# An Efficient VLSI Architecture for Lifting-Based Flipping Discrete Wavelet Transform

P.V.Ram Prathap
Dept. of E.C.E
GMRIT
RAJAM, INDIA

S.Ranjan Kumar
Dept. of E.C.E
GMRIT
RAJAM, INDIA

*Abstract*— **This paper proposes an improved version of lifting-based Discrete Wavelet Transform (DWT). The modifications are made to the lifting scheme, and the intermediate results are recombined and stored to reduce the number of pipelining stages. As a result, the number of registers can be reduced to 18 without extending the critical path. In addition, the two-input/two-output parallel scanning architecture is adopted in our design. For a 2-D DWT with the size of *N* × *N*, the proposed architecture only requires three registers between the row and column filters as the transposing buffer, and a higher efficiency can be achieved.**

*Index Terms*— **Discrete wavelet transform (DWT), flipping structure, lifting-based, VLSI architecture.**

## Introduction

The discrete wavelet transform (DWT) was deduced by Mallat [1]. Many researches on wavelet-based signal analysis and compression have derived fruitful results due to the well time-frequency decomposition. The DWT has been adopted as the transform coder in emerging image coding standards, such as JPEG2000 still image coding and MPEG-4 still texture coding. However, more arithmetic operations may be required for the DWT than the discrete cosine transform (DCT) because of the filter computation. Contrary to the block-based DCT, the DWT is basically frame-based. The huge amount of the memory size and access bandwidth becomes a bottleneck of the implementation for two-dimensional (2-D) DWT [2].

For one-dimensional (1-D) DWT, several convolution-based architectures have been proposed [3],[4] because the DWT computation is intrinsically the filter convolution. After the appearance of the lifting scheme [3] and a factorization method of lifting steps [4], the lifting scheme has been widely used to reduce the computation of DWT and the control complexity of boundary extension. In [3] and [4] some lifting-based architectures have been proposed, which are all based on the direct implementation of the factorized lifting scheme.

The flipping structure is another important DWT architecture that was proposed by Huang *et al.* [10]. With a five-stage pipeline, the critical path can be also reduced to one multiplier. However, the flipping structure has a large temporal buffer, and fewer pipelining stages lead to longer critical path delay.

In this brief, further optimization on the lifting scheme is proposed to overcome shortages in previous works and minimize sizes of the logic units and the memory without loss of the throughput. By recombining the intermediate results of the row and column transforms, the number of pipelining stages and registers is reduced, while keeping the critical path delay as $Tm$. In addition, a novel architecture is developed to implement the 2-D DWT based on the above modified scheme. The parallel scanning method is employed to reduce the size of the transposing buffer. As a result, our design achieves higher efficiency.

## Proposed Algorithm

The lifting scheme was first proposed by Daubechies and Sweldens in 1996 [8], [9]. It shows that every finite-impulse response wavelet or filter bank can be factored into a cascade of lifting steps. That means the polyphase matrices for the wavelet filters can be decomposed into a sequence of alternating upper and lower triangular matrices multiplied by a diagonal normalization matrix

In order to optimize the critical path of the lifting –based implementation, a modified algorithm is employed by changing the coefficients in lifting formulas [14], shown as follows :

$$\frac{1}{\alpha}y(2n+1) = \frac{1}{\alpha}x(2n+1) + x(2n) + x(2n+2) \quad (1)$$
$$\frac{1}{\beta}y(2n) = \frac{1}{\beta}x(2n) + y(2n-1) + y(2n+1) \quad (2)$$
$$\frac{1}{\gamma}H(2n+1) = \frac{1}{\gamma}y(2n+1) + y(2n) + y(2n+2) \quad (3)$$
$$\frac{1}{\delta}L(2n) = \frac{1}{\delta}y(2n) + H(2n-1) + H(2n+1). \quad (4)$$

Based on (1)–(4), the flipping structure can achieve one multiplier delay by pipelining. However, the above flipping- based algorithm also shows obvious limitations. It needs the temporal buffer with the size of $11N$ to cache the intermediate data.

Substituting (1) into (2) and reordering the expression with the associative law, we can get

$$\begin{aligned}\frac{1}{\alpha\beta}y(2n) &= \frac{1}{\alpha\beta}x(2n) + \frac{1}{\alpha}y(2n-1) + \frac{1}{\alpha}y(2n+1)\\ &= \left[\left(\frac{1}{\alpha\beta}+1\right)x(2n) + \frac{1}{\alpha}x(2n-1) + x(2n-2)\right]\\ &\quad + \left[\frac{1}{\alpha}x(2n+1) + x(2n) + x(2n+2)\right]. \qquad (5)\end{aligned}$$

# Proposed architecture

There have been many VLSI architectures proposed for hardware implementation of DWT. For 1-D DWT, the architectures are mainly convolution-based and lifting-based. On the other hand, the direct and line-based architectures are the most feasible implementations for 2-D DWT. In this section, we will introduce these architectures and discuss their advantages

The lifting scheme provides many advantages, such as fewer arithmetic operations, in-place implementation, and easy management of boundary extension. According to [13], the above lifting factorization can further reduce the arithmetic operations over convolution-based architectures by exploring the redundancy between the low pass and high pass filters.

The efficiency of lifting scheme with respect to hardware cost and the complexity of control circuits have been proven. However, the potentially long critical path has not been discussed in literature.

The lifting factorization is essentially composed of a series of computing stages that correspond to the upper and lower triangular matrices. The computing unit of upper triangular matrices is shown in Fig. 1,.The computation node performs the summation of all input signals, the register node stores the data in the previous clock cycle, and the input node receives the coming data in the current clock cycle. Thus, a lifting-based architecture is a serial combination of such computing units, and the computation node of the previous computing stage is connected to the right input node of the next stage. The critical path of a lifting-based architecture would be the sum of the timing delay in each computing unit without pipelining

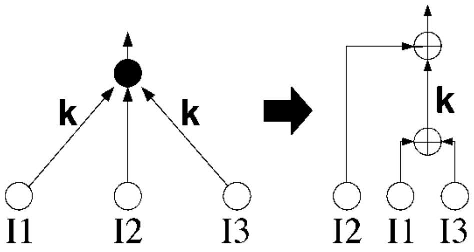


FIG: 1 Modified computing unit

Although pipelining can be used to reduce the critical path, the number of additional registers will increase more rapidly as the required critical path becomes shorter. The penalty of pipelining lifting-based architectures will become very critical for the implementation of line-based 2-D DWT because the number of registers in 1-D DWT can dominate the hardware design of line-based 2-D DWT, which will be described in the next subsection. To reduce the number of multipliers, each computing unit should be modified

### *Two–Dimensional DWT*

In general, RAM-based architectures can be classified into three categories: direct, line-based, and block-based methods [2]. The direct one is the most straightforward implementation method. It performs 1-D DWT in one direction and stores the intermediate coefficients in a frame memory first. Then, it performs 1-D DWT in the other direction with these intermediate coefficients to complete one-level 2-D DWT. Because the size of this frame memory is usually assumed to be off chip. However, the line-based method performs 1-D DWT in both directions simultaneously. Thus, the line-based method does not require a frame memory to store the intermediate data. Instead, some internal line buffers are used to store the intermediate data, and the required size is proportional to the image width. Contrary to the assumption that the input signals are in raster scan order for the above two methods, the block-based method performs 2-D DWT in a block-by-block way. This method can use some internal line buffers to store the boundary data among neighbor blocks such as to keep the required external frame memory bandwidth as low as the line-based method. Which method should be adopted depends on what kinds of hardware constraints are given. However, the external memory access would consume the most power

Although line-based architectures suffer from the requirement of internal line buffers, they can effectively reduce the external memory access and shorten the latency. There have been several line-based architectures proposed for the convolution-based hardware implementation of 2-D DWT

This architecture can be configured to perform 2-D DWT for several DWT filter banks. Furthermore, it uses the direct method for two lifting-stage filters and the line-based method for four lifting stage filters. However, this needs more general processing and memory units for configurable functions such that more registers and internal memory are required.

## Proposed Flipping Structure

The discrete wavelet transform (DWT) became a very versatile signal processing tool after Mallat proposed the multi-resolution representation of signals based on wavelet decomposition. The method of multi-resolution is to represent a function (signal) with a collection of coefficients, each of which provides information about the position as well as the frequency of the signal (function).

Since the timing problem is due to the accumulation of timing delays from the input node to the computation node in each computing unit, we suggest releasing the accumulation by eliminating the multipliers on the path from the input node to the computation node. This can be achieved by flipping each computing unit with the negation of the multiplier coefficient. There are also many alternatives for flipping structures. How many computing units should be flipped is case-by-case and dependent on hardware constraints.

DWT architectures are mostly based on the modified lifting scheme or the flipping structure. In order to achieve a critical path with only one multiplier, at least four pipelining stages are required for one lifting step, or a large temporal buffer is needed.

The input pixels arrive serially row-wise at one pixel per clock cycle and it will get split into even and odd. So after the manipulation with the lifting coefficients ‘a’ and ‘b’ is done, the low pass and high pass coefficients will be given out.

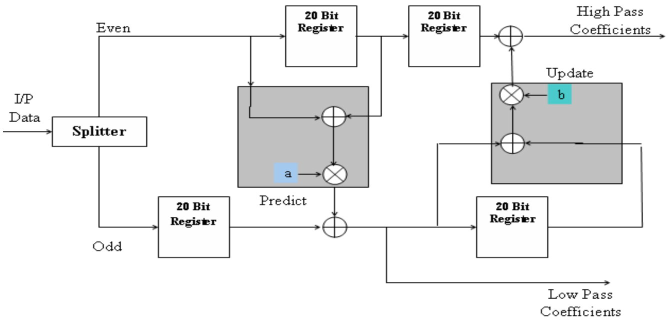


Fig.2 Modified lifting based flipping architecture

Hence for every pair of pixel values, one high pass and one low pass coefficients will be given as output respectively.

The internal operation of the DWT block has been explained above and hence the high pass and low pass coefficients of the taken image were identified and separated. The generated low pass and high pass coefficients are stored in buffers for further calculations.

## Simulation analysis and comparison

Tables I and II show comparisons of the proposed architecture with several previous designs for the 2- and 1-D 9/7 DWTs. As for the 2-D architecture, the RAM-based architecture [14] reduces the size of the transposing buffer to 1.5$N$ . However, the critical path delay is not optimized, and it requires a large memory with the size of 4$N$. In the fast architecture (FA) and the high-speed architecture (HA) [13], the hardware requirements are reduced by reusing the predictor and updater circuits, while it needs the feedback results of the predicting process for the updating process. Thus, the pipelined structure cannot be achieved, and the critical path delay is $Tm + 2Ta$. This design reduces the transposing buffer to two 20-bit register. However, the critical path delay of $4Tm + 8Ta$ limits its applications. The modified flipping structure with a critical path of one multiplier and the temporal buffer with 4$N$ size, while its transposing buffer size is 1.5$N$. Cheng and Parhi [15] is Proposed a VLSI architecture that has high speed at the cost of large hardware requirements. If the parallel level of the 9/7 filter is one, the computation time will be $N^2/2$.

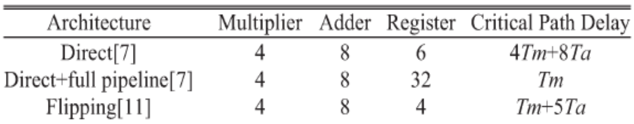
Moreover, its critical patch delay is $Tm + 2Ta$. In [16], a two-input/two-output architecture based on the flipping structure was proposed. The design reconstructs the flipping structure by substituting the traditional five-stage flipping structure with a two-stage structure. All multipliers are included in the first stage, and adders are included in the second stage. This way, the number of registers can be reduced. However, there are accumulated operations in the second stage.

As for the 1-D architecture, the flipping structure [14] provides a new method to shorten the delay. The critical path is dominated by $Tm + 5Ta$, and the 1-D DWT only needs four registers. When this design is pipelined into five stages, the critical path delay is $Tm$. Nevertheless, if this 1-D processor is adopted to construct the 2-D DWT, each intermediate variable of the column transform must be stored, and the five-stage pipelined architecture needs 10 registers. For an image of $N \times N$, the size of the temporal buffer will reach $11N$. A different pipelined architecture was proposed in [17] by Wu and Wang. Except that the multiplication is replaced by shifted additions, this architecture does not make any modifications on the lifting algorithm.

Based on our modified lifting scheme, the proposed two-input/two-output 2-D DWT architecture is implemented with a parallel scanning method. The one step lifting circuit adopts three pipelining stages, and the number of registers is 7 with the critical path delay of $Tm$. For the 2-D DWT architecture, the size of the temporal buffer is $4N$, and the transposing module has only two multiplexers and three registers. At the same time, the computation time for an $N \times N$ *image* is $N^2/2$, and the throughput of our design reaches $2/Tm$.

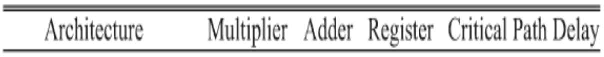
TABLE I

PERFORMANCE COMPARISON OF ONE-LEVEL 1-D DWT ARCHITECTURES FOR 9/7 FILTER



| Architecture | Multiplier | Adder | Register | Critical Path Delay |
|---|---|---|---|---|
| Direct[7] | 4 | 8 | 6 | $4Tm+8Ta$ |
| Direct+full pipeline[7] | 4 | 8 | 32 | $Tm$ |
| Flipping[11] | 4 | 8 | 4 | $Tm+5Ta$ |

Based on modified lifting based flipping structure



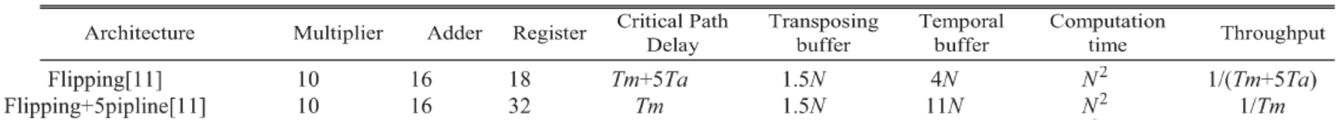
| Architecture | Multiplier | Adder | Register | Critical Path Delay |
|---|---|---|---|---|
| Direct | 2 | 4 | 4 | 3tm+6ta |
| Flipping | 2 | 3 | 4 | 2tm+6ta |

Transposing buffer is the element which is used to store and transfer the pixel data for processing.

TABLE II

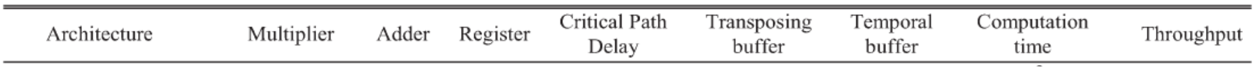
PERFORMANCE COMPARISON OF ONE–LEVEL 2-D DWT ARCHITECTURES FOR 9/7 FILTERWITH $N \times N$ IMAGE SIZE

($Ta$: AN ADDER DELAY, $Tm$: A MULTIPLIER DELAY, $S$: PARALLEL FACTOR, $L$: BIT WIDTH OF INPUT DATA)



| Architecture | Multiplier | Adder | Register | Critical Path Delay | Transposing buffer | Temporal buffer | Computation time | Throughput |
|---|---|---|---|---|---|---|---|---|
| Flipping[11] | 10 | 16 | 18 | $Tm+5Ta$ | $1.5N$ | $4N$ | $N^2$ | $1/(Tm+5Ta)$ |
| Flipping+5pipline[11] | 10 | 16 | 32 | $Tm$ | $1.5N$ | $11N$ | $N^2$ | $1/Tm$ |

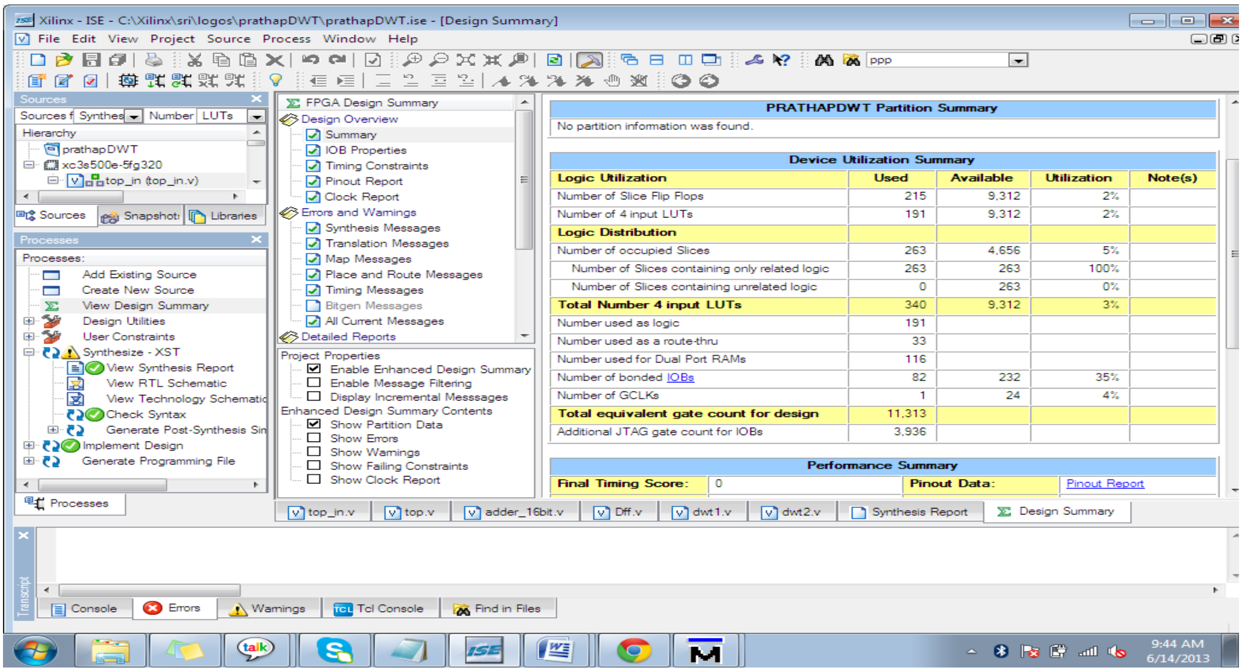
BASED ON MODIFIED LIFTING BASED FLIPPING STRUCTURE



| Architecture | Multiplier | Adder | Register | Critical Path Delay | Transposing buffer | Temporal buffer | Computation time | Throughput |
|---|---|---|---|---|---|---|---|---|
| Flipping | 6 | 10 | 17 | tm+3ta | 1.3N | 2N | $N^2$ | 1/(Tm+3Ta) |

## Device utilization summary:

## Throughput output in XLIINX



Project Properties
☑ Enable Enhanced Design Summary
☐ Enable Message Filtering
☐ Display Incremental Messsages
Enhanced Design Summary Contents
☑ Show Partition Data
☐ Show Errors
☐ Show Warnings
☐ Show Failing Constraints
☐ Show Clock Report

**PRATHAPDWT Partition Summary**

No partition infomation was found.

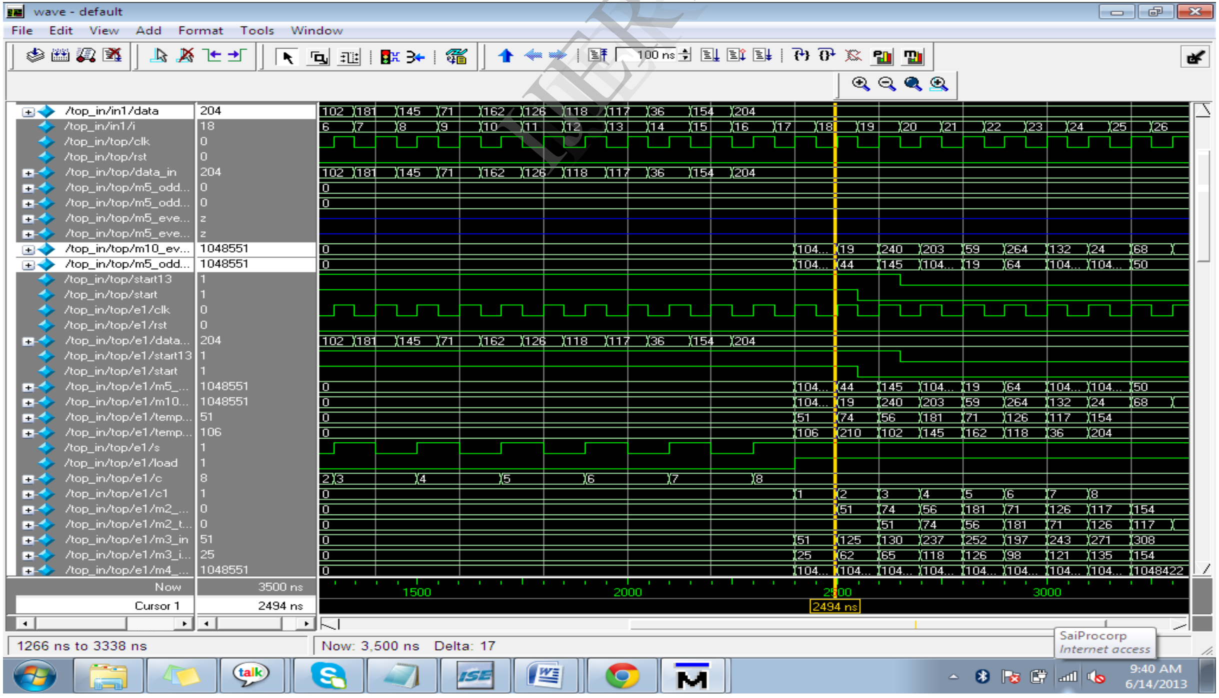
**Device Utilization Summary**

| Logic Utilization | Used | Available | Utilization | Note(s) |
|---|---|---|---|---|
| Number of Slice Flip Flops | 215 | 9,312 | 2% | |
| Number of 4 input LUTs | 191 | 9,312 | 2% | |
| **Logic Distribution** | | | | |
| Number of occupied Slices | 263 | 4,656 | 5% | |
| Number of Slices containing only related logic | 263 | 263 | 100% | |
| Number of Slices containing unrelated logic | 0 | 263 | 0% | |
| **Total Number 4 input LUTs** | 340 | 9,312 | 3% | |
| Number used as logic | 191 | | | |
| Number used as a route-thru | 33 | | | |
| Number used for Dual Port RAMs | 116 | | | |
| Number of bonded IOBs | 82 | 232 | 35% | |
| Number of GCLKs | 1 | 24 | 4% | |
| **Total equivalent gate count for design** | 11,313 | | | |
| Additional JTAG gate count for IOBs | 3,936 | | | |

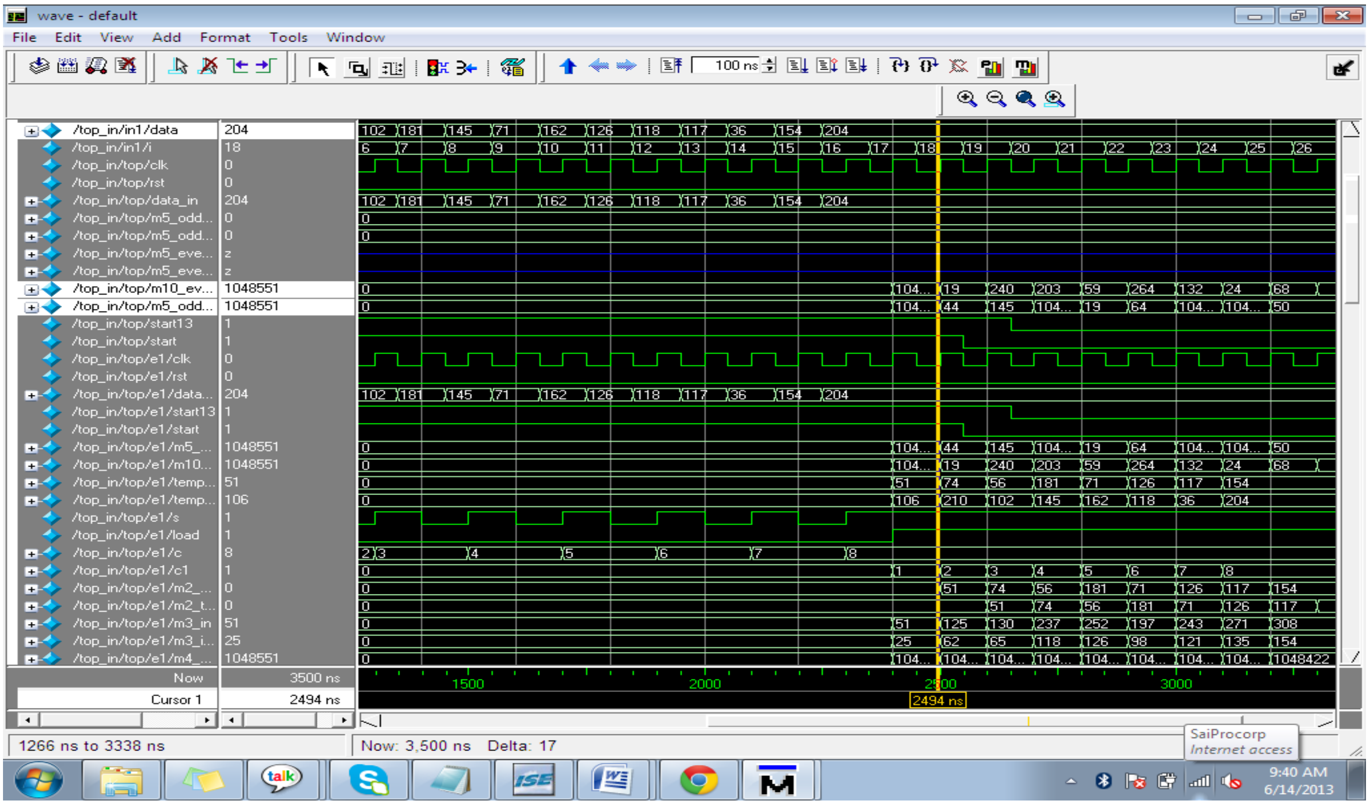
**Performance Summary**

| Final Timing Score: | 0 | Pinout Data: | Pinout Report |
|---|---|---|---|

## Output Waveform In Model Sim



```
/top_in/in1/data        204
/top_in/in1/i           18
/top_in/top/clk         0
/top_in/top/rst         0
/top_in/top/data_in     204
/top_in/top/m5_odd...   0
/top_in/top/m5_odd...   0
/top_in/top/m5_eve...   z
/top_in/top/m5_eve...   z
/top_in/top/m10_ev...   1048551
/top_in/top/m5_odd...   1048551
/top_in/top/start13     1
/top_in/top/start       1
/top_in/top/e1/clk      0
/top_in/top/e1/rst      0
/top_in/top/e1/data...  204
/top_in/top/e1/start13  1
/top_in/top/e1/start    1
/top_in/top/e1/m5_...   1048551
/top_in/top/e1/m10...   1048551
/top_in/top/e1/temp...  51
/top_in/top/e1/temp...  106
/top_in/top/e1/s        1
/top_in/top/e1/load     1
/top_in/top/e1/c        8
/top_in/top/e1/c1       1
/top_in/top/e1/m2_...   0
/top_in/top/e1/m2_t...  0
/top_in/top/e1/m3_in    51
/top_in/top/e1/m3_i...  25
/top_in/top/e1/m4_...   1048551
                Now     3500 ns
           Cursor 1     2494 ns
1266 ns to 3338 ns
Now: 3,500 ns   Delta: 17
```



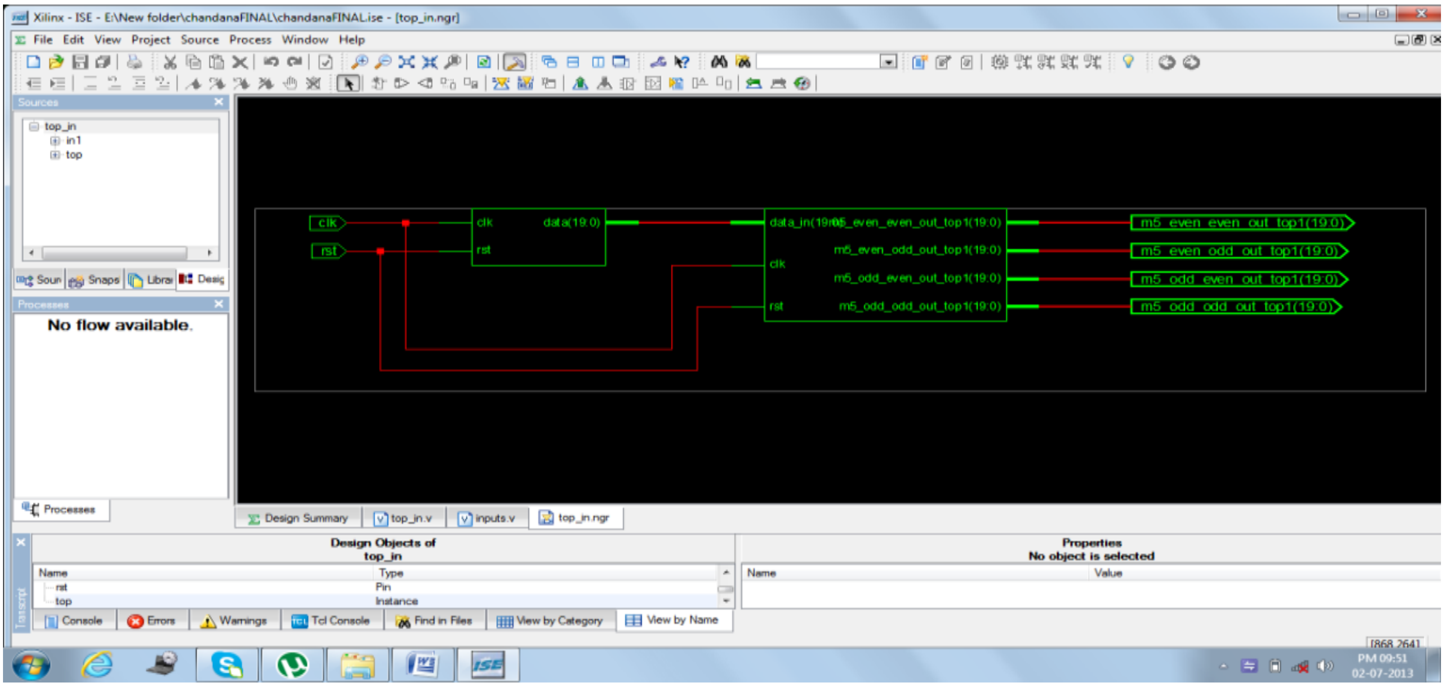
| | |
|---|---|
| /top_in/in1/data | 204 |
| /top_in/in1/i | 18 |
| /top_in/top/clk | 0 |
| /top_in/top/rst | 0 |
| /top_in/top/data_in | 204 |
| /top_in/top/m5_odd... | 0 |
| /top_in/top/m5_odd... | 0 |
| /top_in/top/m5_eve... | z |
| /top_in/top/m5_eve... | z |
| /top_in/top/m10_ev... | 1048551 |
| /top_in/top/m5_odd... | 1048551 |
| /top_in/top/start13 | 1 |
| /top_in/top/start | 1 |
| /top_in/top/e1/clk | 0 |
| /top_in/top/e1/rst | 0 |
| /top_in/top/e1/data... | 204 |
| /top_in/top/e1/start13 | 1 |
| /top_in/top/e1/start | 1 |
| /top_in/top/e1/m5_... | 1048551 |
| /top_in/top/e1/m10... | 1048551 |
| /top_in/top/e1/temp... | 51 |
| /top_in/top/e1/temp... | 106 |
| /top_in/top/e1/s | 1 |
| /top_in/top/e1/load | 1 |
| /top_in/top/e1/c | 8 |
| /top_in/top/e1/c1 | 1 |
| /top_in/top/e1/m2_... | 0 |
| /top_in/top/e1/m2_t... | 0 |
| /top_in/top/e1/m3_in | 51 |
| /top_in/top/e1/m3_i... | 25 |
| /top_in/top/e1/m4_... | 1048551 |
| Now | 3500 ns |
| Cursor 1 | 2494 ns |

1266 ns to 3338 ns

Now: 3,500 ns  Delta: 17

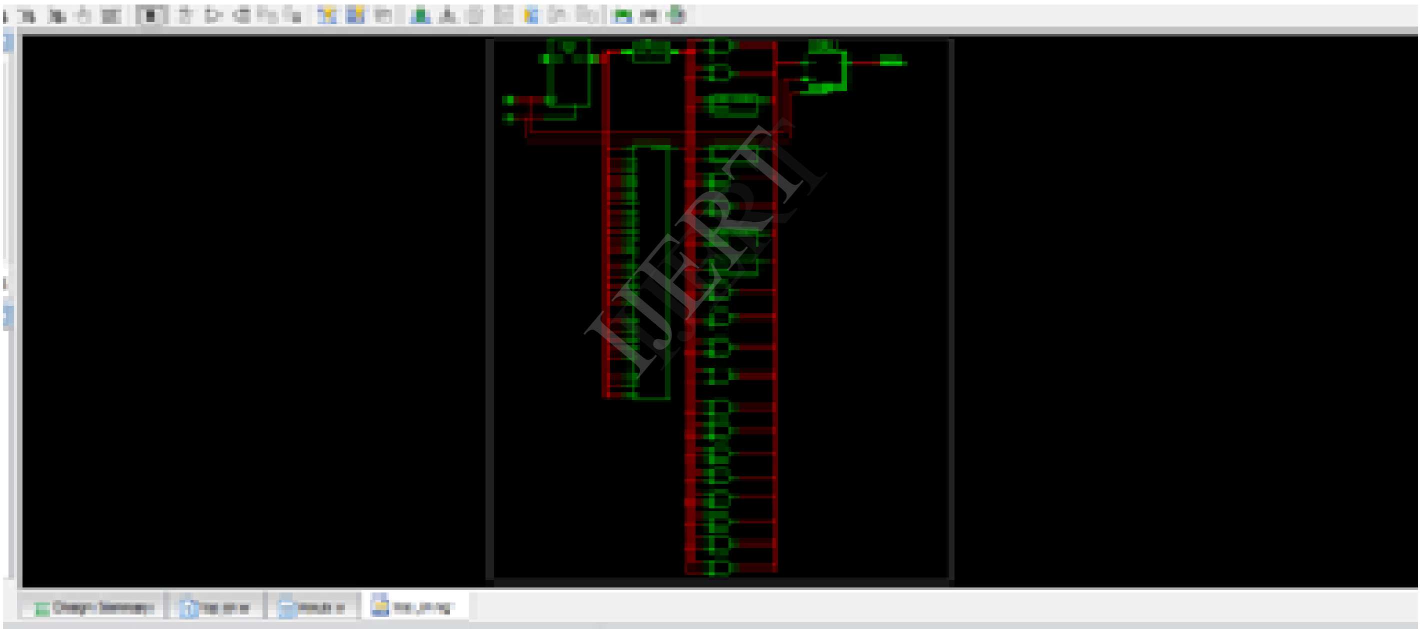
## RTL Schematics

This is first of all simulation steps; those are encountered throughout the hierarchy of the design flow. This simulation is performed before synthesis process to verify RTL (behavioral) code and to confirm that the design is functioning as intended. Behavioral simulation can be performed on Verilog designs. In this process, signals and variables are observed, procedures and functions are traced and breakpoints are set. This is a very fast simulation and so allows the designer to change the HDL code if the required functionality is not met with in a short time period. Since the design is not yet synthesized to gate level, timing and resource usage properties are still unknown.



Xilinx - ISE - E:\New folder\chandanaFINAL\chandanaFINAL.ise - [top_in.ngr]

No flow available.



## Conclusion

In this architecture for the 1- and 2-D DWTs, the modified one lifting step circuit can work within three pipelining stages with fewer registers, and the critical path delay is $Tm$. By flipping some computing units with the inverses of multiplier coefficients, the critical path can be greatly reduced.

For the 2-D DWT architecture, the modified lifting circuit can work within three pipelining stages with fewer registers, and the critical path delay is $Tm$. For the DWT architecture, only the temporal buffer with $4N$ size is used in the column filter. A detailed analysis is performed to compare the proposed architectures with other previous architectures in terms of hardware complexity, critical path delay, storage size, computation time, and throughput. According to the results, the proposed architecture can achieve high speed with lower hardware complexity and smaller storage size.

# References

[1] S. G. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, pp. 674–693, July 1989.

[2] N. D. Zervas, G. P. Anagnostopoulos, V. Spiliotopoulos, Y. An- dreopoulos, and C. E. Goutis, "Evaluation of design alternatives for the 2-D-discrete wavelet transform," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, pp. 1246–1262, Dec. 2001.

[3] W. Sweldens, "The lifting scheme: a custom-design construction of biorthogonal wavelets," *Applied Comput. Harmon. Anal.*, vol. 3, no. 15, pp. 186–200, 1996.

[4] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *Journal Fourier Anal. Applicat.*, vol. 4, pp. 247–269,1998.

[5] K. K. Parhi and T. Nishitani, "VLSI architectures for discrete wavelet transforms," *IEEE Trans. Very Large Scale Integration Syst.*, vol. 1, pp.191–202, June 1993.

[6] C. Chakrabarti, M. Vishwanath, and R. M. Owens, "Architectures for wavelet transforms: A survey," *J. VLSI Signal Process.*, vol. 14, pp.171–192, 1996.

[7] J.-M. Jou, Y.-H. Shiau, and C.-C. Liu, "Efficient VLSI architectures for the biorthogonal wavelet transform by filter bank and lifting scheme," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 2, 2001, pp. 529–532.

[8] K. Andra, C. Chakrabarti, and T. Acharya, "A VLSI architecture for lifting-based forward and inverse wavelet transform," *IEEE Trans. Signal Processing*, vol. 50, pp. 966–977, Apr. 2002.

[9] C.-T. Huang, P.-C. Tseng, and L.-G. Chen, "Efficient VLSI architec- tures of lifting-based discrete wavelet transform by systematic design method," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 5, 2002, pp. 565–568.

[10] W. Jiang and A. Ortega, "Lifting factorization-based discrete wavelet transform architecture design," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, pp. 651–657, May 2001.

[11] G. Xing, J. Li, and Y. Q. Zhang, "Arbitrarily shaped video-object coding by wavelet," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 10, pp. 1135–1139, Oct. 2001.

[12] S. C. B. Lo, H. Li, and M. T. Freedman, "Optimization of wavelet decom- position for image compression and feature preservation," *IEEE Trans.Med. Imag.*, vol. 22, no. 9, pp. 1141–1151, Sep. 2003.

[13] J. M. Jou, Y. H. Shiau, and C. C. Liu, "Efficient VLSI architectures for the biorthogonal wavelet transform by filter bank and lifting scheme," in *Proc. IEEE ISCAS*, May 2001, vol. 2, pp. 529–532.

[14] C.-T. Huang, P.-C. Tseng, and L.-G. Chen, "Flipping structure: An efficient VLSI architecture for lifting-based discrete wavelet trans- form," *IEEE Trans. Signal Process.*, vol. 52, no. 4, pp. 1080–1089, Apr. 2004.

[15] C. Cheng and K. K. Parhi, "High-speed VLSI implement of 2-D discrete wavelet transform," *IEEE Trans. Signal Process.*, vol. 56, no. 1, pp. 393– 403, Jan. 2008.

[16] J. Song and I.-C. Park, "Novel pipelined DWT architecture for dual-line scan," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2009, pp. 373–376.

[17] Z. G.Wu andW.Wang, "Pipelined architecture for FPGA implementation of lifting-based DWT," in *Proc. Int. Conf. Elect. Inform. Control Eng.*,2011, pp. 1535–1538.

[18] W. Sweldens, "The new philosophy in biorthogonal wavelet construc- tions," in *Proc. SPIE.*, 1995, vol. 2569, pp. 68–79.

[19] B. F. Wu and C. F. Lin, "A high-performance and memory-efficient pipeline architecture for the 5/3 and 9/7 discrete wavelet transform of JPEG2000 codec," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 12, pp. 1615–1628, Dec. 2005.