J Sign Process Syst (2015) 78:131–137
DOI 10.1007/s11265-013-0834-3

2-D DWT System Architecture for Image Compression

Boon Hui Ang · Usman Ullah Sheikh · Muhammad Nadzir Marsono

Received: 14 August 2012 / Revised: 14 July 2013 / Accepted: 15 July 2013 / Published online: 2 August 2013

Abstract Wavelet transform has contributed significantly in multiple areas such as image processing, compression, signal analysis, and medical imaging. Discrete wavelet transform (DWT) requires very large memory requirement and is computationally intensive, especially for 2-D transform that has a quadratic computational complexity. In this paper, we propose a dedicated processor for 2-D DWT computation. The DWT system architecture is parameterizable, where its performance can be scaled by increasing or reducing the DWT engines, according to different application needs. This architecture requires significantly less computational resources and internal memory. The proposed architecture can achieve a theoretical throughput of 138 frames per second for a 2048 × 1536 video processing. The DWT system has been designed for scalability to support up to 8 parallel DWT engines.

Keywords 2-D discrete wavelet transform · Fast lifting · FPGA · Parallel architecture

1 Introduction

There are several DWT VLSI implementations, with the simplest is the one-dimensional DWT filter bank [11]. The filter bank is typically implemented using systolic architecture or parallel architecture, where the input data are fed through a chain of filters that perform wavelet transform. A two-dimensional DWT (2-D DWT) can be implemented by cascading systolic 1-D filters in a parallel array. The 1-D and 2-D DWT implemented in hardware filters such as [3, 11] are the direct form of DWT, i.e., convolution based wavelet transform. These architectures require a large number of multipliers and adders, resulting in large chip area. Other architectures such as [6, 9] are based on FPGA-DSP systems. Although reprogrammable, the system board area is larger due to use of multiple components.

Another way to reduce DWT computation complexity is the Fast Lifting Discrete Wavelet Transform (FL-DWT) [8]. The FL-DWT factors the original DWT into smaller and simpler filtering steps. The number of multiplications has been reduced significantly. Additionally, the coefficients used in FL-DWT can be integers instead of floating points, without experiencing precision loss. In this paper, we propose a dedicated processor for 2-D DWT to achieve improvements on throughput, scalability and flexibility compared to prior architectures, particularly the FL-DWT. The proposed architecture can achieve theoretical throughput of 138 frame per second (FPS) for a 2048 × 1536 (QXGA) video processing. The DWT system can support up to 8 parallel DWT engines, with each DWT engine capable to work independently.

2 The Fast Lifting Discrete Wavelet Transform

The main criteria aimed for the 2-D DWT architectures are chip area and speed performance. The chip area is typically determined by the computational block such as multiply-and-accumulate (MAC), memory bits and registers, and chip area for wire routing. The speed is affected by the input/output rate (number of bits processed per cycle) and the clock rate (cycles per second). The product of these two parameters is the throughput. DWT architectures feed the

B. H. Ang · U. U. Sheikh · M. N. Marsono (✉)
Faculty of Electrical Engineering, Universiti Teknologi Malaysia, 81310 Johor Bahru, Malaysia
e-mail: nadzir@fke.utm.my

input data through a chain of filters that perform DWT. Most are the direct form of DWT, i.e., convolution based, which are inefficient – requiring many adders and multipliers, and large memory storage. A good design should balance the external memory access bandwidth and internal buffer size.

Sweldens introduced the FL-DWT in 1996 [10], which is a fast and low computation form of the pyramid algorithm [7]. The lifting algorithm has three important steps: *split*, *predict* and *update*. In the *split* step, the input samples are divided into two separate set of samples, odd samples and even samples. In the *predict* step, even samples are used to forecast the odd samples based on the relationship that exists in the signal. The differences between the odd samples and the estimated values are calculated and labelled as high-pass or detailed coefficients. In the (*update*) step, the low-pass coefficients are computed by updating the even samples with high-pass coefficients. The lifting based DWT is shown in Fig. 1.

A systolic-parallel approach was proposed by Vishwanath et al. based on recursive pyramid algorithm [11]. The algorithm schedules each output at the earliest possible instance. The i-th level decomposition precedes the $(i+1)$-th level decomposition while guaranteeing that all data needed for the next level decomposition is available at the right time. The systolic architecture where a low-pass filter and a high-pass filter produce an output at every cycle [2]. Each filter unit consists of L MAC units (where L is number of filter coefficients), and the latency for each cycle is LT_m where T_m is the delay of each MAC. The systolic-parallel architecture requires huge amount of computation elements. It requires $2L$ MAC units, $2L$ multipliers and $2(L-1)$ adders. The latency of is relatively high and the throughput of 1 output per cycle. Another downside is that is applies direct form of convolution based DWT requiring floating point computation units.

The direct form of convolution based DWT requires a memory storage, S required for the holding cells, is given in Eq. 1:

$$S = LN\left(1+\frac{1}{2}+\frac{1}{4}+\ldots+\frac{1}{2^{J-1}}\right) = LN\sum_{j=1}^{J}\frac{1}{2^{j-1}} \quad (1)$$

where L is the filter length and N is the longest dimension of the 2-D data, and J is the decomposition levels. The

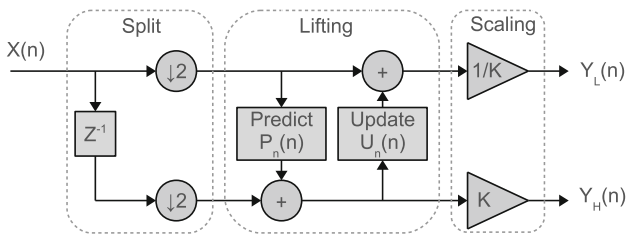

Figure 1 Lifting-based forward DWT [8].

maximum number of holding cells is $2LN$, which translates to $2 \times 9 \times 1024 = 18k$ words for 9/7-F filter, which is tremendously large for ASIC implementation.

Chakrabarti et al. presented a parallel filter architecture [4], which is based on recursive pyramid algorithm consisting of two parallel filter arrays and a storage unit [2]. Each array consists of L^2 multipliers and L^2-1 adders. This architecture is able to operate at very high throughput, where L^2 outputs are produced each cycle, however at the expense of resource utilization, as it requires huge amount of computation elements. For a 9/7-F filter implementation, it requires $2L^2 = 2 \times 9^2 = 162$ multipliers and $2(L^2-1) = 2(9^2-1) = 160$ adders. The storage size is $2LN$ or 18 k words for a 1024×1024 image using 9/7-F filter.

There are various memory based DWT architectures that are structurally similar and are mostly lifting based DWT. First, the 1-D DWT computes the transform along the rows and then computes the transform along the columns. Mansouri in 2009 [8] proposed a memory-based 2-D DWT architecture which was developed based on the architecture in [1]. The intermediate coefficients generated by the row processor are stored in internal buffers, before being processed by the column processor. The LL sub-band coefficients generated by the column processor are stored in an intermediate FIFO prior to the next level of decomposition. The internal buffer and FIFO are needed to minimize external memory cycles. However, the internal buffer would require significant amount of chip area and is a critical issue in 2-D DWT implementations.

There are other memory based architectures e.g., [5, 12] that can be categorized to three categories: *level-by-level*, *block-based* and *line-based*. A typical *level-by-level* architecture uses single DWT engine that processes the rows and followed by the columns. The intermediate results are stored in internal buffers and thus, the memory usage is huge. In the *block-based* architecture, the image is divided into smaller blocks that can be squeezed in smaller internal memory. A typical block based architecture reads the external memory block-by-block and computes the DWT coefficients in a block-by-block manner. *Line-based* architectures are more memory-efficient as it only stores several lines of an image and the memory requirement is a function of the image width or height. The architecture proposed by Mansouri [8] also employs the line-based architecture. For a 1024×1024 image, the proposed architecture only needs 8 adders, 4 multipliers and $2N = 2 \times 1024 = 2$k words of internal buffer for 9/7-F filter implementation. It has clear advantage over other architectures due to lower resource utilization, memory usage and no floating point units. This paper is aimed to improve the throughput and scalability using parallel processing.

3 Proposed 2-D DWT Architecture

The 9/7-F lossy JPEG image compression for forward and inverse integer wavelet filters are first modelled in C++. This is to understand performance related issues for the proposed 2-D DWT architecture and the required resources for processing elements.

3.1 9/7-F Forward and Inverse Filter

The 9/7-F filter lifting steps can be described by Eqs. (2) to (5) with an assumption of scaling factor $K = 1$. The values of lifting coefficients are $a = -203/128$, $b = -217/4096$, $c = 113/128$ and $d = 1817/4096$.

$$y'[2n+1] = x[2n+1] + a(x[2n] + x[2n+2]) \quad (2)$$

$$y'[2n] = x[2n] + b(y'[2n+1] + y'[2n-1]) \quad (3)$$

$$y[2n+1] = y'[2n+1] + c(y'[2n] + y'[2n+2]) \quad (4)$$

$$y[2n] = y'[2n] + d(y[2n+1] + y[2n-1]) \quad (5)$$

To completely eliminate the need of floating point multiplication and integer division, which typically takes more cycles to compute, we approximate the floating point coefficients with rational numbers. The rational number consists of an integer numerator and an integer denominator which can be written as 2^n, where n is an integer, in which division can be done by arithmetic shift right operation. From the software model, it is very clear that a total of 8 additions, 4 multiplications and 4 right shifts are needed for every output computation. This is consistent with the architecture proposed by Mansouri [8].

The 9/7-F inverse filter is an opposite form of forward DWT. The inverse DWT can be obtained by alternating the signal flow to the opposite direction and inverting the filter coefficients. The inverse transform also requires the same number of adders and multipliers.

3.2 Architecture Specification

The 2-D DWT processor is designed to process $N \times M$ pixels images, running at F frames per second (FPS), and with J decomposition levels. The image is compressed by $4\times$ for each decomposition level. The system throughput requirement in filtering operation per second (FOPS) is given as $\frac{8}{3}FNM(1 - \frac{1}{4^J})$ and its upper limit for very large J to be $\approx \frac{8}{3}FNM$.

We propose a 2-D DWT architecture to process 1024 × 768 images at 30 FPS. The processor needs to support 62,915,000 FOPS. It translates to 62.9 MHz clock rate if the filtering operation is done in a cycle. It is impractical to design a DWT processor that performs all filtering operations within a single clock cycle as it will incur very high resource overhead. From the software modelling, 8 additions, 4 multiplications and 4 shifts are needed for the 9/7-F filter. Assuming that the datapath is pipelined properly and 8 cycles are required to complete each filtering operation by using MACs, it translates to $62.9 \times 8 = 503.2$ MHz clock rate. Hence, to support a 2-D DWT processor to process 1024 × 768 images at 30 FPS, we need to design a processor that runs at 503.2 MHz. The typical design guard-band across all process variation, voltage and temperature (PVT) is 30 %, thus the desired operating frequency is 654.2 MHz. However, this frequency is deemed to be very high and the power consumption is significantly higher too. We can resolve this frequency concern by employing parallel processing technique, where the desired frequency is divided by four if we are using four parallel wavelet engines. Assuming we are designing a DWT processor with four wavelet engines, thus the desired frequency is just about 150 MHz. In order to reserve additional headroom for overheads, we decided to set the target frequency at 200 MHz.

3.3 High Level Architecture Overview

The proposed DWT system architecture is leveraged from the architecture presented by Mansouri in 2009 [8]. The key distinguishing feature is the parallel processing and the elimination of LL FIFO. This allows scalability of up to eight parallel wavelet engines that are able to process 1-D transform concurrently.

The proposed architecture shown in Fig. 2 comprises of a main controller, a transpose engine, a central bus and up to eight DWT engines. The main controller contains a state machine and a register block. The system host needs to configure the registers to setup the filtering operation details like the memory base address to retrieve the image and later, store the image once the filtering operation has been completed. Once the setup has been completed, the state machine will instruct the DWT engine to start the filtering operation. At the same time, it will kick-off the transpose operation in the transpose engine. The DWT engine will fetch the image from the memory, perform the 1-D DWT operation and write the output low coefficients into the line buffer in the transpose engine. The transpose engine will read from all line buffers, perform row-column swapping, and write the results back to the main memory. Since the DWT engines and the transpose engine need to access the main memory simultaneously, a central bus with round robin arbiter is needed.

As compared with the original architecture proposed by Mansouri [8], the address generator in the new design is integrated in each 1-D DWT wavelet engine. The address generator is responsible to generate memory addresses to the external memory. The LL FIFO is also removed, which

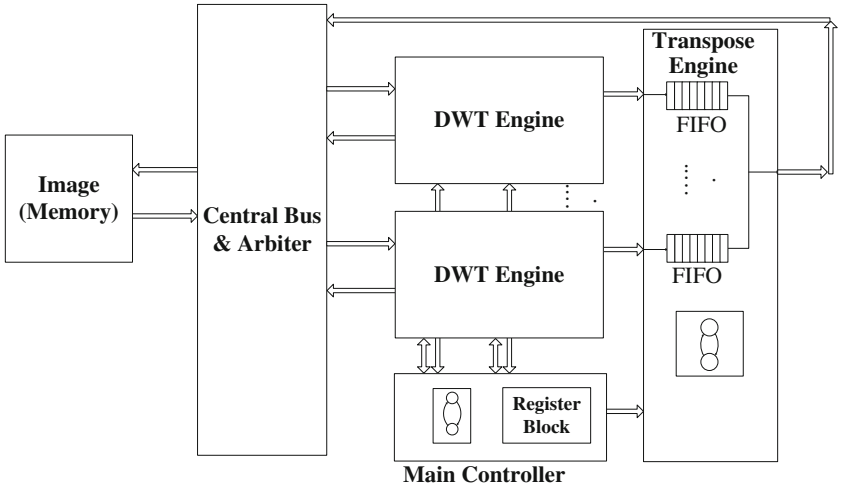

Figure 2 High level conceptual architecture block diagram.

reduces the internal memory usage significantly, but at the expense of memory access bandwidth.

3.4 DWT Engine Design

The DWT engine consists of a FIFO, a memory fetch sequencer, an execution sequencer and a computational block is shown in Fig. 3. The memory fetch sequencer unit is responsible to request central bus to obtain bus ownership for memory read operation. Once the memory read is granted by the arbiter, the memory request is terminated and the current memory address location is incremented. It will fetch image data in 256-bit a chunk from main memory and store it in the FIFO. The execution sequencer will retrieve the data from the FIFO and serialize it to become two sets of 16-bit pixel samples. The two sets of samples (odd and even) are then fed to the DWT computational block. The execution sequencer can provide symmetrical image samples to the computational block to avoid image distortion at the image boundary. Since the pipeline is a 13-stage, a control signal will be generated 13 cycles after the reading of the sample to load the low coefficients into the line buffer in the transpose engine.

The computational block is a fully pipelined design to achieve maximum throughput. The output from the computational block are the low coefficients, which will be stored in line buffers in the transpose engine. It comprises of four lifting calculators (LCs) and a scaling calculator (SC) as shown in Fig. 4. There are pipeline registers placed in between to reduce critical path delay (cpd) and to improve throughput. The even sample is multiplied with a lifting factor (coefficient) and registered first to mitigate the speed path. The rest of the registers are inserted for delay purpose. There is a 16 × 16 multiplier and a 3-input 16-bit adder in

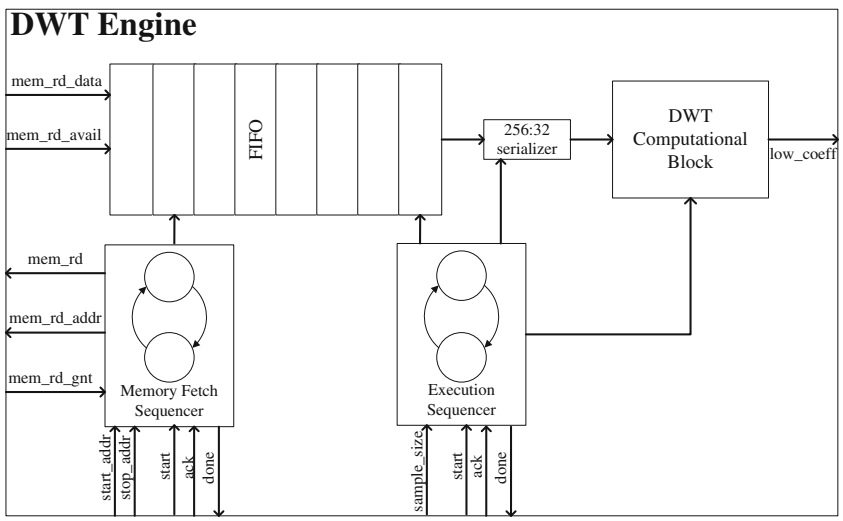

Figure 3 DWT engine architecture block diagram.

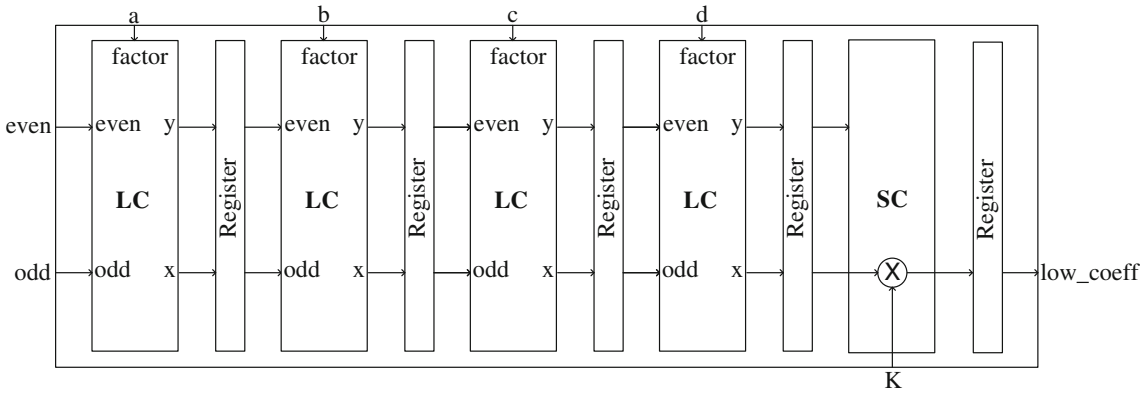

Figure 4 Computational block pipeline design.

the LC. The SC is a 16×16 multiplier. There are 13 pipeline stages in this computational block, thus the latency is 13 cycles.

4 Results and Discussion

The proof-of-concept in this project is done in FPGA, coded in Verilog HDL, and synthesized in Altera Quartus II targeting the Stratix III FPGA family.

4.1 Performance Metrics

The performance metrics for each DWT engine are shown in Table 1. Our architecture needs 10.28 kb memory for 512×512 image processing. A total of 8 kb of buffer is used for intermediate result storage in the transpose engine, while another 0.28 kb of buffer is used to buffer the image from the main memory due to memory access latency. Our architecture does not need LL FIFO compared to Mansouri [8] resulting in 512 kb internal memory saving. The critical path delay is the 3-input 16-bit adder, which has the total of 2.964 ns path delay.

Based on the TimeQuest Timing Analysis tool, the design can perform up to 338 MHz at slow process corner, 85°C of temperature and 1100 mV of operating voltage. The design has 13 pipeline stages (Fig. 4), thus the latency is 13 cycles without considering initial memory access latency. After the initial latency, the design is able to produce 338 Mega Results/s consistently. We also measured the power consumption for the DWT engine by using the PowerPlay Power Analysis tool. Since the full vector is not available, the power consumption was estimated by using vectorless approach, based on assumption of 12.5 % toggle rate. The power consumption measured for each DWT engine is 226.5 mW.

Table 1 Performance metrics of DWT engine.

Parameter	Result
Resource Utilization	
LC Combinationals	233
LC Registers	459
Memory Block Bits	18,720
Timing	
Critical Path Delay (cpd)	2.964 ns
Maximum Frequency (Fmax)	338 MHz
Latency	13 cycles
Throughput	338 Mega results/s
Power	226.5 mW

4.2 Performance Comparison

The 2-D DWT architectures presented earlier (see Section 2) generally require N^2 computation time for $N \times N$ images. We compare the number of multipliers, adders, internal buffer size, and the total computation time with other architectures and the comparison result is shown in Table 2, where m denotes the number of parallel DWT engines. Our proposed architecture uses the least number of adders and multipliers while using the same internal buffer size as Mansouri for the case of $m = 2$ parallel DWT engines. The CT of the proposed architecture is $\frac{3N^2}{32} + \frac{3N^2}{4m}$, where $\frac{3N^2}{32}$ is the number of cycles required to fetch image from the main memory, which is constant regardless of parallel processing. The number of cycles required for filtering operation is $\frac{3N^2}{4m}$, where the computation time can be shorten by parallel processing.

Table 2 Performance comparison with other works.

Architecture	Adders / Multipliers	Buffer size	Computation Time, CT
Conventional Lifting	16 / 12	$N^2 + \frac{N^2}{4}$	$\frac{N^2}{2} + 2\sum \frac{N}{2^j}$
Andra [1]	8 / 4	$4N$	$\frac{N^2}{2} + 2\sum \frac{N}{2^j}$
Wu [12]	36 / 36	$9N$	$\frac{N^2}{2} + 2N$
Hung [5]	12 / 9	$14N$	$N^2 + 4N$
Mansouri [8]	8 / 4	$2N$	$\frac{N^2}{2} + N$
Proposed	8 / 4	$2mN$	$\frac{3N^2}{32} + \frac{3N^2}{4m}$

Our architecture has comparable performance with Mansouri architecture by using 2 parallel DWT engines and performs better than Mansouri architecture [8] with 4 or 8 engines as shown in Fig. 5. The performance improvement from $m = 4$ to $m = 8$ is not very significant due to constant image fetching time from the main memory.

4.3 Peak Signal-to-Noise Ratio (PSNR)

For an 8-bit decompressed image, the PSNR is given as $10\log_{10}\left(\frac{255^2}{MSE}\right)$, where MSE refers to the Mean Square Error between the original and the reconstructed image. For our design using rational number coefficients with denominator greater than 12-bit will give very good decompressed image quality with PSNR of 33 dB. The maximum operating frequency remains consistent regardless of coefficient width.

4.4 Throughput

From the TimeQuest result, the design can operate up to 338 MHz, but realistically we chose 250 MHz as our final operating frequency for a safety margin of 30 %. For an $N \times N$ image, the total CT is given in Table 2. Our design is able to process 1024 × 768 images at 125 FPS with one DWT engine. With 2 DWT engines, it is able to process 1920 × 1080 Full HD video at 85 FPS. It also can process 2048 × 1536 video (QXGA) at 138 FPS by using 8 DWT engines running at 250 MHz. The performance of our DWT design with 2 engines is comparable with Mansouri architecture [8], but with less memory storage.

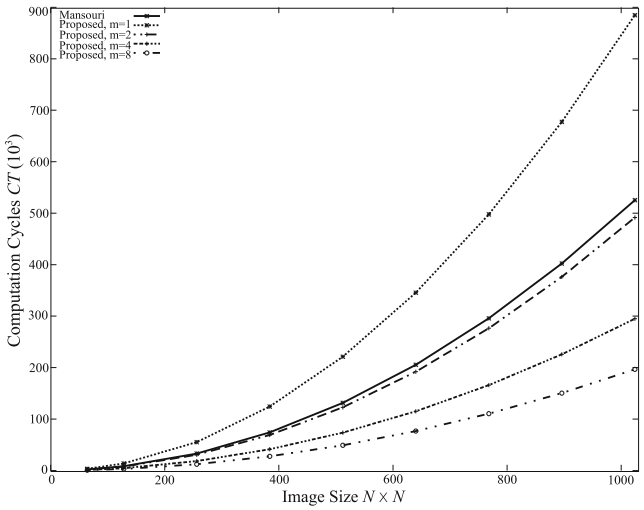

Figure 5 CT comparison with numbers of DWT engines.

5 Conclusion

This paper has proposed and presented a 2-D DWT system architecture for real-time image and video processing which is capable of achieving a maximum throughput of 138 FPS for a 2048 × 1536 image size. The architecture is scalable and can support multiple DWT engines and each DWT engine can operate independently. Hence, the throughput can be increased by a factor of m, where m is the number of parallel DWT engines. The architecture also has advantages of low internal buffer size, low control complexity, fast computational time and low power consumption. The DWT system is modular and flexible enough, and it can be used as generic DWT processor in future for any application that needs wavelet transform. The only change needed is the computational block design, which can be made programmable in the future.

References

1. Andra, K., Chakrabarti, C., Acharya, T. (2002). A VLSI architecture for lifting-based forward and inverse wavelet transform. *IEEE Transaction on Signal Processing*, *50*(4), 966–977.
2. Benderli, O. (2003). *A real-time, low-latency, FPGA implementation of the two dimensional discrete wavelet transform*. M.sc. thesis: The Graduate School of Natural and Applied Sciences of The Middle East Technical University.
3. Chakrabarti, C., & Vishwanath, M. (1995). Efficient realizations of the discrete and continuous wavelet transforms: from single chip implementations to mappings on SIMD array computers. *IEEE Transactions on Signal Processing*, *43*(3), 759–771.
4. Chakrabarti, C., Vishwanath, M., Owens, R.M. (1993). Architectures for wavelet transform. In *Proceedings of the 1993 IEEE workshop on vlsi signal processing* (pp. 507–515). Netherlands: Veldhoven.
5. Huang, C.T., Tseng, P.C., Chen, L.G. (2004). Flipping structure: an efficient VLSI architecture for lifting-based discrete wavelet transform. *IEEE Transactions on Signal Processing*, *52*(4), 1080–1089.
6. Jamkhandi, P., Mukherjee, A., Mukherjee, K., Franceschini, R. (2000). Parallel hardware software architecture for computation of discrete wavelet transform using the recursive merge filtering algorithm. In *International Parallel and Distributed Processing Symposium Workshop* (pp. 250–256). Mexico: Cancun.
7. Mallat, S.G. (1989). A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *11*(7), 674–693.

8. Mansouri, A., Ahaitouf, A., Abdi, F. (2009). An efficient VLSI architecture and FPGA implementation of high-speed and low power 2-D DWT for (9, 7) wavelet filter. *IJCSNS International Journal of Computer Science and Network Security*, *9*(3), 50–60.
9. Martina, M., Masera, G., Piccinini, G., Zamboni, M. (2000). A VLSI architecture for IWT (Integer Wavelet Transform). In *Proceedings of the 43rd IEEE Midwest Symposium on Circuits and Systems* (pp. 1174–1177). USA: Lansing.
10. Sweldens, W. (1996). The lifting scheme: a custom-design construction of biorthogonal wavelets. *Applied and Computational Harmonic Analysis*, *3*(15), 186–200.
11. Vishwanath, M., Owens, R.M., Irwin, M. (1995). VLSI architectures for the discrete wavelet transform. *IEEE Transactions on Circuits and Systems*, *42*(5), 305–316.
12. Wu, B.F., & Lin, C.F. (2003). A rescheduling and fast pipeline VLSI architecture for lifting-based discrete wavelet transform. In *Proceedings of the 2003 IEEE International Symposium on Circuits and Systems (ISCAS 2003), vol. 2, pp. II–732 – II–735. Bangkok*.

Boon Hui Ang received the B. Eng. (hons) degree in Electrical and Electronic Engineering from the Universiti Sains Malaysia, Malaysia, in 2002 and the M. Eng. degree in Electrical - Computer and Microelectronic System from Universiti Teknologi Malaysia, Malaysia, in 2012. He is currently a CPU Design Engineer with Intel Microelectronic Sdn Bhd, Malaysia. His research interests are in low power SoC RTL design and parallel processing.

Usman Ullah Sheikh received his PhD degree (2009) in image processing and computer vision from Universiti Teknologi Malaysia. His research work is mainly on computer vision and embedded systems design. He is currently a Senior Lecturer at Universiti Teknologi Malaysia, Malaysia.

Muhammad N. Marsono received the Ph.D. degree in computer engineering from the University of Victoria, Victoria, BC, Canada in 2007, and the B. Eng. and the M. Eng. degrees in computer engineering and electrical engineering from Universiti Teknologi Malaysia, Malaysia, in 1999 and 2001, respectively. He is currently a Senior Lecturer with the Faculty of Electrical Engineering, Universiti Teknologi Malaysia, Malaysia. His research interests are in multiprocessor system-on-chip, network-on-chip, specialized architectures for network algorithmics and signal processing, and hardware/software co-design.