A New Approach to Implement Discrete Wavelet Transform using Collaboration of Reconfigurable Elements

Asadollah Shahbahrami[1,2], Mahmood Ahmadi[1,3], Stephan Wong[1], and Koen Bertels[1]
[1]Computer Engineering Laboratory, Delft University of Technology, The Netherlands
[2]Department of Computer Engineering, Faculty of Engineering, University of Guilan, Rasht, Iran
[3]Department of Computer Engineering, Faculty of Engineering, Razi University, Kermanshah, Iran
{ *A.Shahbahrami,m.ahmadi,j.s.s.m.wong,k.l.m.bertels* }*@TUDelft.nl*

***Abstract*—The Discrete Wavelet Transform (DWT) is an important operation in applications of digital signal processing. In this paper, we review several traditional DWT implementation approaches, e.g., application-specific integrated circuits, field-programmable gate arrays, digital signal processors, general-purpose processors, and graphic processors, and discuss their limitations in terms of performance and flexibility. In order to provide both high-performance and flexibility, we propose a new approach, namely a parallel architecture exploiting the collaboration of reconfigurable processing elements in grid computing. Grid computing can exploit the task level parallelism to execute the 2D DWT. In addition, reconfigurable computing offers a flexible platform and can be used as hardware accelerators. We mapped the DWT in a grid. Our experimental results show that speedups of up to 4.1x can be achieved.**

***Keywords*-DWT, Grid Computing, Reconfigurable Computing.**

I. Introduction

The Discrete Wavelet Transform (DWT) is a highly effective operation in digital signal processing applications such as image and video compression [1]. This is because it achieves higher compression ratios, e.g. in the JPEG2000 and MPEG-4 standards, than other transforms such as the Discrete Cosine Transform (DCT). In the other hand, the DWT requires more memory than the DCT. The reason is that it operates on an entire image or a large part of it. However a potential problem, however, is the high computational complexity of the DWT. We measured the total execution time consumed by the DWT using the JasPer software tool kit [2]. The results show that the DWT consumes on average 46% of the total encoding time for lossless compression and even 68% for lossy compression. Additionally, results presented by other researchers [3] show that the DWT consumes a significant fraction of the total JPEG2000 encoding time. Considering DWT's popularity, significant research work has been performed on the efficient implementation of the DWT on different platforms.

Different implementation approaches such as application-specific integrated circuits (ASICs) [4], [5], field-programmable gate arrays (FPGAs) [6], [7], Digital Signal Processors (DSPs) [8], [9], and general-purpose processors (GPPs) [10] have been proposed to process the DWT. In addition, recently graphics processing units (GPUs) have been used for parallel implementation of DWT [11].

The aforementioned approaches cannot provide both sufficient flexibility and high-performance at the same time. This is because of the following reasons. The ASIC approaches are a direct mapping of a certain the DWT algorithm to hardware. An ASIC implementation is optimized to provide high-performance for a specific the DWT algorithm with a fixed filter bank length and fixed wavelet decomposition levels and cannot be used for other filter bank lengths nor other transform levels. Reconfigurable elements are more flexible than ASIC designs, but their performance is less than ASIC approaches. More importantly, it has not been focused on exploiting available parallelism for high-performance. DSP approaches do not offer sufficient flexibility and high-performance for different filter bank lengths of the DWT. These approaches have not been optimized to process different the DWT algorithms using floating-point numbers and fixed-point numbers. Media-enhanced GPPs, are flexible and programmable, but have limited performance due to mismatch between storage and computational formats and additional instructions for the conversion between different packed data types and data alignment. The GPUs have been designed as application-specific and have been optimized for specific applications, for instance 3D scene rendering. This means that implementing other applications such as the DWT on the GPUs is a complicated task as mentioned by researchers in [11].

In this paper, a new approach namely a parallel architecture using *collaborating reconfigurable elements in grid computing* is proposed to support various DWT algorithms, filter bank lengths, and transform levels. The main contributions of our work are the following:

- An overview of different approaches to process the DWT is performed.
- In order to benefit both advantages of grid computing and reconfigurable computing, we propose a new approach to process the DWT, collaboration of reconfigurable elements in the grid environment.

978-0-7695-3917-1/09 $26.00 © 2009 IEEE
DOI 10.1109/ReConFig.2009.59

- We map the DWT using collaboration of 4 processing elements in grid computing based on realistic assumptions. Our experimental results show that the speedups of up to 4.1x is achieved over a GPP in a non-grid environment.

This paper is organized as follows. In Section II, the different DWT implementation approaches are discussed. We propose our new approach in Section III. The simulation environment and tools are discussed in Section IV followed by evaluation results in Section V. Finally, overall conclusions is drawn in Section VI.

II. Processor Architectures

The DWT can be processed using GPPs, DSPs, FPGAs, and ASICs. In addition, recently GPUs have been used for parallel processing of the DWT. These approaches are discussed in the following sections.

A. GPPs Enhanced with Multimedia Extensions

There are different implementations of the 2D DWT on GPPs such as [10]. However, implementation of the 2D DWT on GPPs is not efficient. The main reason for this is the discrepancies between the memory access patterns of the two main parts of the 2D DWT, horizontal and vertical filtering. Especially when a large image is processed by both horizontal and vertical filtering, one of them causes to exhibit poor data locality [12]. on cache-aware DWT implementations [12].

Some research work [13], [12] have mapped the 2D DWT on GPPs enhanced with multimedia extensions such as MMX and SSE. Although, Single-Instruction Multiple-Data (SIMD)-enhanced GPPs are flexible and programmable, they have limited performance. This is because of the following reasons [12]. First, there is a mismatch between the computational and the storage formats of the 2D DWT data. This is because the intermediate coefficients are larger than the original input data. In other words, implementing the 2D DWT on GPPs enhanced with SIMD extensions needs many overhead instructions for converting between different packed data types. Second, existing SIMD computational instructions cannot efficiently exploit data-level parallelism of 2D DWT. In order to employ SIMD instructions, data rearrangement instructions are needed to transpose the input matrix. This step takes a significant amount of time.

B. DSP-Based Approaches

Digital signal processors are specifically designed and optimized to process digital signal processing functions. DSPs provide some special hardware units, such as Multiply-ACcumulate (MAC) unit to improve performance. These processors are used to process the DWT. For example, Schelkens et al., [8] implemented an integer DWT on a TMS320C40 platform. They focused on memory optimization and code mapping of an integer lifting scheme. Cho et al., [9] implemented the overlapped block-based lifting scheme using a fixed-point DSP chip. The overlapped block-based approach partitions entire image memory into blocks to fit into the cache size and reorders the sequence of the wavelet lifting. Gnavi et al., [14] presented a performance comparison between traditional convolution-based and lifting scheme approaches on a programmable DSP platform. They observed that the lifting scheme transforms were always faster than their convolution-based counterpart in the context of the JPEG2000. The performance improvement of lifting scheme techniques depended on the wavelet filters length and the number of lifting scheme steps.

Most of the DSP implementations have been focused on integer DWT to achieve high-performance, while floating-point representation of DWT coefficients is usually required in some applications. In addition, DSP architectures do not offer sufficient flexibility and high-performance for different filter bank lengths of the DWT. This is because those processors do not have enough micro-architectures to support memory intensive functions, such as the DWT. The 2D DWT operates on a complete image, it needs many read write operations in the memory system. As was mentioned in [15], one of the biggest bottlenecks in executing digital signal processing algorithms using DSPs is transferring data to and from memory.

C. FPGA-Based Approaches

In order to provide a flexibility for the DWT implementation based on wavelet filter length and wavelet decomposition structure, some reconfigurable hardwares were proposed in [6], [7]. Tseng et al., [6] proposed an architecture, which consists of a reconfigurable processing element array and a reconfigurable address generator. In addition, a reconfigurable architecture for an integer 8-point Haar wavelet transform was proposed in [7]. Reconfigurable architectures are more flexible than ASIC designs, while their performance is less than ASIC approaches. Some of reconfigurable implementations have focused on Fine-Grained Reconfigurable Architectures (FGRA) such as FPGAs. Those FGRAs use more logic blocks to implement the circuit. The resulting interconnection delays hinder the performance.

On the other hand, a Coarse-Grained Reconfigurable Architecture (CGRA) decreases the total number of logic blocks and therefore, the interconnect complexity is reduced by localizing the connections. MATRIX [16] is a CGRA which has been proposed to overcome the drawbacks of FGRAs. MATRIX consists of a 2D array of 8-bit Functional Units (FUs) and each FU consists of an 8-bit ALU, memory, and control logic. While fewer configuration bits are needed for the PEs, fully utilizing the functionality of each PE is difficult, leading to significant underutilization of CGRAs. This can be avoided by either Medium-Grained Reconfigurable Architectures (MGRAs) or by adapting the reconfigurable architecture to the computational characteristics of the

algorithm with application-specific grained processing elements. An application-specific coarse-grained reconfigurable architecture can benefit from the high-performance of ASIC designs as well as the flexibility of the reconfigurable devices such as FPGAs.

D. ASIC-Based Approaches

ASIC approaches have been extensively used in order to provide high-performance for the DWT. ASIC-based architectures to support the DWT can be classified into different categories depending on some metrics such as style of the computation and topology of processing elements. Based on the computational style, different architectures can be categorized into serial and parallel computations. In serial architectures, in each single time step, each MAC unit performs one multiplication and one addition. Parallel architectures have L multipliers, where L is the filter length. The multiplications are performed in parallel. The DWT can be implemented as 1D, 2D, or 3D, therefore, different ASIC approaches were proposed. Each approach is often designed for a specific application in mind. Some 1D architectures have been discussed in [17] and some 2D and 3D architectures have also been explained in [4]. Liao [5] proposed architectures for 1D and 2D lifting scheme wavelet transform. Weeks [18] discussed different ASIC architectures to support the DWT and compared them based on design type, latency, area, memory, and number of multipliers and adders. Most ASIC architectures are based on the integer wavelet transform, especially on the lifting scheme. However, this choice might has negative impact on precision.

ASIC approaches are a direct mapping of a certain the DWT algorithm with a fixed filter bank length to hardware. The implemented hardware is optimized to provide high performance with low power for a fixed filter bank length and a fixed wavelet decomposition level and it cannot be used for various filter bank lengths and various transform levels with different precisions which are used in the DWT implementation. In other words, flexibility is a key requirement to implement the DWT within existing applications and to adapt to new generations of multimedia applications and standards.

E. GPUs-Based Approaches

GPUs were designed as application-specific units with control and communication units that enable the use of many ALUs. Several researchers have implemented the DWT on the GPUs [11], [19]. Wang, et al., [19] have implemented the DWT targeted on the Jasper software on the GPUs. However, those implementations cannot efficiently exploit all the hardware resources available in GPUs. Tenllado [11] mapped the parallel implementation of the traditional convolution-based and lifting scheme approaches of the DWT on GPUs. The GPUs have been designed as application-specific and have been optimized for some applications, for instance the 3D scene rendering. This means that implementing other applications such as DWT on the GPUs is a complicated task as mentioned by researchers in [11].

III. Collaboration of Reconfigurable Elements in Grid Computing

A well-known method to increase the performance is to execute as many as possible parts of the DWT in parallel. Therefore, the architecture executing the DWT has to exploit the potential parallelism. The DWT has a huge number of calculations in parallel and in a regular manner. This is because both low-pass and high-pass filters are applied modularly and regularly to regions of each image. The processing of each region of an image can be performed independently from the other regions. This feature means that the DWT has a large potential for parallelism in terms of data computation. In addition, the exploitation of the parallelism available in grid computing is a very attractive solution for the execution of the DWT. This is the reason to propose a grid computing to process the DWT efficiently. In grid computing, a large pool of heterogeneous computing resources is geographically dispersed over a large network, e.g., the Internet.

The reconfigurable computing offers a flexible platform to implement the DWT with various filter bank lengths and various transform levels. Reconfigurable architectures offer a compromise between the performance advantages of ASIC and the flexibility of DSPs. In order to benefit both advantages of grid computing and reconfigurable computing we propose a parallel architecture, collaboration of Reconfigurable Elements (REs) in grid computing, to process the DWT efficiently. Collaboration of different architectures should be used in order to meet the computational demand of DWT. A general overview of the Collaboration of Reconfigurable elements in Grid Computing (CRGC) is depicted in Figure 1.

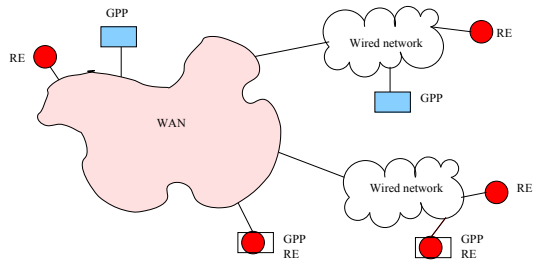

Figure 1. A collaborative system in a grid environment with reconfigurable elements and general-purpose processors.

In CRGC, processing elements communicate and collaborate together based on the *neighborhood concept* [20]. Each grid processing element requests assistance from neighboring processing elements. The neighborhood concept is implemented using some primitives. A primitive is defined as a processing element with related communication link and its equipments, e.g., routers and switches, to the main

processing element. A grid network can be seen as a collection of primitives.

The following steps should be considered in order to execute an application on the CRGC. First, a network topology based on the availability of neighbor processing elements is defined and some parameters such as network bandwidth and packet size are configured. Second, the processing elements are defined and reconfigurable processing elements are configured based on the application characteristics. For example, an application may use the third level DWT decomposition and another application may use the fifth level DWT decomposition. Since some REs are configured for the former one and some REs are also configured for the latter one. Third, an application mapping policy and number of subtasks (gridlets) for each application are determined. Finally, the main processing element packetizes the subtasks and send them to appropriate REs. Additionally, the collaborator REs depacketizes the received packets and process them and send back the calculated results to the main processing element. Furthermore, the main processing element receives the final results and send the remaining other subtasks to the idle REs.

In the following sections, we present simulation environment and the experimental results which have been obtained using the CRGridSim simulator.

IV. Simulation Environment and Tools

The simulation environment is an extended version of GridSim (a traditional Java-based discrete-event grid simulator) [21]. We configured and prepared GridSim simulator based on the DWT properties to support the collaborative processing between reconfigurable elements. This extension of GridSim is called CRGridSim. In CRGridsim, the DWT can be broken down to different subtasks called gridlets. the DWT is packaged as gridlets whose contents include the task length in Millions of Instructions (MIs). The task length is expressed in terms of the time it takes to run on a standard GPP. In this paper, we have used 30, 35, 40, and 50 MIPs for processing elements. To simplify the simulation, the following assumptions have been made. First, REs do not support partial reconfiguration. Second, reconfiguration codes are stored in the target collaborator elements.

The DWT has been simulated on the CRGC with different configurations using a star topology. In our case, one main GPP works with 2 or 3 collaborator processing elements. In other words, we assumed 3 and 4 processing elements in grid computing. Reconfigurable elements and GPPs are used as the collaborator processing elements. It should be noticed that for star topology the Routing Information Protocol (RIP) is executed by the simulator. The arbitrary number of collaborators can be used that is depends on the application requirements and available processing elements. However, increasing the number of processing elements increases the overhead, for example the communication time. In order to map the DWT on reconfigurable elements, the possible reconfigurable parameters should be identified. For example, variable wavelet filter lengths and variable wavelet decomposition levels are two reconfigurable parameters.

The specifications of the simulated environment are depicted in Table I. Performance of reconfigurable elements for grey level co-occurrence matrix (GLCM) and Haralick texture feature for image sizes $512 * 512$, $1024 * 1024$ and $2048 * 2048$ was presented in [22]. The speedups of 4.75 and 7.3 were obtained when compared with a general-purpose processor for GCLM and Haralick co-occurrence matrix, respectively. In addition, a co-occurrence matrix media kernel has been implemented on the various FPGA devices such as Virtex-2 and Spartan-3 and on a media-enhanced GPPs using MMX technology in [23]. Speedups of 20 were obtained using FPGA implementations over media-enhanced GPPs, for image size $512 * 512$. In the results section, the speedups of 10 were used. The reason for

Parameter	Value
Maximum packet size	32 and 65 KBytes
User-router bandwidth	100 Mb/sec
Router-router bandwidth	1000 Mb/sec
Number of images	40
Number of users	1
Size of images	different sizes from 768 to 2848
Minimum speedup for RE	10 in compared to a GPP
Reconfiguration file size	3 Mb
Reconfiguration speed	3 Mb/sec
Reconfiguration time	1 sec
Number of bits per pixel	24 bit

Table I
SPECIFICATIONS OF THE SIMULATED ENVIRONMENT.

choosing the packet sizes of 32 KBytes and 65 KBytes (the largest packet sizes in the networks) is based on our observation. Our results show that using larger packet sizes obtain more performance than smaller packet sizes. Larger packer sizes decreases the communication overhead due to sending less packets. In our case we have 40 images with total size of 294 MBytes that concludes 4539 packets of 65 KBytes and 9219 packets of 32 KBytes.

V. Experimental Results

In this section, we present the experimental results which have been obtained using the CRGridSim simulator.

A. Mapping of the 2D DWT on CRGC

We map the 2D DWT on two different configurations with packet size of 65 KBytes. The first configuration is the collaboration of GPPs. The second configuration is the collaboration of reconfigurable elements with a GPP. Table II and Table III show the mapping of 2D DWT on the first and second configurations, respectively. The GPP0 is the main processing element, while other processors are the collaborators processing elements. The second column shows the total number of processed gridlets by each processing element. The last column represents the total number of executed

instructions by each processing element. For instance, in Table II, GPP1 executes 807 MIs.

Resource	Total # of assigned gridlets	Total # of ins. (MIs)
GPP0	15	1704
GPP1	7	807
GPP2	8	1047
GPP3	10	762

Table II
APPLICATION MAPPING OF THE 2D DWT ON COLLABORATION OF GPPS ON A GRID COMPUTING.

Resource	Total # of assigned gridlets	Total # of ins. (MIs)
GPP0	9	1131
RE1	10	762
RE2	10	1226
RE3	11	1201

Table III
APPLICATION MAPPING OF THE 2D DWT ON COLLABORATION OF RECONFIGURABLE ELEMENTS ON A GRID COMPUTING.

Three collaborator processing elements in Table II process $7+8+10=25$ gridlets, while the main processing element processes 15 gridlets. In other words, the GPP0 processes the most number of gridlets. On the other hand, three collaboration reconfigurable elements in Table III process $10+10+11=31$ gridlets, while the main processing element processes 9 gridlets, which is the least number of gridlets compared to other processing elements. As a result, the collaboration of reconfigurable elements process more gridlets than the collaboration of GPPs.

B. Performance Evaluation Results

Figure 2 depicts the speedups of the different configurations of 3 and 4 collaborator processing elements over a GPP in the first level DWT decomposition with packet sizes of 32 and 65 KBytes. The configuration of 1 GPP means that there is no any collaborator elements, while the configuration of 1 GPP and 3 RE means that 3 REs are collaborating with 1 GPP. The collaboration of 1 GPP and REs increases the performance compared to other configurations. This is because REs have been used as hardware accelerators. In other words, the performance is increased by using hardware accelerators. In addition, the results show that the increasing the packet size from 32 KBytes to 65 KBytes enhances the performances. This is because it reduces the number of packets which should be sent to the collaborators processing elements and therefore, the communication time is decreased. Additionally, increasing the number of collaborators improves the speedups. This is because the submitted subtasks to each collaborators are decreased that reduces the number of processed instructions by each processing element.

In order to increase the computational time of each collaborator processing element, we have implemented the third level DWT decomposition. Figure 3 depicts the speedups of the different configurations of 3 and 4 collaborator processing elements over 1 GPP in the third level DWT decomposition with the packet sizes of 32 and 65 KBytes.

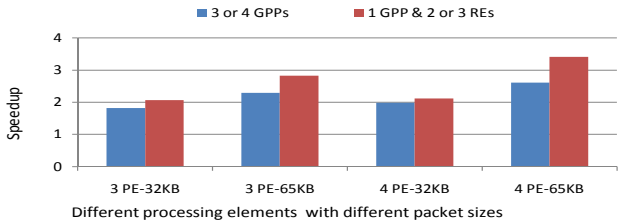

Figure 2. Speedups of the different configurations of 3 and 4 collaborator processing elements over 1 GPP in the first level DWT decomposition with packet sizes of 32 and 65 KBytes.

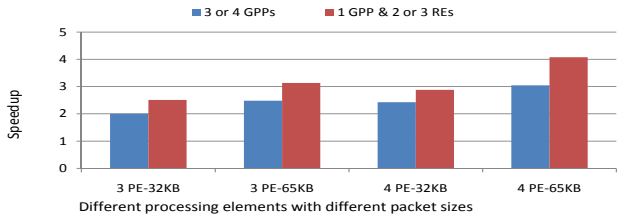

Figure 3. Speedups of the different configurations of 3 and 4 collaborator processing elements over a GPP in the third level DWT decomposition with packet sizes of 32 and 65 KBytes.

The speedups of up to 4.1x are obtained, while for the first level DWT decomposition in Figure 2, the speedups up to 3.4x are yielded. This is because the third level DWT decomposition has much more computations than the first level DWT decomposition.

In general, the communication time and managing the synchronization of processing elements are the bottlenecks. One way to reduce the impact of those bottlenecks is to execute time consuming functions with larger packet sizes as we have performed for second and the third level DWT decomposition in Figure 4.

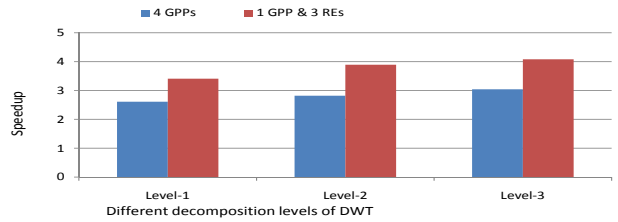

Figure 4. Speedups of the different configurations of 4 collaborator processing elements, over a GPP in different levels the DWT decomposition with packet size of 65 KBytes.

This figure depicts the speedups of the different configurations of 4 processing elements over 1 GPP in different levels the DWT decomposition with the packet size of 65 KBytes. The computational time is increased by processing the higher levels of the DWT decomposition. As a consequence, the speedups are improved by increasing the computational time.

VI. CONCLUSIONS

The requirements of the DWT have been not matched well with the ability of the existing approaches. This is

because ASIC approaches are a direct mapping of a filter bank length to hardware. DSP approaches do not offer sufficient flexibility and high-performance for different filter bank lengths of the DWT. SIMD-enhanced GPPs are flexible and programmable, while they have some limitations. Reconfigurable elements are more flexible than ASIC designs, while their performance is less than ASIC approaches. In addition, implementing the DWT on graphics processing units is still a complicated task. In order to provide both high-performance and flexibility, we used both grid computing and reconfigurable elements. In other words, to benefit from the advantages of grid computing and reconfigurable computing we proposed a parallel architecture, collaboration of reconfigurable elements in grid computing, to process the DWT. We have simulated the collaboration of 4 processing elements in the grid environment using CRGridSim simulator based on realistic assumptions. We have also mapped the DWT on the simulated environment. This approach provides the speedups of up to 4.1x.

REFERENCES

[1] M. Ghanbari, *Standard Codecs: Image Compression to Advanced Video Coding*. The Institution of Electrical Engineers London, 2003.

[2] M. D. Adams and R. K. Ward, "JasPer: A Portable Flexible Open-Source Software Tool Kit for Image Coding/Processing," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, vol. 5, 2004, pp. 241–244.

[3] S. Chatterjee and C. D. Brooks, "Cache Efficient Wavelet Lifting in JPEG 2000," in *Proc. IEEE Int. Conf. on Multimedia*, 2002, pp. 797–800.

[4] K. Andra, C. Chakrabarti, and T. Acharya, "A VLSI Architecture for Lifting-Based Forward and Inverse Wavelet Transform ," *IEEE Trans. on Signal Processing*, vol. 50, no. 4, pp. 966–977, 2002.

[5] H. Liao, M. K. Mandal, and B. F. Cockburn, "Efficient Architectures for 1D and 2D Lifting-Based Wavelet Transforms," *IEEE Transactions on Signal Processing*, vol. 52, no. 5, pp. 1315–1326, 2004.

[6] P. C. Tseng, C. T. Huang, and L. G. Chen, "Reconfigurable Discrete Wavelet Transform Processor for Heterogeneous Reconfigurable Multimedia Systems," *Journal of VLSI Signal Processing Systems*, vol. 41, no. 1, pp. 35–47, 2005.

[7] E. Hyun, M. Sima, and M. McGuire, "Reconfigurable Implementation of Wavelet Transform on an FPGA-Augmented NIOS Processor," in *Proc. IEEE Canadian Conf. on Electrical and Computer Engineering*, 2006.

[8] P. Schelkens, F. Decroos, J. Cornelis, G. Lafruit, and F. Catthoor, "Implementation of an Integer Wavelet Transform on a Parallel TI TMS320C40 Platform," in *Proc. IEEE Workshop on Signal Processing Systems*, 1999.

[9] J. K. Cho, M. C. Hwang, J. S. Kim, B. D. Choi, and S. J. Ko, "Fast Implementation of Wavelet Lifting for JPEG2000 on a Fixed-Point DSP," in *Prof. Int. Conf. on Circuits, Systems, Computers, and Communications*, 2004.

[10] D. Chaver, C. Tenllado, L. Pinuel, M. Prieto, and F. Tirado, "2-D Wavelet Transform Enhancement on General-Purpose Microprocessors: Memory Hierarchy and SIMD Parallelism Exploitation," in *Proc. Int. Conf. on the High Performance Computing*, 2002.

[11] C. Tenllado, J. Setoain, M. Prieto, L. Pinuel, and F. Tirado, "Parallel Implementation of the 2D Discrete Wavelet Transform on Graphics Processing Units: Filter Bank versus Lifting," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 3, pp. 299–310, 2008.

[12] A. Shahbahrami, B. Juurlink, and S. Vassiliadis, "Implementing the 2D Wavelet Transform on SIMD-Enhanced General-Purpose Processors," *IEEE Trans. on Multimedia*, vol. 10, no. 1, pp. 43–51, 2008.

[13] R. Kutil, "A Single-Loop Approach to SIMD Parallelization of 2D Wavelet Lifting," in *Proc. 14th Euromicro Int. Conf. on Parallel, Distributed, and Network-Based Processing*, 2006, pp. 413–420.

[14] S. Gnavi, B. Penna, M. Grangetto, E. Magli, and G. Olmo, "Wavelet Kernels on a DSP: a Comparison Between Lifting and Filter Banks for Image Coding," *EURASIP Journal on Applied Signal Processing*, vol. 2002, no. 1, pp. 981–989, 2002.

[15] S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, 1999.

[16] E. Mirsky and A. DeHon, "MATRIX: a Reconfigurable Computing Architecture with Configurable Instruction Distribution and Deployable Resources," in *Proc. IEEE Symp. on FPGAs for Custom Computing Machines*, 1996, pp. 157–166.

[17] T. C. Denk and K. K. Parhi, "VLSI Architectures for Lattice Structure Based Orthonormal Discrete Wavelet Transform," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 44, no. 2, pp. 129–132, 1997.

[18] M. Weeks and M. Bayoumi, "Discrete Wavelet Transform: Architectures, Design and Performance Issues," *Journal of VLSI Signal Processing*, vol. 35, pp. 155–178, 2003.

[19] J. Wang, T. Wong, P. Heng, and C. Leung, "Discrete Wavelet Transform on GPU," in *Proc. ACM Workshop General-Purpose Computing on Graphics Processors*, 2004.

[20] S. Wong and M. Ahmadi, "Reconfigurable Architectures in Collaborative Grid Computing: An Approach," in *Proc. 2nd Int. Conf. on Networks for Grid Applications*, 2008.

[21] A. Sulistio, G. Poduval, R. Buyya, and C. K. Tham, "On incorporating differentiated levels of network service into gridsim," *Future Generation Computer Systems*, vol. 23, no. 4, pp. 606–615, 2007.

[22] M. A. Tahir, A. Bouridane, F. Kurugollu, and A. Amira, "Accelerating the Computation of GLCM and Haralick Texture Features on Reconfigurable Hardware," in *Proc. of th Int. Conf. on Image Processing*, 2004, pp. 2857–2860.

[23] D. K. Iakovidis, D. E. Maroulis, and D. G. Bariamisa, "FPGA Architecture for Fast Parallel Computation of Co-occurrence Matrices," *Microprocessors and Microsystems*, vol. 31, no. 2, pp. 160–165, 2007.