International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 1 Issue 7, September - 2012

# Survey on Lifting based 2-D DWT VLSI Architectures

S.Srikanth
Assistant Professor
SNS College of Technology

V.Muralidharan
Assistant Professor
Christ the King Engineering College

A.Santhoshkumar
Assistant Professor
SNS College of Technology

***Abstract*- In this paper, we review recent developments in VLSI architectures and algorithms for efficient implementation of lifting based 2-D Discrete Wavelet Transform (DWT). Lifting scheme technique is used for Fast implementation of DWT. This method is entirely based on a spatial interpretation of the wavelet transform. In this paper, we provide a survey of the architectures for 2-dimensional DWT. The architectures are representative of many design styles and range from highly parallel architectures to DSP-based architectures to folded architectures**

***Keywords- Discrete Wavelet Transform, Lifting Scheme, MIMO,***

## I. INTRODUCTION

2-D DWT has evolved as essential part of modern compression system such as JPEG 2000. This is because the DWT can decompose the signals into different subbands with both time and frequency information and facilitate to arrive a high compression ratio [1]. In addition ,a wavelet based compression system, not only presents superior compression performance over DCT ,but provides four dimension of scalabilities resolution, distortion, spatial and color, which are very difficult to achieve in DCT based compression system. In a compression system, the function of DWT is to decorrelate the original image pixels prior to compression step such that they can be amenable to compression. The computation of DWT can be done either by convolution based scheme or Lifting based scheme. The lifting scheme of computation of DWT has, however, become more popular over the convolution-based scheme for its lower computational complexity [2] .The main feature of the lifting-based DWT scheme is to break up the high pass and low-pass filters into a sequence of upper and lower triangular matrices and covert the filter implementation into banded matrix multiplications. Such a scheme has several advantages, including "in-place" computation of DWT, integer-to integer wavelet transform, symmetric forward and inverse transform. The popularity of lifting-based DWT has triggered the development of several architectures in recent years. These architectures range from highly parallel architectures to programmable DSP-based architectures to folded architectures.

In this paper we present a survey of these architectures. We provide a systematic derivation of these architectures and comment on their hardware and timing requirements.

The rest of the paper is organized as follows. In Section II, We explained about the lifting based DWT. We also present a Comparison of the hardware and timing complexities of all the Architectures. In Section III, we present the memory configuration for 2-dimensional DWT architectures, followed by descriptions of a few representative architectures and a comparison of their hardware and timing complexities.

## II. LIFTING BASED DWT

In traditional convolution (filtering) based approach for computation of the forward DWT, the input signal (x) is filtered separately by a low-pass filter (˜ h ) and a high-pass filter ( ˜ g). The two output streams are then sub-sampled by simply dropping the alternate output samples in each stream to produce the low-pass (yL) and high-pass (yH) sub-band outputs.

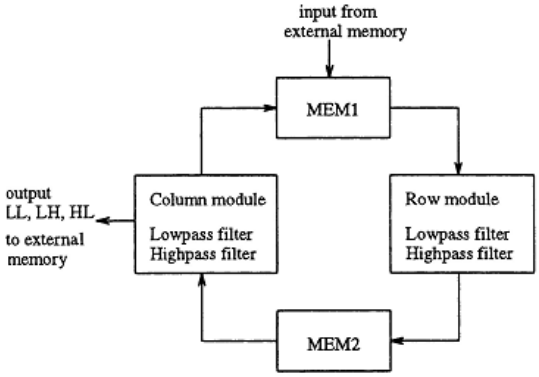
The lifting-based DWT has many advantages over the convolution based approach. Some of them are as follows.

- Lifting-based DWT typically requires less computation (up to 50%) compared to the convolution based approach. However the savings depends upon the length of the filters.
- During the lifting implementation, no extra memory buffer is required because of the in-place Computation feature of lifting. This is particularly suitable for hardware implementation with limited On-chip memory.
- The lifting based approach offers integer to integer transformation suitable for lossless image Compression.
- In lossless transformation mode, the boundary extension of the input data can be avoided because the original input can be exactly reconstructed by integer to integer lifting transformation.

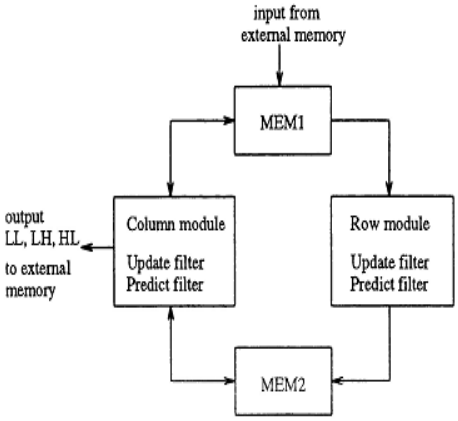
### *III. TWO DIMENSIONAL DWT ARCHITECTURE*

Generally, 2D wavelet filters are separable functions. A straight-forward approach for 2D implementation is to first apply the 1D DWT row-wise (to produce L and H sub bands) and then column-wise to produce four sub bands LL, LH, HL and HH in each level of decomposition. Obviously, the processor utilization is a concern in direct implementation of this approach because it requires all the rows be filtered before the column wise filtering can begin and thus it requires a size of memory buffer of the order of the image size.

The alternative approach is to begin the column-processing as soon as sufficient number of rows has been filtered. The column-wise processing is now performed on these available lines to produce wavelet coefficients row-wise. The overview of the two-dimensional architecture for convolution based DWT is shown in Fig1: (a). the row module reads the data from MEM1 performs DWT along the rows and writes the data into MEM2. The column module reads the data from MEM2 performs DWT along the columns and writes LL data to MEM1 and LH, HL, HH data to external memory.




**a) Convolution based Architecture**




**b) Lifting based Architecture**

**Fig 1: Overview of convolution based and Lifting based 2-DWT Architectures**

A similar approach can be implemented for the lifting scheme as well. The basic idea of lifting based approach for DWT implementation is to replace the parallel low-pass and high-pass filtering of traditional approach by a sequence of alternating smaller filters. The computations in each filter can be partitioned into prediction (dual lifting) and update (primal lifting) stages as shown in Fig1: (b). Here the row module reads the data from MEM1 performs the DWT along the rows (H and L) and writes the data into MEM2. The prediction filter of the column module reads the data from MEM2, performs column-wise DWT along alternate rows (HH and LH) and writes the data into MEM2 in [5] (and into MEM1 in [12]); the update filter of the column module reads the data from MEM2 in [5] (and MEM1 in [12]), performs column wise DWT along the remaining rows, and writes the LL data into MEM1 for higher octave computations and HL data to external memory. Note that this is a generic architectural flow and is the backbone of the existing 2D architectures. An important consideration in the design of 2D architectures is the memory configuration. A trade-off exists between the size of the internal memory and the frame memory access bandwidth. The size of the internal memory is again a function of the way the frame memory is scanned. In Section A, we describe the existing scanning techniques along the lines of [14], [13].Then we describe three representative 2D DWT architectures, namely, the dedicated architecture for the (4,2) filter [12], the generalized architecture [5] and the recursive architecture [8], and compare them with respect to hardware and timing complexities.

## *A. Memory Scan Techniques*

The memory scan techniques can be broadly classified into line-based scan, block-based scan and stripe-based scan. Though most of the existing architectures are based on line scan, we describe all three techniques to (possibly) facilitate development of new 2D DWT architectures.

- *Line-Based Scan*

In line-based scan, the scan order is raster scan. An internal line buffer of size LN is required, where N is the number of pixels in a row and L is the number of rows that are required for that particular filter. A line-based implementation of the traditional convolution based wavelet transform has been discussed in great detail in [10]. For lifting based architectures, the value of L can be determined as in [13] by considering the data dependency graph (Fig2). This is an extension of the 1D data dependency graph (Fig1) with a node now corresponding to a row of pixels. Note that several rows of data corresponding to R2-R4 and coefficients corresponding to R1 and R5 have to be stored. When a new row of data

corresponding to R6 is available, another column operation can be initiated. After this column operation, data in R7–R9 are stored for the next column operation. According to the implementation in [13], the line buffer needs to store six rows of data. The implementation in [14] as well as that in [11] requires only four rows of data to be stored. A detailed analysis of the memory requirements for line scan implementations of both forward and inverse transforms are presented in [7].

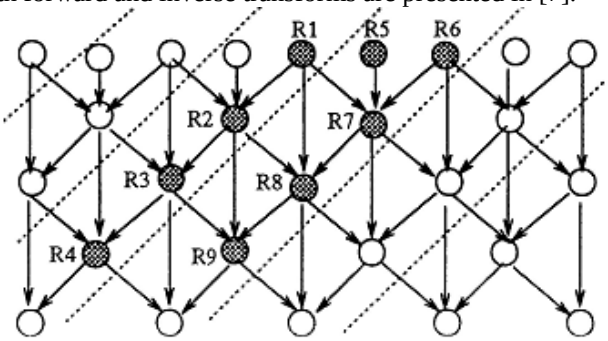



**Fig 2: Line based Scan**

- *Block-based Scan*

In block-based scan, the frame memory is scanned block-by-block and the DWT coefficients are also computed block-by-block. Figure 3 shows two configurations of block based methods where the blocks are scanned in the row direction first. In the non-overlapped configuration, the blocks are not overlapped with each other and in the overlapped configuration, the blocks are overlapped by 2K pixels in the column direction.

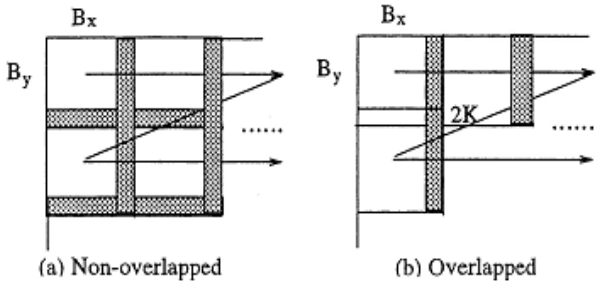



**Fig 3: Block Based Scan**

Here K = _(L − 1)/2_,where L is the number of DWT filter taps. In both cases, intermediate data have to be stored between two adjacent blocks as shown in grey in Fig. 3. The size of the internal buffer for one level for the non-overlapped case is LN + LBy. The first term, LN, is due to the column-wise intermediate data and the second term, LBy is due to the intermediate data between adjacent blocks in a row. The size of the internal buffer can be reduced to only LBy if the column-wise intermediate data is not stored and instead the data is read from the frame memory as needed. The size of the internal buffer for the overlapped case can also be reduced to LBy at the expense of increasing the number of frame memory reads to N2By/(By−2K) [14]. However, this scheme is not directly applicable to multi-level architectures. The block-based technique proposed in [11] first performs filtering operation on neighboring data blocks independently and later combines the partial boundary results together. Two boundary post-processing techniques are proposed - overlap-state sequential which reduces the buffer size for sequential processing and split-and-merge which reduces the interprocessor delay in parallel implementations.

- *Stripe-Based Scan*

The stripe-based scan is equivalent to the line-based scan with Bx = N. In other words, the stripe is a very wide block with width N and height S. As in the case of block-based scan, there are two categories, namely, the non-overlapped stripe based scan also referred to as the optimal Z-scan in [13] and shown in Fig. 4(a) and the overlapped stripe based scan shown in Fig. 4(b). The non-overlapped stripe-based scan has an internal buffer of size LN +LS and N2 frame memory READ accesses. In contrast, the overlapped stripe-based scan has a significantly smaller internal buffer of size LS and N2S/(S – 2K) frame memory READ accesses.

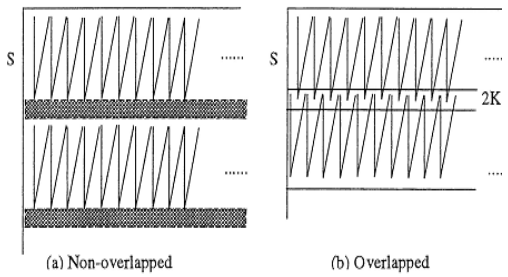



**Fig 4: Stripe Based Scan**

## *B. (4, 2) Filter Architecture*

A dedicated architecture for 2D DWT using the (4, 2) filter from the Deslauriers-Dubuc family has been proposed by Ferretti and Rizzo in [12]. The architecture is shown in Fig5. It consists of two parallel filters to compute the predict and update values along the rows (Pred-row, Upd-row), two parallel filters to compute the predict and update values along the columns, and four buffers A, B, C, D, to hold the intermediate data to support the pipelined computations. The buffers are dual-ported and are organized such that words can be accessed simultaneously.

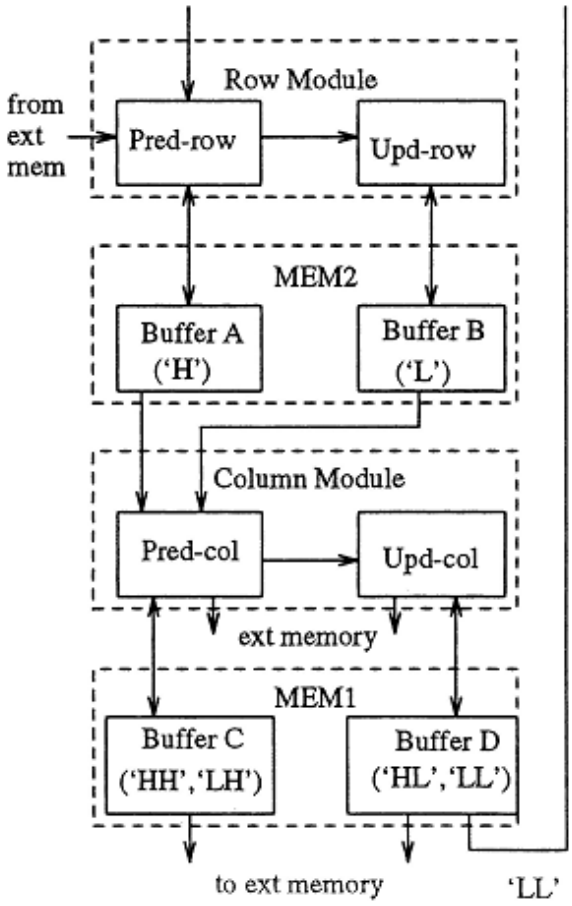



**Fig 5: Block diagram of (4, 2) Filter Architecture**

Each filter consists of multipliers (Lg = 4 for predict filters and Lh = 2 for update filters), adders, shifters and internal buffers (proportional to lg and Lh) to streamline the computations. Pred-row computes on Lg = 4 data, Lg − 1 of which are stored in its internal buffer. It computes the H values. The Upd-row requires LH = 2 „H" values to compute a L value. It obtains these by reading the last value produced by Pred-row and storing the other Lh − 1 in internal registers. It picks up the primary input value from the internal buffer in Pred-row.

Pred-col performs the same basic operations as Pred-row, though working on columns. It reads Lg even position H values along the columns. It produces a new row of wavelet coefficients for every two rows produces by Pred-row. During the time Pred-row produces H values for odd-indexed rows, Pred-col computes on the L values generated by Upd-row. The architecture utilization is only 50% if we only consider the computations in the first level. The higher level computations can thus be easily interspersed with the first level computations using a RPA-based approach. In fact, once the unit delay for any level is determined, the schedule can be easily obtained.

### *C. Generalized 2D DWT Architecture*

The architecture proposed by Andra et al. [8] Is more generalized and can compute a large set of filters for both the 2D forward and inverse transforms. It supports two classes of Architectures based on whether lifting is implemented by one or two lifting steps. The M2 architecture corresponds to implementation using one lifting step or two factorization matrices, and the M4 architecture corresponds to implementation by two lifting steps or four factorization matrices. The dataflow of the M2 architecture that is used to implement the wavelet filters (5,3), C(13,7), S(13,7), (2,6), (2, 10) is similar to that in Fig1: (b). A block diagram of the M2 architecture is shown in Fig. 6.

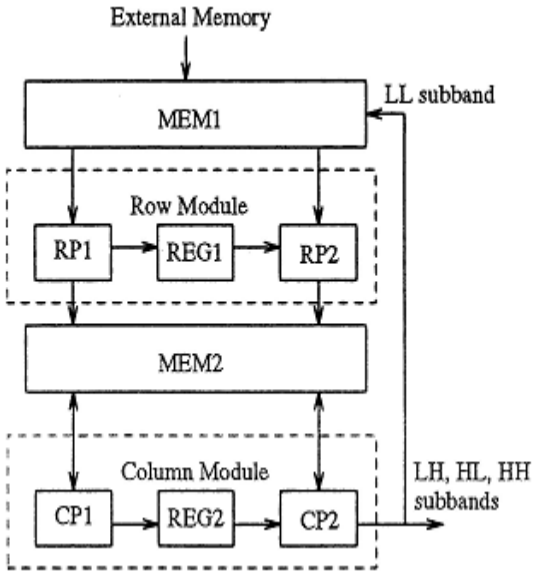



**Fig 6: Block Diagram Generalised M2 Architecture**

It consists of the row and column computation modules and two memory units, MEM1 and MEM2. The row module consists of two processors RP1 and RP2 along with a register file REG1, and the column module consists of two processors CP1 and CP2 along with a register file REG2. All the four processors RP1, RP2, CP1, CP2 in the proposed architecture consists of 2 adders, 1 multiplier and 1 shifter. For the M2 architecture, RP1 and CP1 are predicting filters and RP2 and CP2 are update filters.

The data access pattern for the (5, 3) filter with N = 5. RP1 calculates the high-pass (odd) elements along the rows, y01,y03, while RP2 calculates the low-pass (even) elements along the rows, y00, y02, y04. . . , CP1 calculates the high-pass and low pass elements z10, z11. . . z30, z31 . . . along odd rows and CP2 calculates high-pass and low-pass elements z00,

z01, . . . ; z20, z21, . . . ; z40, z41, . . . along the even rows. Note that CP1 and CP2 start computations as soon as the required elements are generated by RP1 and RP2.

The memory modules, MEM1 and MEM2, are both dual port with one read and one write port, and support two simultaneous accesses per cycle. MEM1 consists of two banks and MEM2 consists of four banks. The multi-bank structure increases the memory bandwidth and helps support highly pipelined operation. Details of the memory organization and size, register file, and schedule for the overall architecture with specific details for each constituent filter have been included in [5].

The dataflow of the M4 architecture that is used to implement the filters (9,7), (6,10) is quite different. Since this is a generalized architecture with the hardware in the row and column modules fixed, the computations span two passes. In the first pass, the row-wise computations are performed using both the modules. Module 1 reads the data from MEM1, executes the first two matrix multiplications, and writes the result into MEM2. Module 2 executes the next two matrix multiplications and writes the result into MEM1. In the second pass, the transform is computed along the columns. Once again, Module 1 executes the first two matrix multiplications and Module 2 executes the next two matrix multiplications.

### *D. 2D Recursive Architecture*

The 2D recursive architecture proposed by Liao et al. [8] is built by the 1D recursive architecture proposed by the same authors in [7,8]. As in the 1D case, the computations of all the lifting stages are interleaved to increase the hardware utilization. The column processor and the row-processor are similar to the1D recursive architecture processor. The image is input to the row-processor in raster scan format. When the row-processor processes the even rows, the high- and low-frequency DWT coefficients of the odd rows are shifted into their corresponding first-in first-out FIFO registers. The use of two FIFOs to separately store high frequency and low frequency components results in lower controller complexity. When the row processor processes the odd lines, the low-frequency DWT coefficients of the current line and lines previously stored in the FIFOs are sent to the column processors. The column-processor starts calculating the vertical DWT in zigzag scan format after one row delay. The computations are arranged in a way that the DWT coefficients of the first stage are interleaved with the other stages. The arrangement is done with the help of the data arrangement switches at the input to the row and column processors, and the exchange switch.

A mix of the principles of recursive pyramid algorithm (RPA) [6] and folded architecture has been adopted by Jung et to design a 2D architecture for lifting based DWT in [9].The row-processor is a 1D folded architecture and does row-wise computations in the usual fashion. The column processor is responsible for filtering along the columns at the first level and filtering along both the rows and the columns at the higher levels. It does this by employing RPA scheduling and achieves very high utilization. The utilization of the row processor is 100%, and that of the column processor is 83% for 5-level decomposition.

### *E. Parallel and Pipelined VLSI Architecture for Multilevel Lifting 2-D DWT*

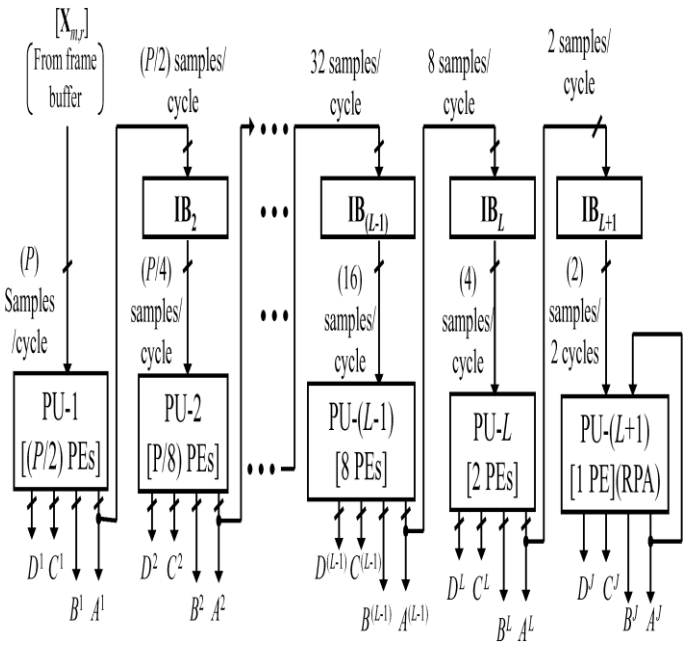



**Fig 7: Structure for J-Level Lifting 2-D Dwt**

According to the pyramid-algorithm (PA), the low-low sub band of a given decomposition level is processed further to generate DWT coefficients of the next higher level, and due to down-sampling, after each level of decomposition, the computational complexity steadily decrease by a factor of four. The amount of hardware resources required to calculate the DWT coefficients of every higher-level of decomposition should, therefore, be reduced by a factor of 4, in order to achieve 100% HUE. The scalable parallel structure [15] for multilevel DWT, based on the above view point, is shown in Fig 7. It is comprised of (L+1) PUs, where L= $[\log_4 P]$ The PU (L+1) is a RPA based structure, while all other PUs are simple PA units. In each cycle, PU-1 receives a block of samples of an input row and produces a block of component of a particular

row of sub band matrices $[C^1,D^1]$or$[B^1,A^1]$ One row of sub band $[A^1]$is obtained from PU-1 after a gap of R cycles.

| Architecture | Multiplier | Adder | On-chip memory | Computing time | control |
|---|---|---|---|---|---|
| (4,2) filter [17] | 10 | 8 | 25N+7[L-2] | $T_m+2Ta+T_s$ | Moderate |
| Generalized [5] | 4 | 8 | $N^2+34$ | $T_m$ | Moderate |
| Recursive[7] | 8 | 8 | 4N | $4\ T_{m+}8\ T_a$ | Complex |
| Folded Rpa [9] | 9 | 2 | 12N | $4T_m+4\ T_a$ | Complex |
| MIMO[16] | 4M | 8M | 10M+4N+ (MN/2) | $N^2/M$ | Simple |

**Table 1: Hardware, memory& Computation Time for various 2-D DWT Architectures**

Each block of sub band output [$A^1$] of PU-1 is split into two blocks of (P/4) samples each, and fed to PU-2 in each cycle, so that one complete row of [$A^1$] is fed in 2R cycles. The computation of PU- j, (for j>1) is similar to that of PU-1. Similar to PU-1PU-j can be designed for the computation of jth level DWT. It consists of ($P/2^{2j-1}$) number of PEs and calculates the j-th level DWT of an input block of ($P/2^{2j-2}$) samples in every cycle, where 1<j<L and L=[$\log_4$P]. Structures of PU-j are similar to that of PU-1, except that the size of each shift register of ID and MD of subcell-2 is $2^{j-1}$ words. Each of the (PU- j)s uses a separate input buffer (IB $_j$).The structure (IB $_j$) is shown in fig .It is comprised of two RAM units and two MULTIPLEXERS. (IB $_j$).receives N' components in each cycle and complete a row of $A^{j-1}$ in one cycle and complete one row in 2R' cycles.

## F. MIMO VLSI Architecture for 2-D Lifting-Based DWT

An efficient multi-input/multi-output VLSI architecture (MIMOA)[16] is constructed as shown in Fig.8, which meets the high processing speed requirement with controlled increase of hardware cost and simple control signals. High processing speed can be achieved when multiple row data samples are processed simultaneously. And time multiplexing technique is adopted to control the increase of the hardware cost for the MIMOA. Furthermore, the control signals are simple, since the regular architecture is a combination of simple single-input/single-output (SISO) modules and two-input/two-output (TITO) modules. It provides a variety of hardware implementations to meet different processing speed requirements by selecting different throughput rates.

## *G. Comparison of Performances*

A summary of the hardware, memory and timing requirements of a few representative architectures is presented in Fig 9 . The hardware complexity has been described in terms of data path components and internal memory size

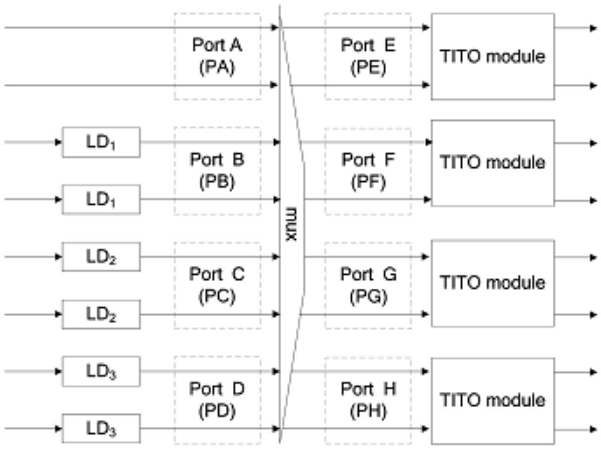



**Fig 8: MIMOA for 2D lifting-based DWT.**

We list only the internal memory size since all the architectures require an External memory of size N2 for input data of size N×N. The Timing performance has been compared with respect to the number of clock cycles to compute L levels of decomposition and the clock period.

Of the five architectures, the architecture in [8] has the smallest internal memory. This is because [8] is an RPA based approach that intersperses the computations at the higher levels with those of the lower levels. The architecture in [8], on the other hand, computes all the outputs of one level before starting the computations at the next level and has an internal memory of size O(N2). The data path complexity of the architecture in [8] is by far the lowest.

The control complexity of the architecture in [8] is significantly higher than the others. This is because of the large number of control signals and switches that are used to organize the data before sending to the row and column Computation units.

In terms of the timing performance, the architecture in [5] is pipelined and has the highest throughput (1/Tm). The

architecture in [8, 9] requires the fewest number of cycles since they are RPA based, though the clock periods are significantly higher.

The architecture in [12] is specific to the (4,2) filters while the RPA concept that is applied to the architectures in [8,9] can be applied to a large set of filters (not just (3,5), (9,7),Daub-4). The architecture in [9] is essentially a programmable architecture which supports implementation of a large set of filters on the same hardware platform.

## *IV. CONCLUSION*

In this paper, we presented a survey of the existing lifting based implementations of 2-dimensional Discrete Wavelet Transform. We briefly described the principles behind the lifting scheme in order to better understand the different implementation styles and structures. We have presented several architectures of different flavors ranging from highly parallel ones to highly folded ones to programmable ones.

## REFERENCES

[1]. S. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 11, no. 7, 1989, pp. 674–693.

[2] T. Acharya and P. S. Tsai, JPEG2000 Standard for Image Compression: Concepts, Algorithms and VLSI Architectures.John Wiley & Sons, Hoboken, New Jersey, 2004.

[3] W. Sweldens, "The Lifting Scheme: A Custom-Design Construction of Biorthogonal Wavelets," Applied and Computational Harmonic Analysis, vol. 3, no. 15, 1996, pp. 186–200.

[4] I. Daubechies and W. Sweldens, "Factoring Wavelet Transforms into Lifting Schemes," The J. of Fourier Analysis and Applications, vol. 4,1998, pp. 247–269.

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

[6] M. Vishwanath, "The Recursive Pyramid Algorithm for the Discrete Wavelet Transform," in IEEE Transactions on Signal Processing, vol. 42, 1994, pp. 673–676.

[7] H. Liao, M.K. Mandal, and B.F. Cockburn, "Novel Architectures for Lifting-Based Discrete Wavelet Transform," in Electronics Letters, vol. 38, no. 18, 2002, pp. 1010–1012.

[8] H. Liao, M.K. Mandal, and B.F. Cockburn, "Efficient Architectures for1-D and 2-D Lifting-BasedWavelet Transform," IEEE Transactions on Signal Processing, vol. 52, no. 5, 2004, pp.1315–1326.

[9] G.C. Jung, D.Y. Jin and S.M. Park, "An Efficient Line Based VLSI Architecture for 2-D Lifting DWT," in The 47th IEEE International Midwest Symposium on Circuits and Systems, 2004.

[10] C. Chrysafis and A. Ortega, "Line-Based, Reduced Memory, Wavelet Image Compression," IEEE Trans. on Image Processing, vol. 9, no. 3, 2000, pp. 378–389.

[11] W. Jiang and A. Ortega, "Lifting Factorization-Based Discrete Wavelet Transform Architecture Design," IEEE Trans, on Circuits and Systems for Video Technology, vol. 11, 2001, pp. 651– 657.

[12] M. Ferretti and D. Rizzo, "A Parallel Architecture for the 2-D Discrete Wavelet Transform with Integer Lifting Scheme," Journal of VLSI Signal Processing, vol. 28, 2001, pp. 165–185.

[13] M.Y. Chiu, K.-B. Lee, and C.-W. Jen, "Optimal Data Transfer and Buffering Schemes for JPEG 20000 Encoder," in Proceedings of the IEEE Workshop on Design and Implementation of Signal Processing Systems, 2003, pp. 177–182.

[14] C.-T. Huang, P.-C. Tseng, and L.-G. Chen, "Memory Analysis and Architecture for Two-Dimensional Discrete Wavelet Transform," in Proceedings of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing, 2004, pp. V13–V16.

[15] Meher, P. K. Mohanty, B. K. and Patra, J. C.( 2008) "Memory Efficient Modular VLSI Architecture for High throughput and Low-Latency Implementation of Multilevel Lifting 2-D DWT" . IEEE Transactions on Signal processing, vol. 59, no. 5, may 2011.

[16] Xin Tian , Lin Wu , Yi-Hua Tan "Efficient Multi-Input/Multi-Output VLSI Architecture for Two-Dimensional Lifting-Based Discrete Wavelet Transform," IEEE Transactions on Computers, vol. 60, no. 8, August 2011.

[17] M. Ferretti and D. Rizzo, "A Parallel Architecture for the 2-D Discrete Wavelet Transform with Integer Lifting Scheme," Journal of VLSI Signal Processing, vol. 28, 2001, pp. 165–185.