International Journal of Computer Applications (0975 – 8887)
Volume 59– No.19, December 2012

Encryption of Compressed MultiMedia Data

Mohammad Jafarabad
Ph.D student, Xidian University, xi'an, China

Mahnaz Rafie
Young Researchers Club, Ahvaz Branch, Islamic Azad University, Ahvaz, Iran

Samira Khodkari
Payam Noor University, Dastjerd Branch, Qom, Iran

Zahra Alizadeh
Payam Noor University, Dastjerd Branch, Qom, Iran

ABSTRACT

To send multimedia data over an insecure communicational network with limited bandwidth, we need an organized management for creating and sending information. So far, there have been a few methods proposed from the combination of compression with symmetric encryption, for sending these files. In this paper, using pipeline compression with implementation upon Huffman algorithm, instead of usual compression, is proposed. Moreover, instead of utilizing symmetric encryption algorithms with a low level of security, the public key encryption algorithms are used. The chosen asymmetric encryption algorithm, for implementing some operations on multimedia data, is similar to RSA encryption and uses the a^b mod m expression to generate the key. Moreover, in this paper there has been a circuit proposed with the goal of increasing the speed of the located multiplier in this mathematical expression. Also, considering the existence of the adder in encryption multiplication circuit, and compression circuit, the use of an special adder is recommended for improving the speed of these parallel multimedia computations.

Keywords

Multimedia data, multiplication, compression, asymmetric encryption, coding.

1. INTRODUCTION

In recent century, multimedia systems have made a dramatic progress such as accessing to modern services like Internet TV, Video Conference, Video Telephony, Video Sharing Websites. With the daily increasing number of multimedia systems' users, the security issue becomes even more important than before because as information technology grows and advances in network-based communications, spying software, hackers, and viruses would try harder to access the information. For instance, nowadays in big companies, instead of using cable CCTV cameras they use wireless cameras with the ability to transfer voice and video, simultaneously. Considering the existing security threats during data transfer wirelessly, designing some systems for monitoring the security of the data transfer is necessary.

By 2015, over 60 percent of the stored data throughout the world will be multimedia data. [6] Sending these data using the network is time consuming. Taking into account the security issues, encryption must be applied on the data prior to sending procedure, which will impose an increase in the size of multimedia. A solution for this problem is to use compression. Applying compression (data size decreasing) and encryption (data size increasing) upon the data prior to sending, and implementing decryption and decompression on the receiver side, requires a specific period of time. So far, there have been many algorithms proposed for symmetric encryption, which have been used for encrypting the multimedia data.

[7], [17] On the other hand, since asymmetric encryption algorithms lower the speed, they have been used rarely for multimedia data. Considering the vital role of security and speed together, it is possible to propose some methods to improve public key encryption speed, and pipeline compression operation. The rest of the article will be as follows: in section 2 the basics of implementing compression algorithm on multimedia data is discussed. Section 3 is about an introduction for encryption, especially public-key encryption. Moreover, section 4 is devoted for proposing a method to combine encryption procedures with compression, and related issues to information coding. Also the hardware implementation of the aforementioned method is proposed. In section 5 we have main idea and comparison with previously proposed methods. Finally, section 6 is for the conclusion.

2. FAST DATA COMPRESSION

2.1 pipeline compression

Data compression is the representation of an information source (e.g. a data file, video signal, speech signal, an image) as accurately as possible using the fewest number of bits. [11] The gigantic volume of the data traffic across the network, high price of bandwidth, and large volume of multimedia data are the reasons that the need for compression is even more than before, these days. Most of the multimedia data that need to be stored on a memory or sent over the network, consist of some long sub-strings with similar properties. This means that a series of similar sub-strings are sent, periodically. While sending data in lower volumes, distinguishing these sub-strings is not affordable, however when sending these repetitive sub-strings for a long time is required, we need a method for organizing, and decreasing the size of the data being sent. To transfer a one-hour audio file with 44 K sample/sec and 16-bit stereo, using two channels we must transfer 3600*44000*2*2 bits of data, which is equal to 633.6 MB, size of data which could be reduced up to ten times using the compression algorithms in MP3 format. Additionally, to send a colorful image of 500*500 pixels without compression we need 750 KB which could drastically decrease 10 to 20 times using JPEG format. Moreover, for transferring a one-minute real time video file, full size, and colorful we need 60*30*640*480*3 bits of data, which is equal to 1.659 GB which could reach up to 200 GB for a two-hour movie. MPEG21 employs some compression techniques, and reduces this massive volume to 3.9GB, that is equivalent to one DVD.[11] Nowadays, different algorithms for compressing the information on multiple channels are in use, and in some cases, multiple compression algorithms are used for one type of data, simultaneously. Compression techniques try to reduce the data rate without losing the desired quality, however, sometimes it is highly required to compress the information as much as possible. For example, for a multimedia data, compressing in FLV, MPEG, WMV, MP4, AVI and other formats could be employed. For instance, for compressing a movie we can eliminate many of the middle frames, and using the consecutive frames technique reduce the size of the data. Compression algorithms are categorized in two main groups of lossy and lossless. In lossless algorithms, after decompressing, all of the information could be recovered without any change, and that is why this algorithm is also known as the reversible compression. However, in lossy compression, it is probable to lose a fraction of data while doing decompression procedure, and as a result, an accurate copy of the resource file cannot be obtained. [16] These implemented irreversible changes on the data, are to reduce the size of the information. Digital images, and multimedia files are some of the sources, which use lossy compression technique to decrease the size. The original data cannot be retrieved from the compressed data unless the compression algorithm is known. However, this does not mean that these systems possess a high level of security because using hardware methods the compressed data could easily be decompressed. The foundation of all coding algorithms, including compression and encryption, is based on mathematical calculation.[8] If we could compress a multimedia data for times, as a result, a drastic decrease in the size of the data will be acquired. Figure 1 demonstrates the pipeline compression.

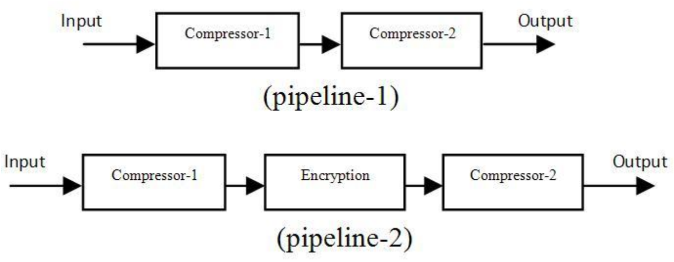

Figure 1. Pipeline compression

Here, we will propose a method for pipeline compression that could be implemented on Huffman compression algorithm. In the following, we will explain one of the most important coding algorithms named Huffman Algorithm, which is used to code the multimedia data for compression.

2.2 pipeline Huffman coding

In 1951, Huffman chose the problem of finding the efficient binary code as a topic of research for one of his courses in MIT University. He succeeded to introduce the idea of ordering based on repetition as an efficient solution for the problem. In his method, Huffman has eliminated the drawback previously seen in semi-optimal Shannon-Fano coding. Instead of building the tree in up-down direction, he built it in down-up direction. [12] Huffman coding is a method to code a sequence of data items with the minimum number of bits necessary, and it is based on the fact that the probability of the symbols in the compressed files has already been obtained. [13] However, if prior to implementing the Hufman algorithm these probabilities were unknown, it is necessary to firstly calculate the occurrence frequency of each of the symbols, and then by drawing the Huffman tree the compression could be accomplished. There have been many papers about hardware implementation of Huffman code published ([12], [5], [13], [10]). In this paper, the proposed method in [11] is used to implement the pipeline algorithm.

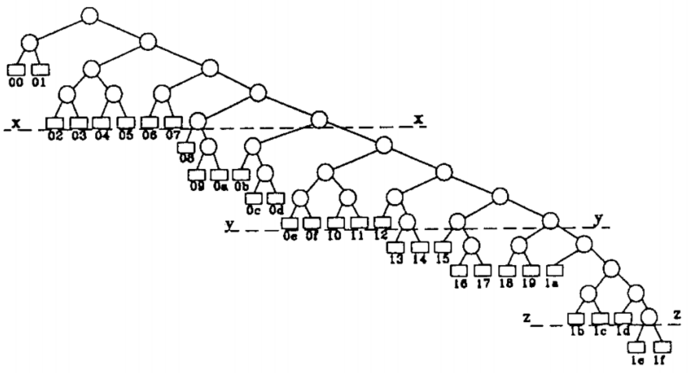

Figure 2. A Huffman tree is partitioned by the cut lines xx, y-y, and z-z. [11]

Figure.2 suggests a method to cut the Huffman tree for executing the pipeline procedure, and eventually demonstrates its implementation in an MPEG video system. For each one of the clusters in Huffman tree a specific table is considered on which the Huffman algorithm will be executed, in a parallel manner.

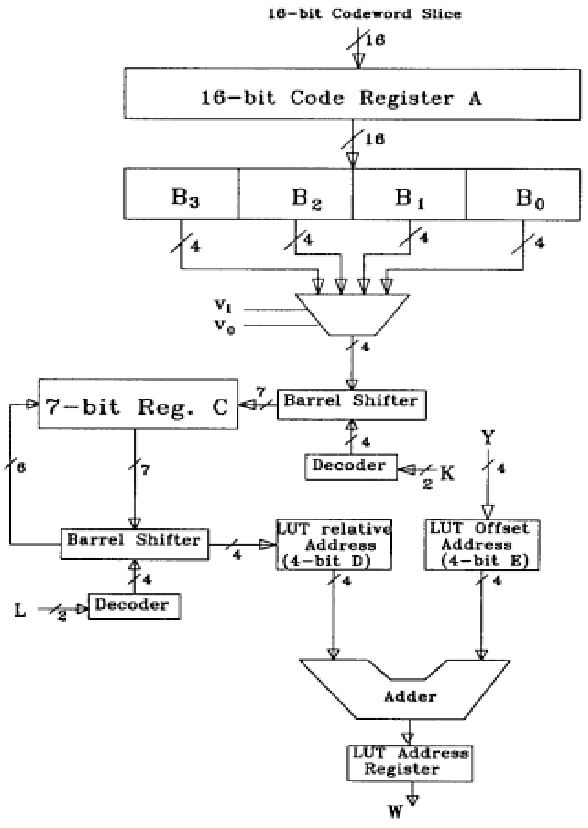

Figure 3. A Huffman circuit [11]

The address-producer block or Look-Up Table (LUT) is the central block of the whole design. This block, depicted in Figure.3, is used to produce the addresses in LUT for the symbols. The input data to the address-producer block is a 16-bit codeword slice which is obtained from the FIFO RAM. This slice will be broken into a group of codewords, and each one these codewords is equal to one Huffman tree with minimum weighted path length from the root[9]. In this example, each codeword is an address, which refers to the next cluster in Huffman tree. The address producing stage initiates with dividing the 16-bit codeword slice into groups with the equal size of L_m, which is the length of the longest branch of the cluster in the tree. In our example, the length of the longest branch of the cluster is equal to four (L_m=4), and as a result, the input data will be divided in four groups. Since all we have is 16-bit data, we will have four groups of 4-bit data. The input data will simultaneously move towards the B_0, B_1, B_2, and B_3 registers. Next step is to use a 4 to 1 multiplexer to transfer one of these registers to the 7-bit register C.

To design LUT table, two Barrel shifters, and two 2x4 Decoders re used. The remaining steps of the hardware implementation of Barrel Shifter until reaching the LUT address register is covered in [12]. As depicted in Figure.3, in the structure of LUT one ADDER is also used to calculate the summation result of LUT relative address and LUT offset address. In section 4, for improving this diagram a method is proposed.

3. SYMMETRIC ENCRYPTION AND PUBLIC KEY CRYPTOGRAPHY

Encryption is the knowledge of studying and recognition of principles and methods of transferring or storing data in a secure manner. Essentially, encryption is about changing the context of a message, using one encryption key, and one encryption algorithm. [14] Here, both the encryption and decryption procedures are handled using the same key. In this method, firstly, sender or receiver generates a secure key, and then they generate a copy of this key and send it to the other end using a secure communication channel. The only drawback of this algorithm is that in the first step a key, which is the most important part of encryption, must be exchanged between the communication parties. In the case that the intruder could access the exchanged key in the first phase of starting communication, they can decrypt all the information that is going to be sent or received after that, easily.

Public key encryption is another method of encryption, which is for increasing the security level. In this method, the receiver generates the private and public keys. After that, it will make a few copies of the public key and sends them to the senders. Moreover, it is possible to use one specific pair of keys for each sender. Sending the public key over the network causes no security issue. After receiving the public key, the sender codes its data and sends it to the receiver. This coded data on sender side is also known as cipher text. The sender generates the cipher text, however, even the sender itself is not able to turn it back to the plain text format. The sender must send the cipher text to the receiver so that the receiver could decrypt it using the private key. The essential point about this method of encryption is that for initiating the data transfer operation, the receiver must generate the key. Stages of production and transfer of keys in asymmetric encryption decrease the speed of encryption, but also increases the security compared to symmetric methods.

4. COMBINATION OF ENCRYPTION AND PIPELINE COMPRESSION OPERATIONS

Establishment of absolute security in sending video files over network requires the simultaneous use of encryption and decryption operations. [4] In the proposed method of this paper, the encryption and compression algorithms will be applied on multimedia data, separately. Taking into account that the encryption and compression happen on one side, and decryption and decompression take place on the other side of communication, there must be an order to perform these operations.

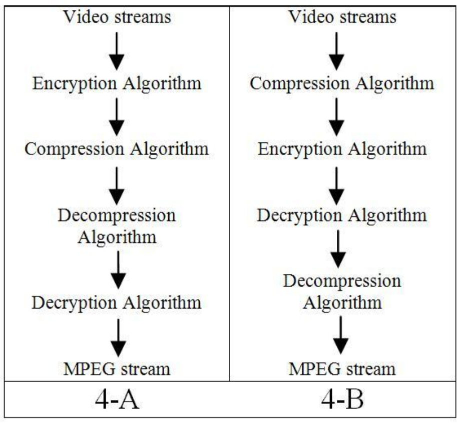

Figure 4. Combination of encryption and compression

For example, in Figure.4-A, first the encryption and then the compression is done by the sender. However, on receiver side, the data is first decompressed but the decryption operation cannot be handled because decompression of multimedia data is lossy which means that some fraction of the sent cipher text is lost, thus decryption is not possible. However, if the consecutive process of encryption and compression is done according to Figure.4-B, then there will be no problem in decryption phase. The reason, as depicted in the Figure 4, is that the encryption operation is implemented on the compressed data, and the same data without any alteration will be decrypted on receiver side. Considering that the encryption operation is lossyless, the compressed data is directly available, without any change, on the output. In the end, the decompression operation can be executed to restore the file. As shown in Figure4.B, a video stream is converted to MPEG stream with decreasing the size and in a highly secure.

4.1 Encryption with MPEG standard

The MPEG file is a set of frames which can be categorized in three major groups, namely: I-frames (Intra frame), B-frames, and P-frames (Predict table frame). Figure.5 shows the order of these frames in a slice layer.

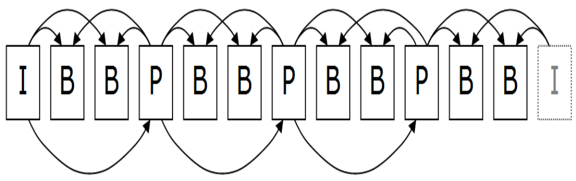

Figure 5. I-frames , B-frames and P-frames

I-frames provide the least amount of compression, however, if in video streams compression only I-frames were used there would be no need to other frames in decoding. The second type of frames is P-frames, in which we utilize the data located in previous frames to establish compression, which results in higher level of compression compared to I-frames. The third type of frames is B-frames, which use the previous and forward frames for compression. The highest level of compression is achieved using B-frames. One of the methods to encrypt video files is to implement the encryption operation on all of the frames. Moreover, this would provide us with the highest security level but due to the increase in calculation complexity, we would suffer from a drastic decrease in speed.

To avoid this speed reduction, a threshold has to be defined between speed and security, which requires selection of a specific group of frames for encryption.

There have been three major solutions proposed for choosing an appropriate set of frames for encryption. [15] First solution was proposed based on a theory that it is not possible to restore B-frames and P-frames without having any information about I-frames. Here, we only encrypt the I-frames which results in 80-85 percent saved processing time. In the second method, all I-macroblocks will be encrypted within I,P, and B frames which brings about 65-75 percent processing time but with lower speed compared to previous algorithm. Finally, we have encryption of I-frames and headers of predicted macroblocks. This algorithm is the most secure and yet the slowest encryption algorithm for video streams with the processing time equal to 25-65 percent.

4.2 Speed and security improvement for encryption and compression on video streams

Encryption and compression operations require high computational time, which leads to a decrease in the speed of system. In the proposed method, public key encryption is used to encrypt the multimedia data. This results in higher security compared to previous methods of symmetric encryption, which were used for multimedia data. As a result, using the public key acquires higher level of security. Additionally, in this method the encryption is implemented only on I-frames because with changing the encryption algorithm, we trust the level of security and do not need to encrypt the I-macroblock. Taking into account previously discussed issues; the main focus in the proposed method is on the speed. The speed of public key algorithms depends on the computational complexity of a^b mode m. According to Figure.3, compression algorithm also needs an adder, which we suggest that adder to be a Wallace tree adder, to improve the speed. Moreover, for improving the speed of public key encryption, we use a fast multiplier.

The proposed multiplier reduces the delay and consumed power in the circuit. Additionally, to improve the speed of the multiplier, we have implemented the multiplier using the Wallace tree adders. Although calculations such as finding prime numbers or calculations in Euler's theorem, Fermat's theory, and Euclid's theory are some part of encryption process, but the highest cost in important cryptography systems such as RSA and Diffie-Hellman is related to calculating the result of a^b mod m. There are three methods to calculate this expression.

The first group of these methods contain some procedures , which improve the speed of exponential operation. As a few examples we can refer to sliding window and M-ary methods. In these methods, the bits are clustered in a special way and by scanning them from left to right the fastest calculation method is achieved. For instance, in [33] by using 4-ary method, M^{506} is obtained only by using 14 multiplication operations.

The second way includs some methods, which utilizes multiplication operations in order to accomplish the exponential operations. It means that they compute a^b as multiplying a by itself b times, and then they use a method to reduce the number of partial products. The Booth algorithm is one of these methods which take advantage of re-coding the multiplicand and the multiplier to decrease the complexity of multiplication computation.

The third group of methods which are more important in this article are some methods that decrease the number of partial products by a new scheme of adders. It means that instead of decreasing the number of partial products we decrease the process of adding those product, which leads to increasing the number of adders in the circuit.

5. THE MAIN PROPOSED IDEA

In previous sections, encryption and related computations were described. Also, some good ways to speed up the coding process were introduced. In this project we plan to use the methods outlined in Part B Section IV, such as implementing the encryption process by using full adder technique. Moreover, Section 2 provides a rapid compression implementation. In the rest of the section, the pipeline Huffman coding was proposed for rapid implementation of the multimedia data compression. In the circuit presented in [11], the Huffman tree was obtained by an adder. So far, the method of implementing the encryption and compression algorithms, using adders, is described. Other methods (Multiplication – exponent) also have been compared until now. In this section we plan to propose one of the appropriate adders to use in these 2 algorithms. Designing of a suitable adder is obtained by utilizing a good VLSI design. An optimal VLSI design should increase the speed and energy-efficiency of the adders, and decrease the required area for each one of them. Among the studied papers in this field, a scheme that was proposed in [18] is for implementing the adders simultaneously in encryption and compression. A 64 bit adder with UMC 2.5 V, 1-Poly, 5-Metal, CMOS technology is designed in [18] which is 20% faster than conventional architecture adder. The reasons for choosing encryption, compression and advantages of each method were discussed in previous sections. Now we introduce the reason of using full adders in multimedia systems. The multimedia processor chips use digital signal processing for arithmetic operations such as adders. High delays of some adders lead to a decrease in video file processing speed. Using the proposed adder in [18], we can increase the process speed for adding operation in multimedia file. Using this type of adder can be the best proposal for the implementation of concurrent operations of file compression and encryption in multimedia data. Also this adder has 8% less delay and 12% more speed compared to Brent – Kung Adder.

6. CONCLUSION

Implementing compression and encryption on multimedia data is a complex issue. Utilizing mathematical computations could be a contributing factor to decreasing the complexity of these calculations. In this paper, we have proposed a method for implementing the Huffman tree using the pipeline procedure for compression. In this method, we cluster the Hauffman tree and implement it by using an adder, and also for encryption we utilize a multiplier to handle the arithmetic operations. In this paper, a multiplier is used to improve the operational speed. Furthermore, we have proposed to use Wallace adder in encryption and compression, simultaneously. Considering the advantages of compression, public key encryption, and optimal multiplication algorithms in [1], [2], and [3], the proposed method will trump the previous techniques. The better operation of the system based on increasing security, and reducing the computational speed are achieved without any change in breaking the mathematical encryption functions, and also with the highest level of compression.

7. REFERENCCES

[1] A.J. Menezes, P.C. Van Oorschot, and S.A. Vanstone, *Handbook of appild cryptography*. The CRC Press series on discrete mathematics and its applications. CRC Press, 2000 N.W. Corporate Blvd., Boca Raton, FL 33431-9868, USA, 1997.

[2] A.K. Jain, "Fundamentals of Digital Image Processing", Prentice-Hall, Inc., A Division of Simon & Schuster Engelwood Cliffs, New Jersey, 1989.

[3] C. Adams and S. Lloyd, *Understanding PKI: Concepts, Standards, and Deployment Concicerations*, 2nd Edition (Addison-Wesley, 2003), 322 pp. [**Korean edition:** INFOBOOK (Person Education), 2003].

[4] C.C. Lu and S.Y. Tseng, "Integrated Design of AES (Advanced Encryption Standard) Encrypter and Decrypter", *in Proceedings of the IEEE International Conference on Application-Specific Systems*, PP. 277-285, July 2002.

[5] R. Hashemian, "Design and Hardware Implementation of a Memory Efficient Huffman Decording", *in Proceedings of the IEEE Transaction on Consumer Electronics*, Vol.40, No.3, PP.345-352, Agust 1994.

[6] G. Engels and S. Sauer, Object–oriented Modeling of Multimedia Applications, Handbook of Software Engineering and Knowledge Engineering, Vol. 2, PP. 21-53, *World Scientific*, Singapore, 2002.

[7] G. Boato, N. Conci, V. Conotter, F.G.B. De Natale, and C. Fontanari, "Multimedia Asymmetric Watermarking and Encryption", *Electronics Letters*, Vol .44, No.9, PP. 601-602, April 2008.

[8] H.D. Lin and D.G .Messerschmitt, "Designing a High –Throughput VLC Decoder PartII-Parallel Decording Methods", *In Proceedings of the IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 2, PP. 197-206, June 1992.

[9] H. Park and V.K. Prasanna, "Area Efficient VLSI Architectures for Huffman Coding", *in Proceedings of the IEEE Transactions on Circuits and Systems, Analog and Digital Signal Processing*, Vol .40, No. 9, PP. 568-575, September 1993.

[10] I. Hifn, The First Book of Compression and Encryption, *hifn whitepaper*, www.hifn.com/docs/a/the-first-book-of-compression-and-encryption.pdf.

[11] J. Kim, J. Kim, and C.M. Kyung, "A Lossless Embedded Compression Algorithm for High Definition Video Coding", *in Proceedings of the 2009 IEEE International Conference on Multimedia and Expo, ICME 2009*, PP. 193-196, New York City, NY, USA, 2009.

[12] L.Y. Liu, j.F. Wang, R.J. Wang, and J.Y. Lee, "Design and Hardware Architectures for Dynamic Huffman Coding", *IEE Proceedings, Computers and Digital Techniques*, Vol. 142, No. 6, PP. 411-418, Novomber 1995.

[13] M. Benes, S.M. Nowick, and A. Wolfe, "A Fast Asynchronous Huffman Decoder for Compressed-Code Embedded Processors", *in Proceedings of the 4th International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC '98)*, PP. 43-56, San Diego, CA, USA, IEEE Computer Society 1998.

[14] M. Hassaballah, M.M. Makky, B. Youssef, and B. Mahdy, "A Fast Frastal Image Compression Method Based Entropy", in *Proceedings of the Electronic Letters on Computer vision and Image Analysis*, Vol. 5, No. 1, PP. 30-40, 2005.

[15] M. Ogata, T. Tsuchiya, T. Kubozono, and K. Ueda, "Dynamic Range Compression based on Illumination Compensation", *in Proceedings of the International Conference on Consumer Electronics, ICCE*, Vol. 47, No. 3, PP. 282-283, 2001.

[16] S. Dikbas and F. Zhai, "Lossless Image Compression using Adjustable Fractional Line-Buffer", *in Proceedings of the Signal Processing: Image Communication*, Vol .25, No. 5, PP. 345-351, January 2010.

[17] X. Yi, C.H. Tan, C.K. Slew, and M.R. Syed, "Fast Encryption for Multimedia", *in Proceedings of the IEEE Transactions on Consumer Electronics*, Vol. 47, No. 1, PP. 101-107, February 2001.

[18] S.X. Guang, M.Z. Gang, and L.F.Chang, "A 64 bit Parallel CMOS Adder for High Performance Processors", *in Proceedings of the 2002 IEEE Asia-Pacific Conference on ASIC*, Vol. 40, No. 3, PP. 205-208, June 2002.