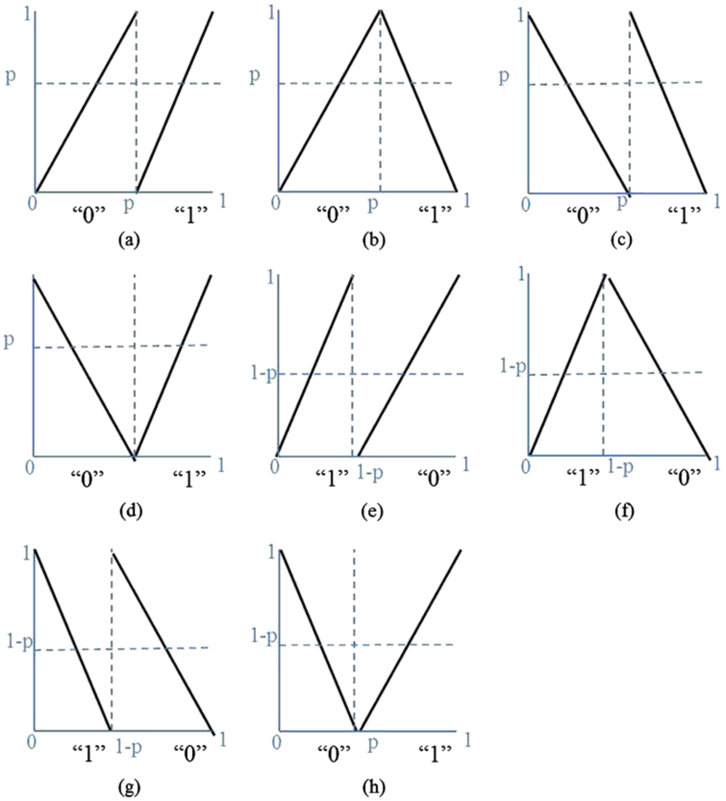


Fig. 7.2 (**a**)–(**h**) show the eight modes of the skewed binary map ($p = 0.6$)

the width of interval remains the same. In next section, we will formulate a mathematical procedure to generate the eight maps and choose between them using the parameter i.

We define the generalized skewed binary map with the following equations:

$$y = \begin{cases} n_1 x + c_1 & \text{when } x \le k \\ n_2 x + c_2 & \text{when } x > k \end{cases} \tag{7.2}$$

$$\text{Decode} \begin{cases} \text{‘0’} & \text{when } x \in [i1, i2] \\ \text{‘1’} & \text{when } x \in [i3, i4] \end{cases} \tag{7.3}$$

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

$$x = \begin{cases} m_1 y + b_1 & \text{when ‘0’} \\ m_2 y + b_2 & \text{when ‘1’} \end{cases} \tag{7.4}$$

n_1, n_2, c_1, c_2, m_1, m_2, b_1, and b_2 values can be precomputed for different maps and stored in a table for look-up for fast access. Table 7.1 gives the value of these parameters for all eight chaotic maps.

Table 7.1 Parameter list for the eight possible choices of chaotic encoder

Parameter	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)
M1	p	p	$-p$	$-p$	p	$-p$	$-p$	p
B1	0	0	p	p	$1-p$	1	1	$1-p$
M2	$1-p$	$p-1$	$p-1$	$1-p$	$1-p$	$1-p$	$p-1$	$p-1$
B2	p	1	1	p	0	0	$1-p$	$1-p$
N1	$1/p$	$1/p$	$-1/p$	$-1/p$	$1/(1-p)$	$1/(1-p)$	$-1/(1-p)$	$-1/(1-p)$
C1	0	0	1	1	0	0	1	1
N2	$1/(1-p)$	$-1/(1-p)$	$-1/(1-p)$	$1/(1-p)$	$1/p$	$-1/p$	$-1/p$	$1/p$
C2	$-p/(1-p)$	$1/(1-p)$	$1/(1-p)$	$-p/(1-p)$	$(p-1)/p$	$1/p$	$1/p$	$(p-1)/p$
I1	0	0	0	0	$(1-p)$	$(1-p)$	$(1-p)$	$(1-p)$
I2	p	p	p	p	1	1	1	1
I3	p	p	p	p	0	0	0	0
I4	1	1	1	1	$1-p$	$1-p$	$1-p$	$1-p$
K	p	p	p	p	$1-p$	$1-p$	$1-p$	$1-p$

Grangetto et al. [9] 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 Figs. 7.2(a) and (e)). Similarly, KSAC [30] can be represented in terms of piece-wise linear maps by removing the condition of continuity of individual maps ($\rho_i(x)$). Each part ρ_i maps a discontinuous interval on x-axis to the interval [0,1).

7.2.3 *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 BCAC, both end of interval are updated at every iteration using a linear transformation $x = my + b$ 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 look-ups (an 8-input LUT required for BCAC to choose the exact chaotic map) involved in chaotic coding to choose the right chaotic map at every iteration which can be efficiently implemented in software or hardware. Thus, CAC encode requires more computations than BAC encode while CAC decode requires less computations than BAC decode.

7.3 Security

7.3.1 *Application to Multimedia/Data Encryption*

CAC is Shannon-optimal in terms of compression efficiency. By varying the mapping $\rho_k \rightarrow s_i$, we can obtain different maps, all of which give same compression efficiency but different intervals for the 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.

As such, the CAC (or BCAC) can be used as a joint compression-cum-encryption technique for data encryption tasks particularly in low-power embedded systems such as surveillance cameras, video sensors, mobile phones and netbooks/ipads. For BCAC, we have eight possible maps for every encoded bit (see Fig. 7.2) giving up to

3 bits of encryption key per encoded symbol. The large keyspace makes the scheme secure against exhaustive trails. 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. Most arithmetic coders in practice are binary, i.e. work with only two input symbols '0' and '1' because of the resulting large complexity of arithmetic coder when using multiple symbols [19]. We have a keyspace of $3N$ bits for N bit plaintext. The large keyspace makes it extremely difficult to launch brute-force attacks. Since the key remains the same for multiple iterations, in effect the effective length of key bits is much less than the plaintext bits.

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 [35]. Scalable Multimedia Encryption finds its applications in modern pervasive/cloud-based multimedia applications where different types of user want to access the same multimedia content at different resolutions and access-privileges.

7.3.2 Threat Model

BCAC coder/decoder pair is treated as encryption/decryption oracle, respectively. In our threat model, an attacker is able to choose a plaintext of chosen length and obtain its corresponding ciphertext from the encryption oracle. He can repeat this process for at most P times. In other words, we allow the attacker to adaptively select plaintext and use the encryption/decryption oracle for a polynomial number of times.

7.3.3 Security Enhancements (SE)

As mentioned previously (in Sect. 7.2), arithmetic coding-based encryption schemes (such as RAC and SAC) have been found to be vulnerable to cryptographic attacks (chosen-ciphertext attack [37], ciphertext-only attack and chosen-plaintext attack [28]).

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 three SE modes for BCAC, which add considerable levels of security to the design. In this discussion, we use the following convention:

The encryption oracle has a BCAC encoder and some added operators for SEs. K_{3N} is the initiating key value for the encryption oracle. $I(n)$ and $O(n)$ are the input and output sequences of encryption oracle for nth iteration. $PT(n)$, $CT(n)$ and $K(n)$ are the Plaintext, Ciphertext and Key values for the nth iteration of BCAC coder. The length of $PT(n)$ and $I(n)$ is N bits, the compressed outputs $CT(n)$ and $O(n)$ have $M(n)$ bits ($M(n) \leq N$ in general), K_{3N} and $K(n)$ have $3N$ bits.

Algorithm 1 PM mode

```
{$PT(n)$}: Input to CAC encoder for $n$th pass
{$CT(n)$}: Encoded Output of CAC Encoder for $n$th pass
{$K(n)$}: Key value for $n$th pass
{$I(n)$}: Plaintext Input for $n$th pass
{$O(n)$}: Ciphertext Output for $n$th pass
{$K_{3N}$}: Initiating Key
{$W$}: PM parameter
PM_mode()
$K(n) = K_{3N}$
for $w = 1; j \leq W; j++$ do
    $PT(w) = I(w) \oplus K(w)$
end for
$O = CT =$ CAC.encode($PT, K$)
```

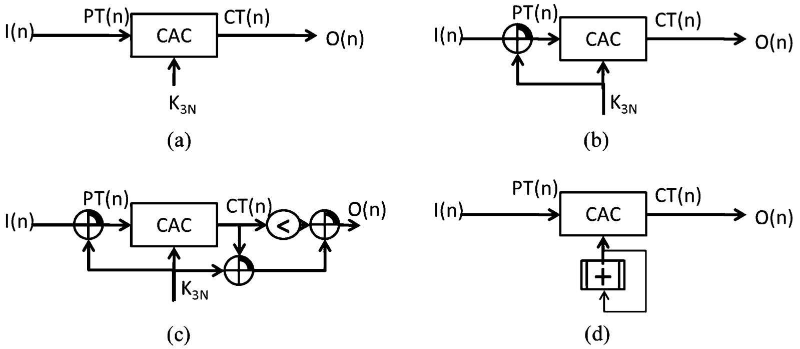
7.3.3.1 Plaintext Modulation (PM) Mode

In the Plaintext modulation mode, the first F bits of input plaintext are XORed with the input key values. The value of F is chosen according to application requirements. When $F \lll N$ (F is small), there is negligible losses in compression performance but the security level obtained is low. On the other side, when $F \approx\approx N$, the security level is highest but the compression performance will be compromised. Thus, it is appropriate to choose large values of N (say $N = 1000$) and relatively small f ($f \approx 30$). The stepwise details of BCAC+PM algorithm is given in Algorithm 1 and shown in Fig. 7.3(b). The idea of XORing the first few bits of plaintext with key, make it behave like a one-time pad at the beginning of BCAC. Moo and Wu [23] discuss how it is extremely difficult (exponential in F) to reconstruct the remaining plaintext in arithmetic coding in case, we lose the first F bits. However, in case of BCAC encoding, this complexity compounds manifold because of large keyspace for two reasons:

1. The uncertainty in knowledge of exact chaotic interval at end of first F iterations of BCAC is of the order of 8^F against 2^F in case of arithmetic coder. This is because, unlike BAC where any interval can split into either two intervals, the interval can split into eight intervals in case of BCAC.
2. Lack of key information implies that decoding of remaining bitstream is impossible, even if we are able to resynchronize the end interval at the end of encoding first F bits.

7.3.3.2 Key and Output Mixing (MIX) Mode

In the MIX mode, $CT(n)$ and $K(n)$ are mixed with each other to obtain $O(n)$ and $K(n+1)$. This operation is performed as follows:



(a) (b) (c) (d)

Fig. 7.3 Different modes for SEs in CAC. (**a**) Normal CAC encoder, (**b**) PM mode—the N plaintext bits are mixed with the key, (**c**) MIX mode—$3N$ key bits are mixed with compressed output, the compressed output is rotated left by L arbitrary bits and mixed with M bits of mixed key, (**d**) PWIK mode—a new pair-wise-independent key is generated in each iteration by adding a Initial Value modulo a prime p in GF(256)

Algorithm 2 Mix mode

```
Mix_mode()
    {$A(n) \oslash B(n)$}: XOR $A$ with cyclically extended or shrunk $B$, i.e. $A(n) \oplus B(n \text{ modulo}(\text{size}(B)))$ for $1 \le n \le \text{size}(A)$
    {$ROL(a, b)$}: Rotate $b$ to left by a bits
    {$L$}: Number of bits to be rotated to left
$PT(n) = I(n)$
$CT(n) = $CAC.encode($PT(n), K(n)$)
$K(n+1) = K(n) \oslash O(n)$
$O(n) = ROL(L, O(n)) \oslash K(n+1)$
```

1. We XOR $K(n)$ and $O(n)$ to obtain $K(n+1)$. Since $O(n)$ is of length $M(n) \le 3N$, we cyclically repeat $O(n)$ to make it $3N$ bits long.
2. We rotate $O(n)$ with L arbitrary bits to the left (cyclical left shift). This operation is easily performed in hardware using wire permutations, and in software using simple command for left rotate.
3. We XOR rotated $O(n)$ with first $M(n)$ bits of $K(n+1)$ to get the ciphertext output $O(n)$ of the encryption oracle.

All XOR operations can be implemented cheaply in commercial hardware and software. See Algorithm 2 and Fig. 7.3(c) for description and figure. There is no loss in compression efficiency or throughput of the system.

Mix mode allows efficient mixing of Key and ciphertext, making it unintelligible for an attacker to recover the relationship between input and key values using output values.

Algorithm 3 PWIK mode

```
PWIK_mode( )
$\{PR\}$: Largest prime in GF($2^{256}$)
$K(0) = K_{3N}$;
$PT(n) = I(n)$
$K(p) = (K(p-1) + \text{InitValue}_2) \mod 2^{256}$
if $(K(p) < \text{InitValue}_2)$ then
    return $K(p) = K(p) + 2^{256} - PR$
end if
$CT(n) = \text{CAC.Encode}(PT(n), K(n))$
$O(n) = CT(n)$
```

7.3.3.3 Pair-Wise-Independent Keys (*PWIK*) Mode

In *PWIK* mode, independent keys are generated for each iteration of the BCAC coder using initial key value. The same values can be reconstructed in the decoder side with prior knowledge of these initial values. However, the generated key values are pair-wise independent from each other. This method uses Galois field mathematics and we take $3N \leq 256$ or $N = 85$ for BCAC to simplify the operation. The generated keys are shown to be pair-wise independent by Jutla et al. [12]. There is no loss in compression efficiency or throughput of the encoder. See Algorithm 3 and Fig. 7.3(d) for details. This mode has the restriction that N is should be keep to a value such that $3N \leq 256$ or an exponent of 2 (for efficient finite field implementation).

7.3.4 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 [7] and DES [8] also. We 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 [13] 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 the KSAC work has shown serious weaknesses of these permutations [11, 28, 37, 38]. Jakimoski and Subbalakshmi [11] present a cryptanalysis of KSAC where they reveal that a key of length 2000 bits can be broken with as few as 50,000 plaintexts.

The decoding algorithm for CAC involves iteration on chaotic maps. Kocarev [14] discusses how various properties of chaotic maps have direct correlation to cryptographic algorithms. For example, decoding CAC with any slightly wrong value (making a wrong guess) will make the output appear random even if correct knowledge of maps is given. This is analogous to diffusion property of cryptographic ciphers. Similarly, the iterations on chaotic maps (1 iteration per encoded bit) is similar to rounds in encryption algorithms. Thus, the chaotic decoder will behave like a random number generator and without exact knowledge of key (coding parameters) and initial seed (coded message), the output of decoder will be completely uncorrelated with the encoded message. This property of chaotic maps implies that unlike BCAC, two closely related plaintext values may be mapped to completely different (random) output values even with same key. The same message will be mapped to completely different output value with two closely related key values.

This makes it difficult to launch **known-plaintext attacks** on our system. It is, however, possible to mount **chosen-plaintext attacks** on the system because an attacker can modulate the plaintext input to iteratively guess the key stream beginning from first bit of plaintext (last bit of BCAC encoder). Such an attack has been mounted against KSAC [11].

The proposed SEs can alleviate the attacks at little computational overhead.

For BCAC+PM, the first few bits of plaintext are modulated, these bits and the key bits are unknown to the encoder. Therefore, it will be impossible for an attacker to observe and infer any correlation from chosen plaintexts. In BCAC+Mix operation, at every iteration we XOR the key with the output of encoder and update the key. This randomizes the output (like a one-time pad) for an adversary to draw any inference. The BCAC+PWIK mode allows us to resist chosen- and known-plaintext attacks because the keys used in different iterations are pair-wise 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 (Mix and PWIK) have no effect on the compression efficiency, which is a significant advantage against some proposed techniques [2, 13, 24]. A drawback of PWIK mode is that it involves GF mathematics: the length of input bits should suit the GF operations. For example, with GF(2^{256}) implementation, the length of plaintext will be 85 bits.

BCAC, like arithmetic coding, is more sensitive to errors in the decoded bitstream for errors in the beginning of the stream and not to those which are towards the end. However, BCAC+Mix mode has bit rotate and XOR operations which mask this property from the adversary.

7.3.5 *Comparison with BAC+AES*

BAC followed by encryption with AES is the naive candidate which should provide best security. AES is extremely fast when it is fully pipelined in hardware [36]. However, the sequential nature of BAC coder becomes the bottleneck in a combined BAC+AES system.

The arithmetic operations required for one bit encoding or decoding using BAC is two adders and one multiplier (discussed in Sect. 7.2.3). AES-128 bits require 40 sequential transformation steps composed of simple and basic operations such as table look-ups, shifts, and XORs. It needs approximately 336 bytes of memory and approximately 608 XOR operations or roughly 3 bytes memory and 5 XOR operations per bit of encoding.

BCAC coder requires two adders and two multipliers for encoding and only one adder and one multiplier for decoding. Thus, the hardware requirements of BCAC coder are much less than BAC and AES combined. The BCAC decoder is particularly simpler than CAC decoder (without AES), which is desired for most common video applications which involve real-time decoding in mobile and embedded devices.

7.3.6 *Key and Plaintext Sensitivity*

Confusion and diffusion are two important properties desired for operation of a secure cipher. Confusion refers to making the relationship between the key and the ciphertext as complex and involved as possible. This makes it very hard to find the key even if one has a large number of plaintext–ciphertext pairs produced with the same key. In particular, changing one bit of the key should change the ciphertext completely.

We conducted experiments for different values of N by changing one bit in either of the N symbols comprising the key. In the ideal case, for a single bit flip, the ciphertext output must be changed from original value in $\frac{N}{2}$ positions, as was the case with BCAC. Figure 7.4 shows the plot for $N = 100$ and $p = 0.7$ with mean value of 1000 simulations. BCAC, BCAC+PWIK and BCAC+Mix have same encoder output for the first iteration, so we have used a single line to represent this. It can be seen that all the schemes (including BCAC+PM) give a Hamming distance of new ciphertext to a value around 50. Over different iterations, BCAC+Mix will eventually lead to different key being used in different iterations, leading to a variable Hamming distance. This increases the confusion property of the scheme and it is shown in Fig. 7.5.

Diffusion means that the output bits should depend on the input bits in a very complex way. This is ensured by the arithmetic coding (or CAC) scheme itself because the relationship between input and output bits is non-linear. The input bits iteratively decide the interval of final output, which is then used to obtain the shortest length code from that interval. However, it has been observed in case of BAC

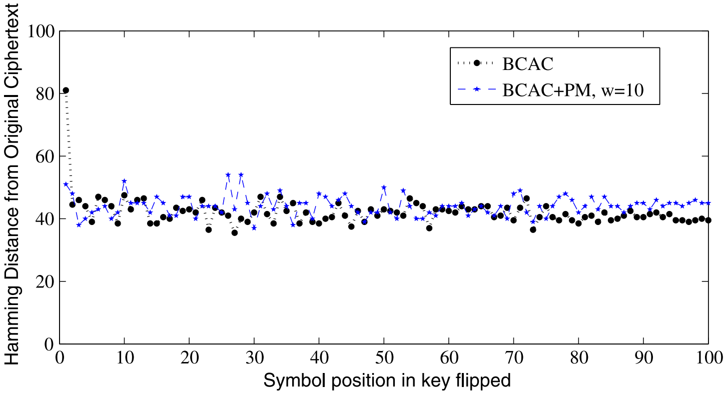


Fig. 7.4 Plot showing the number of bit flips in output text with change in one corresponding symbol in the key

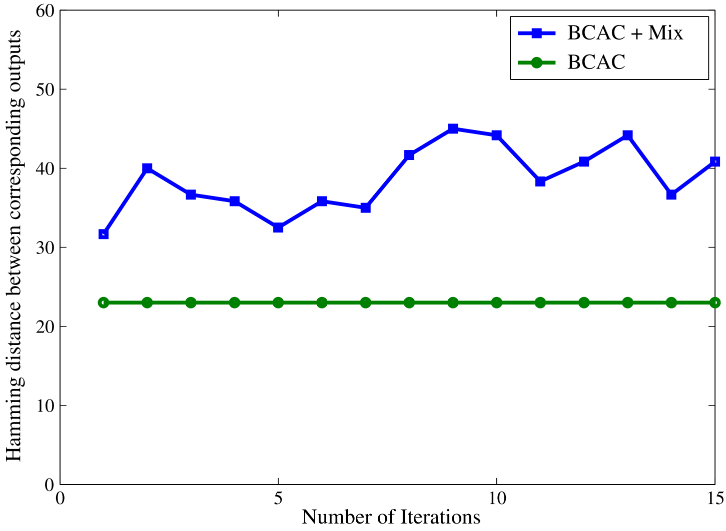


Fig. 7.5 Plot of Hamming distance between correspondingly coded values when a single bit of key ($N = 50$) was changed

that the last few bits have less impact on the first bits of the ciphertext. We conducted an experiment where plaintexts were varied one bit at a time and Hamming distance of new ciphertext over last ciphertext was reported. The mean value over 1000 such simulations is reported in Fig. 7.6. BAC and BCAC show similar trend vs. change in plaintext bits. BCAC+PWIK, BCAC+PM have similar performance trend as BCAC.

BCAC+Mix mode result in mixing of encoder output with the key value and this mixing leads to an excellent value of Hamming distance, as plotted in Fig. 7.6.

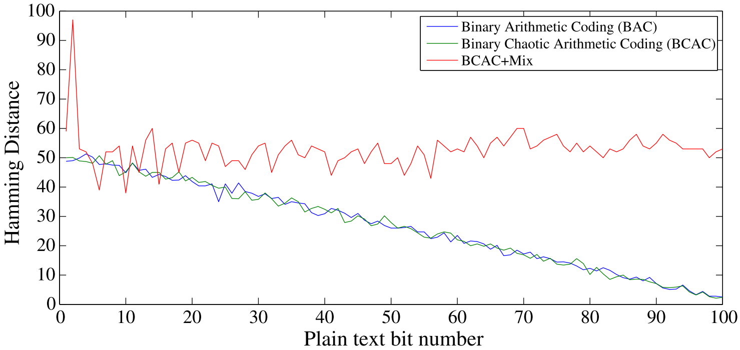


Fig. 7.6 Plot showing the number of bit flips in output text vs. change in a single bit position in the plaintext

Table 7.2 Results for image reconstruction quality with wrong decoding keys. Selective encryption of DWT coefficients was done using BCAC

Image	0.1 % encryption		0.4 % encryption	
	SSIM	PSNR	SSIM	PSNR
Tank	0.057	−6.32	0.00	−11.55
Couple	−0.23	−6.56	−0.03	−8.5
Girl	−0.05	−4	0.005	−7
Grass	−0.08	−6.57	0.002	−11.43
Peppers	0.06	−7.936	0.012	−10.51
San Diego	0.111	−6.2	0.06	−10.22
House	0.16	−4.17	0.035	−4.21

7.3.7 Selective Encryption Using BCAC

In this section, we present results for selective image encryption using BCAC. The sample images were taken from USC SIPI database and each has resolution 512 × 512 pixels. Two cases are considered: (a) Encryption of LL coefficients and (b) all coefficients of 6th level DWT decomposition. This corresponds to encryption of 0.1 % and 0.4 % coefficients. SSIM (Structural Similarity Index) and PSNR (Peak Signal to Noise Ratio) were considered to measure the image reconstruction with corrupted keys. The results are presented in Table 7.2. It can be observed that SSIM—Structural Similarity is close to 0 while PSNR is negative, which indicate strong de-correlation between wrongly reconstructed image with original image.

7.4 Compression

BCAC gives the same compression efficiency as the BAC coder. We performed some experiments to verify these facts. We ran an implementation of BCAC over

Table 7.3 Compression performance of BAC and BCAC for various length strings. The average length of codeword is presented for various p values and lengths of input string

p	$N = 10$		$N = 100$		$N = 1000$	
	BAC	BCAC	BAC	BCAC	BAC	BCAC
0.5	0.025	8.733	120	108.16	999.1	999.84
0.55	9.025	8.983	98.17	98	992.30	992.34
0.6	8.7899	8.882	95.95	95.84	970.98	971.14
0.65	8.442	8.316	91.90	91.23	934.30	934.04
0.7	7.918	7.936	86.96	86.37	881.07	881.84
0.75	7.47	7.54	80.31	80.37	811.27	811.84
0.8	6.701	6.333	71.11	71.03	721.07	720.84
0.85	5.551	5.342	60.28	59.1	609.30	609.84
0.9	4.122	4.055	45.66	46.39	469.06	468.84
0.95	2.698	2.773	27.46	28.8	287.00	286.34

Matlab 7.11.0.584 (R2010b) 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 slightly 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 7.3. (The reported value is the average length of output bitstream and the standard deviation.) 1000 simulations each were run in Matlab to obtain the mean value of output bitstream lengths.

Figure 7.7 gives the relative compression performance of CAC, BCAC and various SEs (for $N = 100$). BCAC+PM has a slight compression overhead for $w = 5$ or 10, but it increases drastically for $w = 20$, making $w = 20$ unsuitable for practical applications. PWIK and MIX modes (not shown in figure) have similar compression as BCAC.

7.5 Hardware Implementation

In the regular coding mode, prior to the actual arithmetic coding process the given binary data enter the context modeling stage, where a probability model is selected such that the corresponding choice may depend on previously encoded syntax elements. Then, after the assignment of a context model, the bin value along with its associated model is passed to the regular coding engine, where the final stage of arithmetic encoding together with a subsequent model updating takes place (see Fig. 7.8). We shall restrict the focus of further discussions on the final arithmetic encoding (and decoding) stages of CABAC coder.

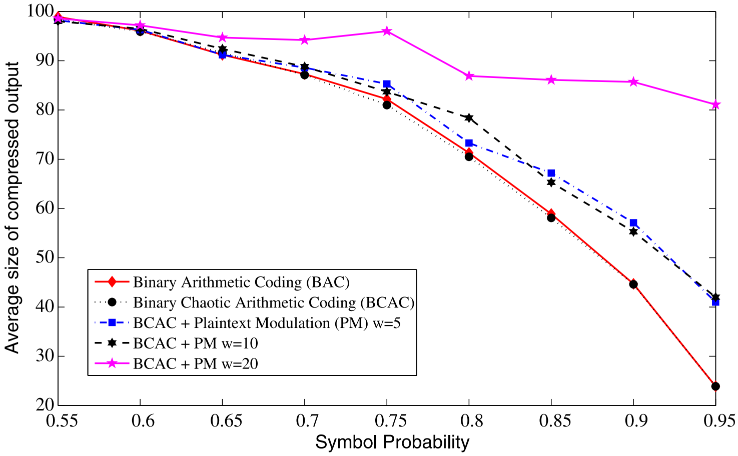


Fig. 7.7 Compression performance of proposed schemes

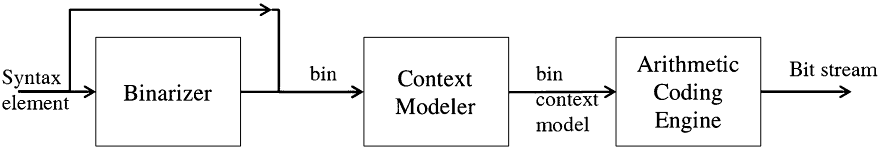


Fig. 7.8 Block diagram of CABAC coder

7.5.1 Literature Review

Adaptive minimum-redundancy (Huffman) coding is expensive in both time and memory space, and is handsomely outperformed by adaptive AC besides the advantage of AC in compression effectiveness [21]. Fenwick's structure requires just n words of memory to manage an n-symbol alphabet, whereas the various implementations of dynamic Huffman coding [6, 29] consume more than 10 times as much memory [22].

Hardware architectures have been proposed in research literature for arithmetic coding using CACM model [31] or related work [10, 15, 21]. CABAC or Context-Adaptive Binary Arithmetic Coder is used in H.264 AVC and SVC. The critical path of coder is the multiplier, which is removed in CABAC and recent implementations [5, 17, 25] by using a look-up approximation (leading to some compression inefficiency).

There has been little work [1, 26], however, in implementation of chaotic maps on hardware. However, the recent trend toward joint compression and encryption using chaotic maps and arithmetic coding for low-power embedded systems would be greatly complemented by an efficient hardware architecture, as presented in this paper.

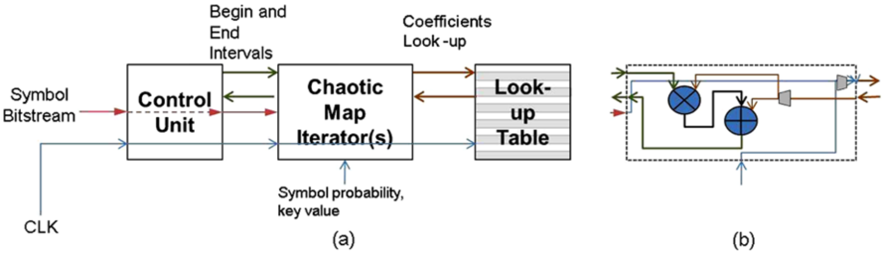


Fig. 7.9 Generalized Hardware Architecture for Chaotic Maps. (**a**) Generalized architecture and (**b**) Circuit details for Chaotic Map Iterator

Many designs have been proposed for accelerating the CABAC decoding. The major timing limitation is caused by the BAC, because it is inherently dependent on control statements and arithmetic operations. As a result, the BAC becomes a throughput bottleneck of the entire JPEG2000 encoding system, but its serial processing nature makes it difficult to exploit parallelism.

7.5.2 Implementation Details

The chaotic encoder operation inverse inverse mapping of interval $[0, 1)$ on the chaotic map according to input symbol. For binary arithmetic coder, we have a fixed map to be iterated in each cycle.

Figure 7.9(a) shows the basic architecture for coding using chaotic maps. The control unit receives the input bit stream, which is passed on to the chaotic map Iterator (CMI). The control unit passes the bitstream, one symbol per cycle (unless in the case of multiple-symbol encoding, which will be discussed later). For encoding, the initial interval passed to CMI is [0, 1), which is transmitted as the beginning (B_n) and end (E_n) interval values. Both the intervals are then iterated over CMI (using two instances of CMI), and then the output is sorted so that $B_n < E_n$. If the difference $(D_n = E_n - D_n)$ is lower than a threshold, we need to renormalize the encoder. The renormalization procedure for arithmetic coding has been discussed in [21]. A similar extension of renormalization procedure may be possible for chaotic maps. But, for the evaluation designs considered in this work, we have considered 64 bit encoder without any renormalization procedure.

In case of decoding, Control Unit (CU) transmits the coded symbol into CMI, which is then iterated over Piece-wise linear map and reported back to CU. The CU makes a comparison with chaotic map indicated by the key and outputs a single bit output.

CMI has a multiplier and an adder to perform chaotic iteration. The internal details of this operation are given in Fig. 7.9(b). The multiplication and addition coefficients are obtained from a look-up table/RAM collating the input symbol, key value and probability value as the input address. The Look-ed up value or a word is demultiplexed to obtain the multiplication and addition coefficients. This option can work fine for at most binary case, and for the case where p value is limited to fixed precision, say 8 bits. Such fixed precision approximations have been introduced in CABAC [19], however, it leads to approximation of results. Alternatively, we can use a multiplexer which can implement look-up using physical circuits to compute the return values. The second approach has been implemented in this work, as it allows more flexibility in design and accuracy in computation.

For implementation, the input and output intervals to the Chaotic Map Iterator are represented in 64 fixed point (0 bits integer and 64 bits fraction, shortly I.F 0.64) arithmetic. The symbol probability has been quantized to 8 bits (I.F 0.8).

7.5.3 Binary Arithmetic Coder (BAC) Architecture

To implement BAC in proposed architecture, we target a design which processes 1 symbol (1 bit in this case) per cycle. The CMI has 1 bit symbol input, 8 bit symbol probability and no bits for choice of chaotic map (there is only one map in this case). The 9 bit look-up can be implemented using a 512 words RAM or Look-up Table. One word is 16 bits—8 bits each for multiplication and addition coefficients. Alternatively, this can be implemented using a multiplexer and hardware adder/subtracter to obtain the coefficients. The latter approach was used for BAC implementation. The design was synthesized in Xilinx Virtex-6 XC6VLX75t FPGA using Xilinx ISE Design Suite 12.0 environment. The same target FPGA, which is one of the low end Virtex-6 family member is used in all synthesis/translate/map/place and routes.

The two 64×8 bit multiplications are mapped in hardware into 10 DSP48E slices. A slice usage of 302 was obtained and the design achieved a clock frequency of 510 MHz, with one symbol per clock cycle. The optimized implementation of multiplication, using carry-chains of FPGA fabric was synthesized to remove the use of DSP slices. This implementation requires 1585 slices and achieves a clock frequency of 500 MHz. The throughput of this implementation is 1 bit per cycle with a 500 MHz clock, i.e. 500 Mbps.

7.5.4 Binary Chaotic Arithmetic Coder and Encryption (BCAC) Architecture

The architecture for BCAC differs from binary arithmetic coder in the sense that the choice of chaotic map is made based on a key value, and is not precomputed. For this implementation, the CMI has 1 bit symbol input, 8 bit symbol probability and 3 bits for choice of chaotic map (for binary case $N = 2$, hence number of different chaotic maps is $N2^N = 8$. The 12 bit look-up can be implemented using a 512 words RAM or Look-up Table, with 16 bits word. Alternatively, we used 8-to-1 multiplexer to obtain the coefficients corresponding to a key, each coefficient being generated based on value in Table 1 in [27]. The implementation on target FPGA gave a clock frequency of 500 MHz, utilizing 321 slices and 10 DSP48E1 slices (which have optimized multiplier and accumulator operation implemented in VLSI). Mapping these multiplication to FPGA logic increased the slice usage to 1474, without any change in achievable clock frequency.

The BCAC decoder hardware utilization was 173 slice LUT with five DSP slices (806 slice LUTs with LUT multiplier) with a clock frequency of 510 MHz (500 MHz). The 64 × 8 bit multiplier is implemented by ISE into five DSP slices. However, the same multiplier can be optimized and implemented without hardware multipliers using other multiplier such as square root multiplier, reconfigurable constant multipliers etc. The hardware requirements are basically dependent on size of Look-up logic which increases exponentially with increase of N. The throughput of this implementation is 1 bit per cycle with a 510 MHz clock, i.e. 510 Mbps. To consider the area effectiveness of this design, we consider throughput per slice, with the second implementation where we implement multiplication in LUTs rather than using DSP48E1 slices present in device. The throughput/slice for this design is obtained as 322 Kb/slice.

7.5.5 Cost of Encryption

Comparing the BAC and BCAC architectures, we obtain a zero latency, same throughput and little hardware overhead (20 slice LUTs) in implementing this encryption scheme against AES or other schemes which have significant overhead. For instance, Chang et al. [3] reports AES implementation using 156 slices, two Block RAMs to obtain a lower clock of 306 MHz.

To increase the throughput per slice for a bitstream, we intuitively consider the dimension of increasing the number of symbols in dictionary used in arithmetic coding. For example—considering three or four symbols in the dictionary.

7.5.6 N-ary Chaotic Arithmetic Coder and Encryption (NCAC) Coding

N-ary arithmetic encryption using the entire possible key space quickly turns out-of-bounds for a FPGA device. Moving from two to three piece-wise linear maps, we have a tremendous increase in key size. We implemented tri-nary CAC coder in FPGA device to obtain a device usage of 492 slices and 10 DSP48E slices (1800 slices without DSP slices), but the achievable clock frequency dropped to 127 MHz. The tri-nary decoder hardware utilization was 419 slice LUT with five DSP slices (1052 slice LUTs with LUT multiplier) with a clock frequency of 442 MHz (369 MHz). The hardware requirements are basically dependent on size of Look-up

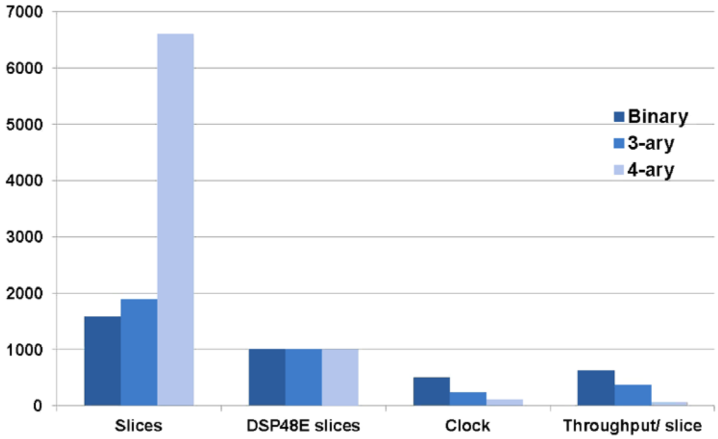


Fig. 7.10 N-ary arithmetic coding and encryption architectures: Comparative performance. The # of slices, # of DSP slices (×100), clock frequency (MHz) and throughput per slice (×1000) are reported in the figure. It can be observed that increasing the size of dictionary significantly reduces the throughput. The figure is drawn by scaling the throughput/slice legend to consider the fact that a four symbol dictionary will require half the words of a two symbol dictionary

logic which increases exponentially with increase of N ($N|!2^N$), making it infeasible to scale-up the throughput/slice.

A simple way to restrict this band-width explosion is to used the algorithm for encryption proposed in [32]. They restrict the keyspace and instead use only a small fragment of keys from the entire range, for encryption. However, the approach presented in [32] has other computationally inefficient parts.

The results are shown in Fig. 7.10. The number of slice LUTs is reported directly, number of DSP slices is scaled directly and clock frequency is measured in MHz. The throughput comparison is tricky because using a 4-symbol dictionary (4-ary coding) will lead to reduced bitstream (around 50 % reduction) than the bitstream generated by 2-symbol dictionary. Thus, to compare these values on a graph, we multiply each throughput with N value (2 for binary) to indicate relative throughput. It can be observed that increasing the size of dictionary significantly reduces the throughput, even after such considerations due to exponential increase in hardware usage for key implementation.

Although our experiment to scale to multiple-symbol dictionary failed, the reason is not the same as for traditional designs for arithmetic coding [19]. Rather, the key explosion is the main reason for such limitations. We next consider increasing the system throughput by encoding multiple binary symbols in a single pass. This approach is different from the previous approach in the sense that multiple probability values are not involved.

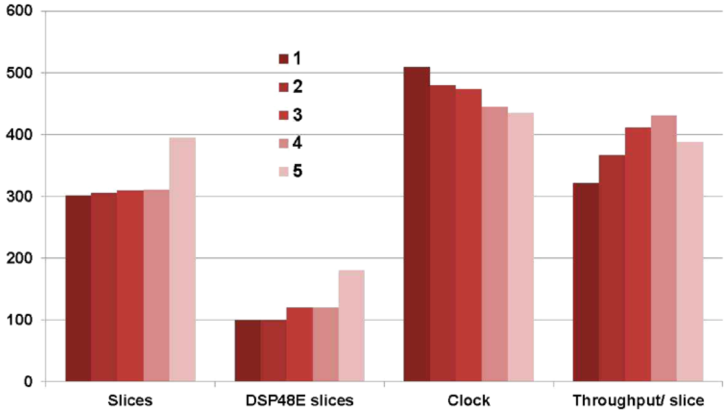


Fig. 7.11 Multiple symbols per cycle (BAC): Comparative performance. The # of slices, # of DSP slices (×10), clock frequency (MHz) and throughput per slice (×1000) are reported in the figure. It can be observed that four symbols per cycle achieve highest throughput before LUT explosion due to increased precision and maps

7.5.7 Multiple Symbol per Cycle Arithmetic Coding

Let us consider the case of arithmetic coding where we want to encode two symbols in a single iteration of chaotic map. In this case, the chaotic map will spit into multiple (four instead of two) piece-wise maps. Arithmetic coding with encryption is still going to suffer with band-width expansion, but we observe that the band-width expansion is much less (or order of 2^N) instead of $N2^N$. Consider, for example the case where we want to encode two symbols together (‘01’ instead of ‘0’ and ‘1’ in two separate iterations) using BAC. In this case, the resultant chaotic iterator will have four (instead of two) piece-wise linear maps and their precision of implementation will be increased (16 instead of 8 bits). This analysis can be extended to three, four or more symbols.

In this case, the increase is caused by increase in fixed point precision of coefficients (and hence multipliers and adders), and increase in number of piece-wise maps. However, against the case of MCAC where there was a band-width explosion due to increase in key size, we observe a considerable different result of implementation on Virtex-6 device. These results are reported in Fig. 7.11. The results are interesting to note, because contrasting with the traditional notion of one-symbol per cycle, we show that we can scale up to four symbols per cycle and achieve a higher throughput per slice. As we go from two to four, we observe a increase in throughput which is then checked by the exponential increase in hardware resources caused by multiple-symbol use. This value of 4 cannot be a device constraint (restrictions due to finite area or size of device) because the pure LUT mapping based implementation requires only 5480 slices out of 43,000 slices present in target xc6vls75 device. The highest throughput achievable is 431 Kbits per slice for the case of four symbols.

For the sake of brevity, we have restricted our discussion in last sections to NCAC and multiple-symbol BAC encoder, but the same trend follows for the decoder also.

7.6 Related Work

While we presented chaotic arithmetic coding in this paper, it is worth presenting other notable work related to entropy coding using the approach of joint compression and encryption. We present a brief overview here, with pointers to original research articles for further readings.

7.6.1 Multiple Huffman Tables

The encryption cipher and the entropy coder do bear some resemblance in that both of them turn the original data into redundancy-free bit streams, which cannot be decoded without certain information. For encryption, the information is the key; for entropy coding, the information is the statistical model. It is important to explore whether hiding this model could effectively prevent decoding of the compressed bit stream. The authors in [33] propose the use of m statistical models instead of one to overcome the problem of limited key/model space. The m models are alternatively used to encode the input bit stream. By hiding the manner of swapping these models and using it as a key, the size of key space increases exponentially in m. The basic idea of this scheme, called Multiple Huffman Table (MHT) is enumerated below:

1. Choose m different Huffman coding tables. They are numbered from 0 to $(m-1)$.
2. Generate a random vector P constituting of n integers of length $\lceil \log_2 m \rceil$ bits.
3. Use $p_{i \pmod n}$ table to encode ith symbol.

Multiple Huffman tables can be derived without sacrificing compression efficiency as follows: From a large pool of sample or training images or audio samples choose a representative set of images which represent the entire section of input images. This representative set can be used to obtain one table. Using different sets we can generate a large pool of Huffman tables.

Another, relatively simpler process can be used to generate Huffman tables. This is called Huffman tree mutation. As shown in Fig. 7.12(a), a standard Huffman tree has every left-hand-side branch labeled ‘0’ and every right-hand-side branch labeled ‘1’. If we permute the label-pairs as indicated in Fig. 7.12, we will get a new Huffman tree as shown in Fig. 7.12(b). For a Huffman table with t entries, its Huffman tree would have t leaves and $(t-1)$ inner nodes and label-pairs, which provides us the opportunity to make $(t-1)$ decisions about whether to permute each label-pair. Therefore, 2^{t-1} Huffman trees can be derived.

The major advantage of MHT method is achieving encryption with a reasonably high level of security while compression is unaffected requiring almost negligible

Fig. 7.12 Procedure of MHT encoding

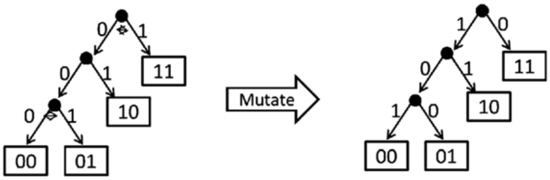
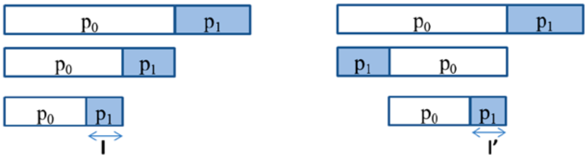


Fig. 7.13 Procedure of RAC encoding



additional overhead. Nevertheless, the basic MHT method is claimed to be only secure under cipher-only and known-plaintext attack and is vulnerable under chosen-plaintext attack. To improve the security, several kinds of enhanced MHT scheme have been proposed by either inserting random bit in the encrypted bit stream or integrating with a stream cipher [33]. Recently, another scheme via random rotation in partitioned bit streams has been reported [34] and has been applied to a MHT system [4]. In [39], the authors suggest empirical criteria for Huffman table selection to alleviate known-plaintext and chosen-plaintext attacks.

7.6.2 Randomized Arithmetic Coding

Grangetto et al. [9] propose JPEG2000 encryption by randomized arithmetic coding. They propose an efficient scheme to encrypt multimedia contents which adopt the entropy coding approach, and propose a scheme that provides conditional access by means of a modified AC stage. This scheme, called Randomized Arithmetic Coding or RAC exploits the fact that arithmetic decoding is very sensitive to errors in the compressed data, which tend to propagate throughout the decoded block. This otherwise undesirable property can be used to design a robust multimedia encryption cum compression algorithm. A single erroneous decoding step is able to cause an irreversible drift, thus making the data decoded any further completely useless. The proposed scheme changes the ordering of ‘0’ and ‘1’ intervals in a Binary Arithmetic Coder (BAC) based on a key. Figure 7.13 shows the compression of a symbol using two keys (say ‘000’ and ‘010’). The flip in second bit leads to pre-alignment of p_1 interval in arithmetic encoding process. The RAC approach can be applied to any AC, including adaptive and context-based ACs (CABAC used in H.264 standard) and their multiplierless approximations, which are very popular in the major international standards.

Although the arithmetic coder of the JPEG2000 pipeline is altered, the approach has no influence on the compression performance. This can be attributed to the fact that compression performance of AC is determined by the choice of symbol probabilities and length of symbol interval and not on interval beginning position. The basic idea of their approach is to change the order of the probability intervals in the arithmetic encoding procedure. It is a convention (agreed upon by both the encoder and the decoder) regarding the choice of interval (either that of the most probable or that of the least probable symbol) is the preceding one. In RAC, the ordering of the intervals is chosen securely randomly (by using a random bit from the PRNG). Selective/partial application of this encryption approach is possible.

Packet header information is left unencrypted. It is noted that their method might be susceptible to known-plaintext attacks, but it is argued that these kinds of attack are not relevant for the proposed encryption systems. Due to performance issues, Grangetto et al. propose the usage of a weak PRNG (with a 32 bit key) based on the standard rand function of the Linux C library. However, a key size of 32 bit is too short for serious security. However, more secure and efficient PRNG can be considered, e.g., AES in OFB mode.

Both total and selective encryptions are possible by choosing the layers or resolution levels to encrypt. Selective encryption of regions-of-interest is made possible since JPEG2000 is a code block based algorithm. To encrypt a region of interest, we have to apply the encryption on the codeblocks contributing to precincts of the region considered as presented in [20].

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 Figs. 7.2(a) and (e)).

7.6.3 Secure Arithmetic Coding

Wen et al. [30] modified the traditional AC by removing the constraint that a single continuous interval is used for each symbol. However, they preserve the sum of the lengths of intervals allocated to each symbol. This is achieved by splitting the intervals associated with one of the symbols using a key. This modified AC is called Key Splitting AC (KSAC), and it was shown that it can provide certain level of security while introducing vanishing coding efficiency penalty. Aiming to provide an AC system capable of offering high level of security, they suggested an enhanced version called the secure AC (SAC) [13], by applying a series of permutations at the input symbol sequence and the output codeword of the ISAC encoder.

Let $S = s_1 s_2 \dots s_N$ be the symbol sequence to be encoded, and the splitting vector be $K = k_1, k_2, \dots, k_N$. The encoding procedure of KSAC is described as follows:

1. Set the initial interval $I = [0, 1)$ and set $i = 1$.
2. Fetch a symbol s_j from S.
3. Partition the interval I according to $p(A)$ and $p(B)$ and k_i as shown in Fig. 7.14.

Similarly, KSAC [30] can be represented in terms of piece-wise linear maps by removing the condition of continuity of individual maps ($\rho_i(x)$). Each part ρ_i maps a discontinuous interval on x-axis to the interval [0, 1).

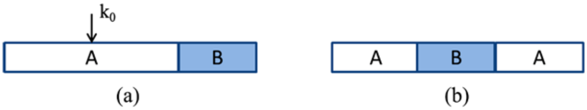


Fig. 7.14 Procedure of SAC encoding

7.7 Conclusion

In this work we presented a joint compression and encryption scheme for video/images using chaotic maps. We presented some SEs to alleviate the weaknesses of presented scheme against known cryptanalysis.

The presented scheme incurs no loss to compression performance, has a simpler decoder, while at the same time it encrypts data. It was shown to achieve higher throughput than the naive encryption algorithms.

References

1. Addabbo, T., Alioto, M., Fort, A., Rocchi, S., Vignoli, V.: Low-hardware complexity prbgs based on a piecewise-linear chaotic map. IEEE Trans. Circuits Syst. II, Express Briefs **53**(5), 329–333 (2006)
2. Bose, R., Pathak, S.: A novel compression and encryption scheme using variable model arithmetic coding and coupled chaotic system. IEEE Trans. Circuits Syst. I **53**(4), 848–857 (2006). doi:10.1109/TCSI.2005.859617
3. Chang, C.-J., Huang, C.-W., Chang, K.-H., Chen, Y.-C., Hsieh, C.-C.: High throughput 32-bit AES implementation in FPGA. In: IEEE Asia Pacific Conference on Circuits and Systems, APCCAS 2008 (2008)
4. Cheong, I.K., Huang, Y.C., Tung, Y.S., Ke, S.R., Chen, W.C.: An efficient encryption scheme for mpeg video. In: International Conference on Consumer Electronics, ICCE 2005 Digest of Technical Papers, pp. 61–62. IEEE Press, New York (2005)
5. Chuang, T.D., Chen, Y.J., Chen, Y.H., Chien, S.Y., Chen, L.G.: Architecture design of fine grain quality scalable encoder with CABAC for H. 264/AVC scalable extension. J. Signal Process. Syst. **60**(3), 363–375 (2010)
6. Cormack, G.V., Horspool, R.N.: Algorithms for adaptive Huffman codes. Inf. Process. Lett. **18**(3), 159–165 (1984)
7. FIPS 197: Announcing the Advanced Encryption Standard (2001)
8. FIPS 46-2: Announcing the standard for Data Encryption Standard (1993)
9. Grangetto, M., Magli, E., Olmo, G.: Multimedia selective encryption by means of randomized arithmetic coding. IEEE Trans. Multimed. **8**(5), 905–917 (2006). doi:10.1109/TMM.2006.879919
10. Howard, P.G., Vitter, J.S.: Analysis of arithmetic coding for data compression. Inf. Process. Manag. **28**(6), 749–763 (1992)
11. Jakimoski, G., Subbalakshmi, K.P.: Cryptanalysis of some multimedia encryption schemes. IEEE Trans. Multimed. **10**(3), 330–338 (2008). doi:10.1109/TMM.2008.917355
12. Jutla, C.S.: Encryption modes with almost free message integrity. In: EUROCRYPT, pp. 529–544 (2001)
13. Kim, H., Wen, J., Villasenor, J.D.: Secure arithmetic coding. IEEE Trans. Signal Process. **55**(5), 2263–2272 (2007). doi:10.1109/TSP.2007.892710
14. Kocarev, L.: Chaos-based cryptography: a brief overview. IEEE Circuits Syst. Mag. **1**(3), 6–21 (2002)
15. Langdon, G.G.: An introduction to arithmetic coding. IBM J. Res. Dev. **28**(2), 135–149 (1984)

16. Langdon, G., Rissanen, J.: Compression of black-white images with arithmetic coding. IEEE Trans. Commun. **29**(6), 858–867 (1981)
17. Lo, C.C., Tsai, S.T., Shieh, M.D.: Reconfigurable architecture for entropy decoding and inverse transform in H. 264. IEEE Trans. Consum. Electron. **56**(3), 1670–1676 (2010)
18. Mao, Y., Wu, M.: A joint signal processing and cryptographic approach to multimedia encryption. IEEE Trans. Image Process. **15**(7), 2061–2075 (2006)
19. Marpe, D., Schwarz, H., Blättermann, G., Heising, G., Wieg, T.: Context-based adaptive binary arithmetic coding in the h.264/avc video compression standard. IEEE Trans. Circuits Syst. Video Technol. **13**, 620–636 (2003)
20. Massoudi, A., Lefebvre, F., De Vleeschouwer, C., Macq, B., Quisquater, J.-J.: Overview on selective encryption of image and video: challenges and perspectives. Int. J. Inf. Secur. **2008**, 5–1518 (2008). doi:10.1155/2008/179290
21. Moffat, A., Neal, R.M., Witten, I.H.: Arithmetic coding revisited. ACM Trans. Inf. Syst. **16**(3), 256–294 (1998)
22. Moffat, A., Sharman, N., Witten, I.H., Bell, T.C.: An empirical evaluation of coding methods for multi-symbol alphabets. Inf. Process. Manag. **30**(6), 791–804 (1994)
23. Moo, P.W., Wu, X.: Resynchronization properties of arithmetic coding. In: Proceedings of International Conference on Image Processing, ICIP 99, vol. 2, pp. 545–549. IEEE Press, New York (1999)
24. Nagaraj, N., Vaidya, P.G., Bhat, K.G.: Arithmetic coding as a non-linear dynamical system. Commun. Nonlinear Sci. Numer. Simul. **14**(4), 1013–1020 (2009)
25. Osorio, R.R., Bruguera, J.D.: Arithmetic coding architecture for H. 264/AVC CABAC compression system (2004)
26. Pande, A., Zambreno, J.: Design and hardware implementation of a chaotic encryption scheme for real-time embedded systems. In: International Conference on Signal Processing and Communications (SPCOM), pp. 1–5. IEEE Press, New York (2010)
27. Pande, A., Zambreno, J., Mohapatra, P.: Joint video compression and encryption using arithmetic coding and chaos. In: IEEE Intl. Conf. Internet Multimedia Systems Architecture and Application (IMSAA) (2010)
28. Sun, H.-M., Wang, K.-H., Ting, W.-C.: On the security of the secure arithmetic code. IEEE Trans. Inf. Forensics Secur. **4**(4), 781–789 (2009). doi:10.1109/TIFS.2009.2031944
29. Vitter, J.S.: Design and analysis of dynamic Huffman codes. J. ACM **34**(4), 825–845 (1987)
30. Wen, J., Kim, H., Villasenor, J.D.: Binary arithmetic coding with key-based interval splitting. IEEE Signal Process. Lett. **13**(2), 69–72 (2006). doi:10.1109/LSP.2005.861589
31. Witten, I.H., Neal, R.M., Cleary, J.G.: Arithmetic coding for data compression. Commun. ACM **30**(6), 520–540 (1987)
32. Wong, K.-W., Lin, Q., Chen, J.: Simultaneous arithmetic coding and encryption using chaotic maps. IEEE Trans. Circuits Syst. **57**, 146–150 (2010). doi:10.1109/TCSII.2010
33. Xie, D., Kuo, C.C.J.: Enhanced multiple huffman table (mht) encryption scheme using key hopping. In: Proceedings of the 2004 International Symposium on Circuits and Systems, ISCAS'04, vol. 5, p. 568. IEEE Press, New York (2004)
34. Xie, D., Kuo, C.C.J.: Multimedia data encryption via random rotation in partitioned bit streams. In: IEEE International Symposium on Circuits and Systems, ISCAS 2005, pp. 5533–5536. IEEE Press, New York (2005)
35. Yu, H.H.: Scalable encryption for multimedia content access control. In: Proceedings of International Conference on Multimedia and Expo, ICME'03, vol. 1, pp. 633–636 (2003). doi:10.1109/ICME.2003.1220997
36. Zambreno, J., Nguyen, D., Choudhary, A.N.: Exploring area/delay tradeoffs in an AES FPGA implementation. In: Proc. IEEE Intl. Conf. Field Programmable Logic and Applications, FPL 2004, pp. 575–585 (2004)
37. Zhou, J., Au, O.C., Fan, X., Wong, P.H.W.: Joint security and performance enhancement for secure arithmetic coding. In: ICIP, pp. 3120–3123 (2008)

38. Zhou, J., Au, O.C., Wong, P.H., Fan, X.: Cryptanalysis of secure arithmetic coding. In: ICASSP, pp. 1769–1772 (2008)
39. Zhou, J., Liang, Z., Chen, Y., Au, O.C.: Security analysis of multimedia encryption schemes based on multiple Huffman table. IEEE Signal Process. Lett. **14**(3), 201–204 (2007). doi:10.1109/LSP.2006.884012

Chapter 8
Conclusion

Abstract This book gave a glimpse to joint approaches to the design of secure embedded multimedia systems.

This book gave a glimpse to joint approaches to the design of secure embedded multimedia systems. We began with a literature review of existing techniques, to understand the need of selective encryption considering the large volumes of multimedia data. Next, we studied the basics behind joint encryption and compression schemes. We studied various examples: Poly-DWT combined architectural optimization with compression, SWT combined compression with encryption using DWT or Discrete Wavelet Transform while CAC combined arithmetic coding based compression with encryption. There are no hard and fast rules on how to approach the joint compression and encryption problem. But a strong background in image coding and signal processing with knowledge of encryption basics is required to propose a new scheme which can ideally perform encryption at no extra computational cost, providing real-time or application specific security and also without compromising on compression performance.

The authors sincerely hope that the readers are inspired to analyze themselves different parts of the compression system and come up with new schemes, experiment with them and test their security and privacy (offered) with the application requirements.

A. Pande, J. Zambreno, *Embedded Multimedia Security Systems*,
DOI 10.1007/978-1-4471-4459-5_8, © Springer-Verlag London 2013

Index

A. Pande, J. Zambreno, *Embedded Multimedia Security Systems*,
DOI 10.1007/978-1-4471-4459-5, © Springer-Verlag London 2013