American Journal of Applied Sciences 8 (6): 566-573, 2011
ISSN 1546-9239

Single Core Hardware Module to Implement Partial Encryption of Compressed Image

[1]Mamun Bin Ibne Reaz, [1]Md. Syedul Amin,
[1]Fazida Hanim Hashim and [2]Khandaker Asaduzzaman
[1]Department of Electrical, Electronic and Systems Engineering,
University Kebangsaan Malaysia,
43600, UKM, Bangi, Selangor, Malaysia
[2]Plasma Research Laboratory, Research School
of Physical Sciences and Engineering,
Australian National University, Oliphant Building 60,
Mills Road, Canberra 0200, Australia

Abstract: Problem statement: Real-time secure image and video communication is challenging due to the processing time and computational requirement for encryption and decryption. In order to cope with these concerns, innovative image compression and encryption techniques are required. **Approach:** In this research, we have introduced partial encryption technique on compressed images and implemented the algorithm on Altera FLEX10K FPGA device that allows for efficient hardware implementation. The compression algorithm decomposes images into several different parts. We have used a secured encryption algorithm to encrypt only the crucial parts, which are considerably smaller than the original image, which result in significant reduction in processing time and computational requirement for encryption and decryption. The breadth-first traversal linear lossless quadtree decomposition method is used for the partial compression and RSA is used for the encryption. **Results:** Functional simulations were commenced to verify the functionality of the individual modules and the system on four different images. We have validated the advantage of the proposed approach through comparison, verification and analysis. The design has utilized 2928 units of LC with a system frequency of 13.42MHz. **Conclusion:** In this research, the FPGA prototyping of a partial encryption of compressed images using lossless quadtree compression and RSA encryption has been successfully implemented with minimum logic cells. It is found that the compression process is faster than the decompression process in linear quadtree approach. Moreover, the RSA simulations show that the encryption process is faster than the decryption process for all four images tested.

Key words: Real-time secure image, Data Encryption Standard (DES), encryption algorithm, Field-Programmable Gate Arrays (FPGA), video communication, encryption techniques, partial encryption, quadtree compression

INTRODUCTION

The rapid growth of image and video communication nowadays is powered by ever-faster systems demanding greater speed and security. Real-time secure image and video communication is challenging due to the processing time and computational requirement for encryption and decryption. In order to cope with these concerns, innovative image compression and encryption techniques are required.

Although a vast number of compression and encryption algorithms exist, they have been traditionally developed independently of each other. A partial encryption scheme for images that takes advantage of the image compression algorithm has been proposed by Liu *et al*. (2011); Cheng and Li (1996; 2000) and Cheng (1998). The scheme makes use of a compression algorithm that decomposes an image into several different parts. A secure encryption algorithm is then used to encrypt only the crucial parts, which are considerably smaller than the original image. This will result in significant reduction in processing time and computational requirement for encryption and decryption.

Corresponding Author: M.B.I. Reaz, Department of Electrical, Electronic and Systems Engineering, Universiti Kebangsaan Malaysia, 43600, UKM, Bangi, Selangor, Malaysia Tel: +603-89216311 Fax: +603-89216146

Other researchers have also proposed partial encryption, or combined compression and encryption methods (Liu *et al*., 2011; Ahmed, 2010; Akter *et al*., 2008a; 2008b; Reaz *et al*., 2006a; 2007a; Tho *et al*., 2004). Dang and Chau (2000) has proposed the joint image compression and encryption scheme using Discrete Wavelet Transform (DWT) and Data Encryption Standard (DES). Jakobsson *et al*. (1999) developed a "Scramble All, Encrypt Small" technique that encrypts only a small block of an arbitrarily long message. However, the former is less efficient than a partial encryption scheme and utilizes an encryption algorithm (DES) that is no longer secure. The latter requires an ideal hash function that is hard to realize and may not be suitable for images as it was designed for data encryption. In another work, Prasad and Kurupati (2010) proposed a combination of Arnold scrambling and DWT for secured image compression. But Arnold scrambling alone is not sufficient enough to provide significant security with the implementation of RSA (Wei *et al*., 2009).

Traditionally, image compression and encryption algorithms have been restricted to the software realm and developed separately. Although the advantages of software are ease of update, flexibility and portability, hardware implementation is faster and more physically secure, especially when secret key storage security are concerned.

The Field-Programmable Gate Arrays (FPGA) offers a potential alternative to speed up the hardware realization (Marufuzzaman *et al*., 2010; Reaz *et al*., 2007b). From the perspective of computer-aided design, FPGA comes with the merits of lower cost, higher density and shorter design cycle (Choong *et al*., 2005). It comprises a wide variety of building blocks. Each block consists of programmable look-up table and storage registers, where interconnections among these blocks are programmed through the hardware description language (Reaz *et al*., 2004a; Reaz *et al*., 2003). This programmability and simplicity of FPGA made it favorable for prototyping digital system. FPGA allows the users to easily and inexpensively realize their own logic networks in hardware. FPGA also allows modifying the algorithm easily and the design time frame for the hardware becomes shorter by using FPGA (Choong *et al*., 2006; Ibrahimy *et al*., 2006).

This study aims to investigate the hardware feasibility and performance of a novel partial encryption scheme for compressed images using FPGA by means of using a standard hardware description language VHDL. The use of VHDL for modeling is especially appealing since it provides a formal description of the system and allows the use of specific description styles to cover the different abstraction levels (architectural, register transfer and logic level) employed in the design (Pang *et al*., 2006; Reaz *et al*., 2006b). In the computation of method, the problem is first divided into small pieces; each can be seen as a submodule in VHDL. Following the software verification of each submodule, the synthesis is then activated. It performs the translations of hardware description language code into an equivalent netlist of digital cells. The synthesis helps integrate the design work and provides a higher feasibility to explore a far wider range of architectural alternative (Reaz *et al*., 2005; 2004b).

The FPGA implementation combines compression and a secure encryption algorithm that encrypts only crucial parts of the compressed image. The algorithms chosen for implementation are the lossless quadtree compression and the RSA algorithm. The hardware implementation was done using Altera FLEX10KE device.

MATERIALS AND METHODS

The partial encryption scheme depends on a compression algorithm that decomposes the input image into a number of different logical parts. The output consists of parts that provide significant amount of information about the original image, referred to as the important parts. The remaining parts have little meaning without the important parts, hence known as the unimportant parts. In this partial encryption approach, only the important part needs to be encrypted by a secure encryption algorithm. When the important part is considerably smaller than the total output of the compression, the encryption and decryption time can be reduced significantly.

Quadtree compression: The quadtree decomposition method converts an image into a quadtree structure with intensity values attached to the leaf nodes of the tree. The quadtree structure reveals the outline of objects in the original image (Cheng and Li, 2000). Since the quadtree indicates the location and size of each homogeneous block in the image while the intensity values do not reveal much information, partial encryption is possible by encrypting only the quadtree structure. Here, the quadtree structure is the important part whereas the intensity values form the unimportant part. In the case of lossless compression on a b-bit image, the total size of the leaf values is b(3k +1) bits, where k is the number of internal nodes, which is equivalent to multiplying the size of each leaf value with the number of leaf nodes in the quadtree. An approximate upper bound on the relative quadtree size, which is the ratio of the size of the quadtree and the total size of the compressed image, is given in Eq. 1:

$$\frac{4k+1}{4k+1+b(3k+1)} = \frac{4+\frac{1}{k}}{3b+4+\frac{b+1}{k}} \approx \frac{4}{3b+4} \quad (1)$$

Where size of quadtree = number of nodes = 4k +1 size of compressed image = size of quadtree + size of leaf values = 4k +1+ b(3k +1).

For 8-bit images, b = 8, the size of the quadtree relative to the lossless quadtree compression output is at most 14.3%. The approximation is valid for large value of k, which is typically at least 1000 for 256×256 images and greater for larger images. For lossy compression, this calculation is not applicable because variable number of bits is used to represent leaf values. Results collected from experiments performed by Cheng (1998) on test images show that for typical images, the relative quadtree size is between 13 and 27%. Therefore, only 13-27% of the output of lossy quadtree algorithm is encrypted for typical images.

The lossless quadtree compression algorithm with Leaf ordering II has been used in this research, as it is computationally simpler and secure.

Linear lossless quadtree: Representing quadtree in a tree structure requires the use of pointers. However, the amount of space required for pointers from a node to its children is not trivial. Samet (1985) suggested that each node in a quadtree is stored as a record containing six fields. The first five fields contain pointers to the node's parent and its four children labeled as NW, NE, SW and SE; whereas the sixth field describes the intensity value (color) of the image block that the node represents. The pointers would occupy nearly 90% of the memory space required to store the quadtree (Dang and Chau, 2000). As a result, several pointerless quadtree representations have been proposed by researchers such as Lin (1996) and Gargantini (1982).

This research is based on the breadth-first traversal of linear quadtree proposed by Chan and Chang (2001) and Chang *et al*. (2008). It consists of two lists, i.e., a tree list and a color list. The tree list stores the quadtree structure, where '0' denotes a leaf node and '1' denotes an internal node. The color list simply stores the pixel values of the image in a sequence defined by the tree structure.

RSA Encryption: Since the encrypted part of the proposed partial encryption scheme is preferably small, public key algorithms has been applied directly to it.

In RSA a plaintext block M is encrypted to a cipher-text block *C* by:

$$C = M^e \bmod n \quad (2)$$

And the plaintext block is recovered by:

$$M = C^d \bmod n \quad (3)$$

RSA encryption and decryption are mutual inverses and commutative, due to symmetry in modular arithmetic. Also, (2-3) show that both encryption and decryption are based on the same operation, which is modular exponentiation. Therefore, hardware implementation of RSA allows the encryption and decryption to share the same architecture, which helps reduce the hardware size.

VHDL modeling: The VHDL model for the proposed work consists of four sub-modules. The overall implementation is known as the PARTIAL_ENCRYPT chip and it consists of the functional sub-modules Compression, Encryption, Decryption and Decompression.

Linear quadtree compression/decompression: Linear quadtree compression and decompression are implemented in two separate blocks, QT_ENCODER and QT_DECODER respectively. The combination of these two functional blocks is named QT_CODEC. The linear quadtree codec connects both QT_ENCODER and QT_DECODER in parallel and to the memory block RAM256X8. There are four input control signals, i.e., CLK, RESET, GO and E_D. The architectures of QT_CODEC, QT_ENCODER and QT_DECODER are implemented using Moore state machines with asynchronous reset. The reset signal (RESET) is used to set the state machine to its initial idle state, while a high GO signal switches it from idle state to the next state. A low E_D signal activates the QT_ENCODER while a high E_D activates the QT_DECODER. The READY signal is high when the compression operation is completed.

For compression, the input image is scanned in an order, where each quadrant is scanned in the NW, NE, SW and SE directions. The input image is stored in the RAM from addresses 00 to 3F (hex) in raster scan order, i.e., from left to right and from top to bottom. For a pixel in an 8×8 image indexed by row I and column J where I, J = 0, 1, 2, …7, its corresponding address in the RAM is expressed by:

$$\text{Address} = (8\times I) + J \quad (4)$$

For 8×8 input images, the sequence of RAM addresses in the appropriate scan order is:

$$\text{Address} = 32K_5 + 4K_4 + 16K_3 + 2K_2 + 8K_1 + K_0 \quad (5)$$

Where, K5, K4, K3, K2, K1 and K0 are the individual bit (0 or 1) values of a 6-bit counter that counts from 0 to $(111111)_2$.

The output of the compression is a tree list that describes the quadtree structure ('0' for leave node and '1' for internal node) and a color list that contains the intensity values of the quadtree. On the other hand, the linear quadtree decompression performed by the QT_DECODER block is just the reverse process of the compression. In linear quadtree compression, if a 2× 2 block of an image is homogeneous, it is reduced to one block containing the pixel value; otherwise it is reduced to an 'I' block. The intensity values in an 'I' block are stored in a list. This continues recursively until the 8×8 image is reduced to only one block. The tree list and color list are stored at RAM addresses beginning with 40(hex) and 80(hex) respectively.

RSA encryption implementation: RSA Module consists of 3 sub-modules. They are RSA_LOAD, RSA_CORE and RSA_OUTPUT. RSA_CORE performs encryption and decryption, RSA_LOAD serially captures incoming message to be encrypted or decrypted and RSA_OUTPUT serially outputs the decrypted/encrypted message. A RAM with 2048 bits in size was design to provide the storage element for the RSA encryption modules.

Arithmetic Logic Unit accepts 32 bits data as input and produce 32 bits output. The input data is stored temporary in a larger register (34-bits). Arithmetic operations are performed on the temporary register. The working result is then moved to the output port when the operation is done. The design uses four large registers (34 bits) to hold the working results and 2 small registers (5 bits) to hold the loop variables (i, j). The extra 2 bits in the 4 registers are used in order to prevent overflowing during addition operations.

After consideration on the trade-off between security and speed, the size of parameters and signals of the RSA_CORE module for the VHDL model are chosen as follows:

- M is the 32-bit plaintext for encryption, or the 32-bit cipher-text for decryption
- E and N_C are the 32-bit public key (e, n) used for encryption, or the 32-bit private key (d, n) used for decryption
- CLK is the clock input signal
- RST sets the state machine implemented in RSA_CORE architecture to the initial idle state
- GO switches the state machine from idle state to the next state
- C is the 32-bit cipher-text produced by encryption, or the 32-bit plaintext recovered by decryption
- DONE is high when the encryption or decryption operation is completed, otherwise it is always low

Top level design: The overall design incorporates the RSA_CORE module into the linear quadtree codec. The top level entity is named as PARTIAL_ENCRYPT where a low E_D signal activates the QT_ENCODER block to perform linear quadtree compression on the input image stored in the RAM256X8 block. When compression is completed, the RSA_CORE is activated to encrypt the tree list stored at RAM addresses 80 to 82 (hex). The encrypted tree list is then stored at addresses 88 to 8B. On the other hand, a high E_D signal starts the decryption operation of RSA_CORE on the encrypted tree list to recover the tree list. Then decompression is performed by the QT_DECODER to reconstruct the original image.

Four test images are used as inputs to verify the correctness of the design using functional simulation. All of the test images are grayscale with dimensions 8×8. For clarity, each image is arranged in a 8×8 table, in which the cells correspond to the pixel intensity values (grayscale level or color). The size of each pixel is 8 bits and its value is expressed in 2 hexadecimal digits.

RESULTS

Theoretical results for test image 1 and its quadtree: The output of linear lossless quadtree compression is a tree list that contains the quadtree nodes and a color list that contains the pixel values of the image. In the tree list, binary '0' denotes a leaf node and '1' denotes an internal node. The results of linear lossless quadtree compression are:

Tree list = 1001110000012 = 9C1
Color list = 00 FF 00 FF 00 00 FF 00 00 FF FF 00 FF 00 FF FF FF 00 00
Size of image = 64×8 bites = 512 bits
Size of tree list = 12 bits
Size of color list = 152 bits
Compression ratio = Size of image / (Size of tree list + Size of color list):

$$= \frac{512}{12+152} = 3.12:1$$

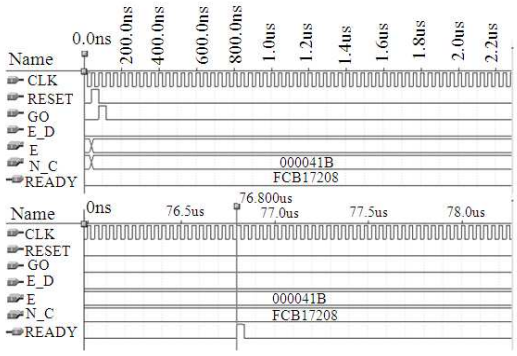

Fig. 1: Simulation of Test Image 1 compression and partial encryption

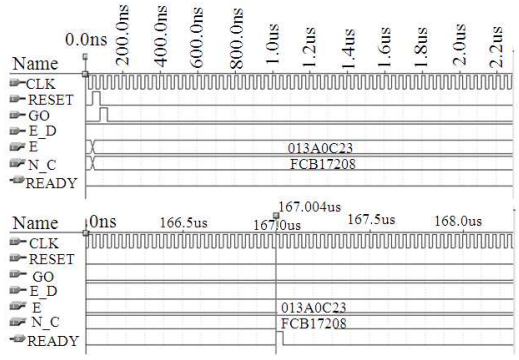

Fig. 2: Simulation of Test Image 1 partial decryption and decompression

Table 1: Comparison of functional simulation

Image	Compression Ratio	Compression (clock cycles)	Decompression (clock cycles)
Test Image	1 3.12:1	291	291
Test Image	2 1.91:1	350	332
Test Image	3 56.89:1	219	200
Test Image	4 0.96:1	489	350

Functional simulation for linear quadtree compression/decompression: Functional simulation is performed to test the logic function of the hardware design and it is presented to verify the correctness of the algorithms implemented by the quadtree codec and the partial encryption module. Table 1 shows the functional simulation results on four test images.

Partial encryption simulation: Functional simulation of the partial encryption module (PARTIAL_ENCRYPT) is performed on four-test image with 40ns simulation clock period (25 MHz). In this study the simulation for test image 1 is shown in Figs. 1-2 using following key pairs:

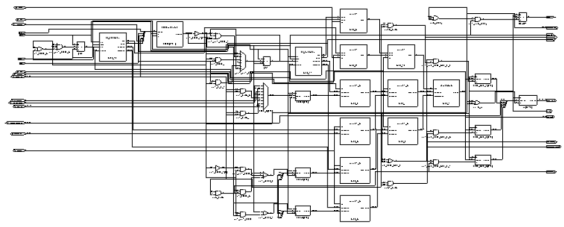

Fig. 3: RTL view of the PARTIAL_ENCRYPT (1 of 3 sheets)

- Public key for encryption, E = 0000041B16, N_C = FCB1720816
- Private key for decryption, E = 013A0C2316, N_C = FCB1720816

Synthesis: In regard to the designated hardware realization, The VHDL code is synthesized by considering Altera FLEX10K: EPF10K10LC84 FPGA chip on LC84 ackage. The FLEX 10K family provides the density, speed and features to integrate entire systems, including multiple 32-bit buses into a single chip. The RTL view for the output layer is shown in Fig. 3.

DISCUSSION

During the formulation of theoretical results for test image 1, we have omitted the root node and bottommost leaf nodes in the tree list in order to achieve better compression ratio, as the decompression algorithm does not need them. Since decompression is simply the reverse process of compression, its results can be deduced from those of the compression.

Functional simulation of the linear quadtree codec (QT_CODEC) is performed on the four test images with 20ns simulation clock period (50 MHz). The time interval between high GO signal and high READY signal is divided by the simulation clock period to calculate the processing time for compression or decompression. Form the results of functional simulation for linear quadtree compression and decompression as shown in Table 1, it is observed that the processing time is longer with smaller compression ratio and decompression is faster than compression.

In the functional simulation for partial encryption, the time interval between high GO signal and high READY signal is divided by the simulation clock period to calculate the processing time for combined compression and partial encryption or partial decryption and decompression and the results are compared in Table 2. It is concluded that the partial decryption and decompression is much slower because the decryption time of the RSA_CORE module is twice longer than the encryption time.

Table 2: Comparison of the combined processing time for test image 1 (functional simulation)

Image	Compression ratio	Compression and partial encryption (clock cycles)	Partial decryption and decompression (clock cycles)
Test Image	1 3.12:1	1918	4173

Throughout the synthesis results, there are a few points worth to be discussed. Firstly, from the synthesis results, the RSA Core module utilized around 20% of the chosen FLEX 10KE device. Nevertheless, the clock frequency report showed the critical frequency is only 34.7MHz. This has given the limitation of the frequency of the RSA Module, even though the serial to parallel and parallel to serial converters could achieve 133.9MHz and 89.6MHz respectively. For the RSA Core module, though the 34.7MHz is acceptable, it is not fast enough compare to today's FPGA technology. However, the critical frequency can possibly be increased further by optimizing the circuit through place and route the internal probes. The synthesis of the whole RSA encryption, which included the RAM implementation, has taken up 554 units of logic cell (LC). This is about 35% utilization of the chosen device. Lastly the top level design, which is the PARTIAL ENCRYPT entity, was synthesized. A total of 2928 units of LC were used and it is about 58% utilization of the device (Altera EPF10K100EQC208-1). The frequency achieved was 13.42 MHz.

CONCLUSION

In this research project, the FPGA prototyping of a partial encryption of compressed images algorithm that allows for efficient hardware implementation had been implemented. The lossless quadtree compression and RSA encryption algorithms are chosen for implementation due to their computational simplicity in hardware. It is found from the simulation results that in linear quadtree approach the compression process is faster than the decompression process. Moreover, the RSA simulations show that the encryption process is faster than the decryption process for all four images tested.

REFERENCES

Ahmed, C.S., 2010. Stego encrypted message in any language for network communication using quadratic method. J. Comput. Sci., 6: 320-322. DOI: 10.3844/jcssp.2010.320.322

Akter, M., M.B.I. Reaz, F. Mohd-Yasin and F. Choong, 2008a. A modified-set partitioning in hierarchical trees algorithm for real-time image compression. J. Commun. Technol. Elect., 53: 642-650. DOI: 10.1134/S1064226908060065

Akter, M., M.B.I. Reaz, F. Mohd-Yasin and F. Choong, 2008b. Hardware implementations of image compressor for mobile communications. J. Commun. Technol. Elect., 53: 899-910. DOI: 10.1134/S106422690808007X

Chan, Y.K. and C.C. Chang, 2001. Concealing a secret image using the breadth first traversal linear quadtree structure. Proceedings of the 3rd International Symposium on Cooperative Database Systems for Advanced Applications (codas), Apr. 23-24, Beijing, China, pp: 194-199. DOI: 10.1109/CODAS.2001.945167

Chang, C.C., Li C.F. and Y.C. Hu, 2008. Code transformation algorithms for two breadth-first linear quadtrees. Proceedings of the International Workshop on Education Technology and Training, Dec. 21-22, Shanghai, China, pp: 799-802. DOI: 10.1109/ETTandGRS.2008.233

Cheng, H. and X. Li, 1996. On the application of image decomposition to image compression and encryption. Communications and multimedia security II: Proceedings of the IFIP TC6/TC11 International Conference on Communications and Multimedia Security, Sep. 23-24, Essen, Germany, pp: 116-127. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.29.569&rep=rep1&type=pdf

Cheng, H. and X. Li, 2000. Partial encryption of compressed images and videos. IEEE Tran. Signal Process., 48: 2439-2451. DOI: 10.1109/78.852023

Cheng, H., 1998. Partial encryption for image and video communication. M.S. Thesis, University of Alberta, Edmonton, Canada. http://portal.acm.org/citation.cfm?id=336117

Choong, F., M. B. I. Reaz and F. Mohd-Yasin, 2005. Power quality disturbance detection using artificial intelligence: A hardware approach. Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05), Workshop, Apr. 4-8, Denver, USA., pp: 146a. DOI: 10.1109/IPDPS.2005.348

Choong, F., M.B.I. Reaz, T.C. Chin and F. Mohd-Yasin, 2006. Design and implementation of a data compression scheme: A partial matching approach. Proceedings of the Computer Graphics, Imaging and Visualisation: Techniques and Applications (CGIV'06), Jul. 26-28, Sydney, Australia, pp: 150-155. DOI: 10.1109/CGIV.2006.94

Dang, P.P. and P.M. Chau, 2000. Image encryption for secure internet multimedia applications. IEEE Trans. Consumer Elect., 46: 395-403. DOI:10.1109/30.883383

Gargantini, I., 1982. An effective way to represent quadtrees. Commun. ACM., 25: 905-910. DOI: 10.1145/358728.358741

Ibrahimy, M.I., M.B.I. Reaz, M.A. Mohd Ali, T.H. Khoon and A.F. Ismail, 2006. Hardware realization of an efficient fetal QRS complex detection algorithm. WSEAS Trans. Circuits Syst., 5: 575-581. http://direct.bl.uk/bld/PlaceOrder.do?UIN=188369709&ETOC=RN&from=searchengine

Jakobsson, M., Stern, J.P. and M. Yung, 1999. Scramble all, encrypt small. Fast Software Encryption Lecture Notes Comput. Sci., 1636: 95-111. DOI: 10.1007/3-540-48519-8_8

Lin, T.W., 1996. Image compression using fixed length quadtree coding. Proceedings of the 3rd International Conference on Signal Processing (ICSP), Oct. 14-18, Beijing, China, pp: 970-973. DOI: 10.1109/ICSIGP.1996.566252

Liu, Y.Z., Y. Xiao and G. Sun, 2011. Encryption algorithm of RSH (round sheep hash) chaoqun. Inform. Technol. J., 10: 686-690.

Marufuzzaman, M., M. B. I. Reaz, M. S. Rahman, M.A. M. Ali, 2010. Hardware prototyping of an intelligent current dq PI controller for FOC PMSM drive. Proceedings of the 6th International Conference on Electrical and Computer Engineering (ICECE 2010), Dec. 18-20, Dhaka, Bangladesh, pp: 86-88. DOI: 10.1109/ICELCE.2010.5700559

Pang, W.L., M.B.I. Reaz, M.I. Ibrahimy, L.C. Low and F. Mohd-Yasin *et al*., 2006. Handwritten character recognition using fuzzy wavelet: A VHDL approach. WSEAS Trans. Syst., 5: 1641-1647. http://www.worldses.org/journals/systems/systems-july2006.doc

Prasad, V.V.R. and R. Kurupati, 2010. Secure image watermarking in frequency domain using Arnold scrambling and filtering. Adv. Comput. Sci. Technol., 3: 236-244. http://www.ripublication.com/acstv3/acstv3n2_15.pdf

Reaz, M. B. I., F. Choong, M. S. Sulaiman and F. Mohd-Yasin, 2007a. Prototyping of wavelet transform, artificial neural network and fuzzy logic for power quality disturbance classifier. J. Elect. Power Components Syst., 35: 1-17. DOI: 10.1080/15325000600815431

Reaz, M. B.I., Islam, M.T., Sulaiman, M.S., Ali and M.A.M. Sarwar *et al*., 2003. FPGA realization of multipurpose FIR filter. Proceedings of the Parallel and Distributed Computing, Applications and Technologies (PDCAT), Aug. 27-29, Chengdu, China, pp: 912-915. DOI: 10.1109/PDCAT.2003.1236448

Reaz, M. B.I., M.I. Ibrahimy, F. Mohd-Yasin, C. S. Wei and M. Kamada, 2007b. Single core hardware module to implement encryption in TECB mode. Inform. MIDEM, J. Microelect., Elect. Components Mater., 37: 165-171. http://www.midem-drustvo.si/VOL%2037%282007%293.pdf

Reaz, M. B.I., Sulaiman, M.S., Yasin, F.M. and T.A Leng, 2004a. IRIS recognition using neural network based on VHDL prototyping. Proceedings of the 2004 International Conference on Information and Communication Technologies: From Theory to Applications, ICTTA 2004, Apr. 19-23, Damascus, Syria, pp: 463-464. DOI: 10.1109/ICTTA.2004.1307832

Reaz, M.B.I., F. Mohd-Yasin, M.S. Sulaiman, K.T. Tho and K.H. Yeow, 2004b. Hardware prototyping of boolean function classification schemes for loss-less data compression. Proceedings of the Second IEEE International Conference on Computational Cybernetics (ICCC 2004), Aug-Sep. 30-01, Vienna, Austria, pp: 47-51. DOI: 10.1109/ICCCYB.2004.1437664

Reaz, M.B.I., F. Choong and F. Mohd-Yasin, 2006a. VHDL modeling for classification of power quality disturbance employing wavelet transform, artificial neural network and fuzzy logic. Simulation: Trans. Soc. Model. Simulation Int., 82: 867-88. DOI: 10.1177/0037549707077782

Reaz, M.B.I., M. Akter and F. Mohd-Yasin, 2006b. Image compression system for mobile communication: Advancement in the recent years. J. Circuits, Syst. Comput., 15: 777-815. DOI: 10.1142/S0218126606003301

Reaz, M.B.I., P.W. Leong, F. Mohd-Yasin and T.C. Chin, 2005. Modeling of data compression using partial matching: A VHDL approach. Proceedings of the 6th World Wireless Congress (WWC), May 25-27, Palo Alto, USA., pp: 411-416. http://delson.org/wwc05/study.htm or http://www.scopus.com/record/display.url?src=sandorigin=ctoandctoId=CTODS_161278959andstateKey=CTOF_161 278960andeid=2-s2.0-20444443530

Samet, H., 1985. Data structures for quadtree approximation and compression. Commun. ACM., 28: 973-993. DOI: 10.1145/4284.4290

Tho, K.T., K. H. Yeow, F. Mohd-Yasin, M. S. Sulaiman and M. I. Reaz, 2004. VHDL modeling of Boolean function classification schemes for lossless data compression. WSEAS Trans. Comput., 3: 365-368. http://www.wseas.us/e-library/conferences/tenerife2003/studys/462-234.pdf

Wei, Y., Hao Y. and Y. Li, 2009. A multipurpose digital watermarking algorithm of color image. Proceedings of the 2009 IEEE International Conference on Mechatronics and Automation, Aug. 9-12, Changchun, China, pp: 112-117. DOI: 10.1109/ICMA.2009.5246380