A High-Throughput and Memory Efficient 2-D Discrete Wavelet Transform Hardware Architecture for JPEG2000 Standard

G. Dimitroulakos1, M. D. Galanis2, A. Milidonis3, and C.E. Goutis4
VLSI Design Lab., Electrical and Computer Engineering Department, University of Patras, Patras, Greece
{dhmhgre1,mgalanis2,milidon3,goutis4}@ee.upatras.gr

Abstract— The design and implementation of an efficient hardware architecture in terms of speed and memory requirements for computing the tile-based Two-Dimensional Forward Discrete Wavelet Transform for the JPEG2000 still image compression standard, is described in this paper. This architecture is derived from a well-established architecture template for calculating the Two-Dimensional Forward Discrete Wavelet Transform. The filters of that template are replaced by our previously published throughput-optimized ones. A proper scheduling algorithm has been developed that it matches to the special features of our filtering units. The performance improvements are due to the throughput-optimized filters. Also, due to the developed scheduling algorithm, reduced memory requirements are achieved when compared with previously published architectures.

I. INTRODUCTION

The Discrete Wavelet Transform (DWT) has been introduced as an effective and flexible methodology for subband decomposition of signals [1]. This transform exhibits good algorithmic characteristics, which are the reasons for its wide usage in contemporary multimedia compression standards, such as the JPEG2000 [2] and MPEG-4 [3].

A large variety of hardware architectures for implementing the Two-Dimensional separable Forward (2D-DWT) and Inverse DWT (2D-IDWT) have been presented [4], [5], [6], [7], [8] and [9]. These architectures are composed by filters for performing the One-Dimensional (1D) DWT and memory units for storing the results at each stage of transformation. The requirement for optimizing the filters’ architecture in terms of performance is imposed by the fact that multimedia applications, in which the DWT is a part, are characterized by high throughput requirements. The minimization of the memory size can be achieved by setting up a proper sequence of the computations, called time scheduling. The goal of scheduling is to maximize the utilization of the filtering units, and to minimize the memory buffering between the computation stages.

In principle, the 1D-DWT architectures can be extended to architectures for computing the separable 2D-DWT. This is due to the fact that the separable 2D-DWT can be computed by 1D-DWT filtering on rows of an input image followed by 1D-DWT filtering on columns. In the Tile Based (TB) 2D-DWT the input image is optionally decomposed into a number of non-overlapping rectangular blocks, called tiles and the separable 2D-DWT is applied inside each tile independently.

In this paper, an optimized architecture in terms of performance and memory requirements for computing the TB 2D-DWT in a JPEG2000 encoder is presented. The proposed architecture is based on a well-known architecture template, presented in [8], where the four conventional filters [5], [7] have been replaced by the four Throughput-Optimized (TO) ones presented in [10]. An efficient scheduling, that it is based on the line-based algorithm for computing the separable 2D-DWT [9] and it is suited to the filters’ special characteristics, is proposed. This scheduling algorithm minimizes 2D-DWT computation memory requirements between the levels of decomposition. Also, it results in improved throughput characteristics, which are due to the usage of our filters in the DWT architecture.

The rest of the paper is organized as follows: Section 2 presents the related work while section 3 gives the basic theoretical background. Section 4 illustrates the 2D-DWT encoder architecture and presents the scheduling algorithms. The memory requirements and performance of the proposed 2D-DWT encoder architecture and their comparison with existing encoders are illustrated in section 5. Finally, section 6 concludes the paper.

II. RELATED WORK

Chakrabarti and Vishwanath [4] have proposed an extensible architecture for the encoder based on the non-separable 2D-DWT. This architecture consists of two parallel filtering units of size $K^2$ and a storage unit of size $\approx N\cdot K$. A parallel filter of size $M$ consists of $M$ multipliers and a tree of adders to add the $M$ products. Vishwanath et al. [6] proposed an architecture for separable 2D-DWT, which consists of two systolic arrays of size $K$, two parallel filters of size $K$, and a storage unit of size $\approx N(2\cdot K^2J)$. A disadvantage of this architecture is that two rows of the input image are supplied to the two systolic arrays every two cycles and as a result, an additional data converter is required to convert the raster scan input (one per cycle) into two per two cycles output.

Chakrabarti and Mumford [8] proposed an architecture for the analysis (synthesis) filters based on the 2D-DWT. Two scheduling algorithms for computing the forward (inverse) 2D-DWT were also described. The goal was to minimize the memory requirements and to keep the data-flow as regular as possible. Zervas et al.[7] compares the three main hardware architectures for computing the
2D-DWT - level by level, line-based [9] and block-based - in terms of memory requirements, throughput and energy dissipation.

III. PRELIMINARIES

As previously mentioned, the computation of the separable 2D-DWT is based on the application of two 1D-DWTs on the rows and columns of the input image. The 1D-DWT is a two-channel sub-band decomposition of an input signal \( x[n] \):

\[
y_1[n] = \sum_y x[y] h[2n - k] \quad (1)
\]

\[
y_0[n] = \sum_y x[y] w[2n - k] \quad (2)
\]

The filters' architecture used in the 2D-DWT computation is based on the Eqs. (1) and (2). Their detailed architecture is given in [10]. These filters are composed of a modified delay line that it can be derived by a well-known architecture template presented in [8], by: (i) by replacing the filters with the ones presented in [10], and (ii) by estimating the size for the memory units. So, the encoder architecture consists of: (a) two parallel computational units - Filter 1 (F1) and Filter 2 (F2) - for computing the \( L \) and \( LL/H \) 1D-DWT coefficients respectively, along the rows, (b) two parallel computational units - Filter 3 (F3) and Filter 4 (F4) - for computing the \( (LL)LL, (LL)LH \) and \( LL\)LL 1D-DWT coefficients respectively, along the columns, and (c) two Storage Units (SUs) for storing the intermediate results among the stages of computation.

The scheduling algorithm consists of two parts. The first part describes the scheduling of the filters for performing the 2D-DWT in a tile and it is based on the line-based approach [9]. The second part refers to the scheduling of the filters among the tiles. This is performed for reducing as much as possible the computational overhead imposed by the TB 2D-DWT. Both the architecture and the scheduling algorithm comply with the following principle: proceed to the next layer of filtering as soon as possible, without interfering filtering along a row of the input image.

A. Scheduling algorithm description inside the tile

The proposed encoder architecture is based on an algorithm similar to the 1D Recursive Pyramid Algorithm [6]. According to this algorithm, F1 scans the rows of the input image, which have size \( N \), and produces \( N/2 \) high-frequency transform coefficients and \( N/2 \) low-frequency coefficients noted as \( H \) and \( L \), respectively. The coefficients are produced in a row-major order, one \( L \) and one \( H \) coefficient per clock cycle and then are stored into SU 1. This filter is always occupied with the calculations of the \( L \) and \( H \) sub-band.

F3 reads the low-pass coefficients \( LL\)LL from SU1 and produces the \( (LL)LL \) and \( (LL)LH \) coefficients \( (0 \leq \alpha < \beta) \) by applying vertical filtering along the columns in a row-major order. In this way the consumption of the coefficients resembles the way they are produced. Also a higher priority is given to the lower stages calculations, since F3 is used for all \( (LL)LL \) coefficients.

F4 works in parallel with F3, but reads the high-pass coefficients \( (LL)LH \) from SU1 and computes the \( (LL)LL \) and \( (LL)LH \) coefficients along the columns in the same way as F3 operates. Both filters load the coefficients for each filtering in parallel. So, one \( (LL)LL \) and \( (LL)LH \) coefficient is produced per each clock cycle. F2 computes the \( (LL)LL \) and \( (LL)LH \) coefficients.

\[
\begin{figure}
\end{figure}
\]
as shown in Fig.2, and its outputs are stored into SU1. The filtering is not interleaved along a row. Also, the low-pass coefficients \((LL)L\) are consumed by F2 in the same way they are produced by F3 since F2 and F3 operate in a lock-step manner. Therefore, there is no need for buffering between F2 and F3.

SU1 loads the \((LL)L\) coefficients for applying the filtering to subsequent stages. For each stage the horizontal sliding of the vertically positioned filter mask of Filter 3 and 4 imposes the need of storing \(K/(N/2^a)\) coefficients \((1 \leq a \leq J)\) for meeting the requirements of the aforementioned filters’ operation. Thus, the total memory size (in words), required for \(J\) stages of decomposition, equals to:

\[
K \cdot N + K \sum_{a=1}^{\infty} \frac{N}{2^{a+1}} = K \cdot N \left(1 - \frac{1}{2^J}\right)
\]

In Fig.3, an example of our architecture is presented for the case of an image size of 16x16 pixels, for \(J=3\) levels of decomposition and for using the 5/3 filter bank.

---

The proposed scheduling algorithm consists of the following steps:

i) Schedule the \(L_{ij}\) and \(H_{ij}\) coefficients at time instance:

\[
t_{L,H}(i,j) = \frac{N}{2} \cdot i + j
\]

(ii) Schedule the \((LL)L_{ij}, (LL)H_{ij}, (LL)L_{ij}, (LL)H_{ij}\) coefficients at time instance:

\[
t_{LL}(i,j) = 2^a \cdot (K-1) \cdot \frac{N}{2} + (1 - \frac{1}{2^a}) \cdot N + 2^{a+1} \cdot N \cdot i + j + 1
\]

(iii) Schedule the \((LL)L, (LL)H\) coefficients at time instance:

\[
t_{LL}(i,j) = 2^a \cdot (K-1) \cdot \frac{N}{2} + (1 - \frac{1}{2^a}) \cdot N + 2^{a+1} \cdot N \cdot i + j + \frac{K}{2}
\]

where \(i=0,1,2,\ldots,N/2^a\); \(j=0,1,2,\ldots,N/2^a\); \(1 \leq a \leq J\) and \(i, j\) stands for the \(i\)-th row and the \(j\)-th column respectively.

B. Scheduling algorithm description among tiles

Taking into account the scheduling Eqs. (4), (5) and (6), it can be deduced that the 2D-DWT computation on a tile can be separated into three phases (Fig.4a). At each stage, a filtering unit may be idle; meaning that either waits for the appropriate number of input samples to start a computation or that the computation for this filter has ended for the specific tile.

At the first stage, only F1 operates while F2, F3 and F4 are in an idle state waiting for the appropriate number of samples. This stage lasts until the first sample of the \((LL)\) level is produced. Hence, according to Eq. (5) this stage holds for \(c_1 = N(K-1)/4 + 1\) cycles. After this time instant, F2, F3 and F4 start to operate until the end of the filtering of the tile. However, F1 doesn’t work all the time. It switches to idle state after producing the \(L\) and \(H\) sub-band. According to Eq. (4) this takes place at time instant \(c_2 = N^2/2 - 1\). According to Eq. (5), the number of cycles required to perform the filtering in one tile is:

\[
c_1 = (2^a - 1) \frac{N(K-1)}{4} + N \left(1 - \frac{1}{2^a} \right) + 2^{a+1} \frac{N}{2} - 1 + \frac{N}{2^a}
\]

Figure 4. a) Phases in the tile computation, b) interleaved and non-interleaved scheduling

Thus, according to Fig.4a it is concluded that the time where F1 remains idle is \(c_2 = c_3, c_4\). Instead of letting F1 going into idle state, the encoder architecture starts processing the next tile without having conflict with the previous one. This is ensured by assuming that the number of decomposition levels is greater than 1. Then, \(c_4 > c_3\) is always satisfied and interleaving is possible. Thus, after F2, F3 and F4 have finished the filtering with the previous tile, they start processing the already produced sub-band samples \((L)\) of the next tile without passing through the idle state (Fig.4b). Equation (9) gives the number of cycles for performing the TB 2D-DWT
with interleaving computations among tiles where \( l \) is the factor by which the image is separated into tiles. Additionally, Eq.10 gives the number of cycles without interleaving.

\[
C_{l-int}(N,a) = c_l\left(\frac{N}{2^l},a\right) + (2^l - 1)c_l\left(\frac{N}{2^l},a\right)
\]  
\[
C_l(N,a) = 2^l \cdot c_l\left(\frac{N}{2^l},a\right)
\]

By comparing Eqs. (9) and (10) it is easily shown that \( C_l(N,a) > C_{l-int}(N,a) \). Hence, by interleaving the filtering among tiles, a reduction in the additional cycles imposed by the TB 2D-DWT can be achieved.

V. COMPARISON RESULTS

In this section, the proposed and previously published 2D-DWT encoder architectures are compared in terms of the memory size and performance. The results for the total cycles needed to perform the 2D-DWT on a 128x128 image, for the case of the 9/7 filter bank, are shown in Fig.5. The experimental setup is based on 12 different scenarios regarding the tile factor \( l \) (1, 2, 3) and the number of decomposition stages \( J \) (2, 3, 4).

For each scenario, the total number of cycles required for performing the 2D-DWT, using our architecture, - with three different ways for implementing the 2D-DWT (TB with interleaving, TB without interleaving and non-TB case) - are compared with the cycles needed by the architectures of [7] and [4]. It is deduced from Fig.5, that the performance of the proposed architecture outperforms approximately by a factor of two that of the existing architectures for all scenarios when applying the 2D-DWT without tiling. Even when tiling is used, the performance of the proposed architecture is still better in most cases than that of the previously published architectures. The experimental results for image sizes 64x64 and 256x256 and for the other filter banks are analogous.

In Fig.6, the comparison of the memory requirements for the proposed 2D-DWT encoder and the existing encoding architectures presented in [7], [4] and [5] is shown for the case of an input image with size 128x128 without the use of tiling. The results are based on 12 different scenarios regarding the levels of decomposition \( J \) (2, 3, 4) and the filter length (5, 9, 10, 18). It is deduced that the memory requirements of the existing architectures overcome these of the proposed one in all cases. In particular, the memory requirements of the most efficient encoder, which is presented in [7], are 2% to 9.3% larger than the proposed one while its performance is significantly worse.

VI. CONCLUSIONS

In this paper a JPEG2000 compliant 2D-DWT architecture is presented. It is characterized by higher performance and low memory requirements when compared to existing 2D-DWT encoder architectures. Also, it partially overcomes the performance bottleneck imposed by the TB 2D-DWT, by interleaving the computations among tiles and still can benefit from the low memory requirements of this type of wavelet transform.

ACKNOWLEDGEMENTS

We thank the European Social Fund (ESF), Operational Program for Educational and Vocational Training II (EPEAEK II), and particularly the Program HERAKLEITOS, for funding the above work.

REFERENCES